

Multi-Structured Models for Transforming and Aligning Text

Kapil Thadani

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2015

©2015
Kapil Thadani
All Rights Reserved

ABSTRACT

Multi-Structured Models for Transforming and Aligning Text

Kapil Thadani

Structured representations are ubiquitous in natural language processing as both the product of text analysis tools and as a source of features for higher-level problems such as text generation. This dissertation explores the notion that different structured abstractions offer distinct but incomplete perspectives on the meaning encoded within a piece of text. We focus largely on monolingual *text-to-text generation* problems such as sentence compression and fusion, which present an opportunity to work toward general-purpose statistical models for text generation without strong assumptions on a domain or semantic representation. Systems that address these problems typically rely on a single structured representation of text to assemble a sentence; in contrast, we examine joint inference approaches which leverage the expressive power of heterogeneous representations for these tasks.

These ideas are introduced in the context of supervised sentence compression through a compact integer program to simultaneously recover ordered n-grams and dependency trees that specify an output sentence. Our inference approach avoids cyclic and disconnected structures through flow networks, generalizing over several established compression techniques and yielding significant performance gains on standard corpora. We then consider the tradeoff between optimal solutions, model flexibility and runtime efficiency by targeting the same objective with approximate inference techniques as well as polynomial-time variants which rely on mildly constrained interpretations of the compression task.

While improving runtime is a matter of both theoretical and practical interest, the flexibility of our initial technique can be further exploited to examine the multi-structured hypothesis under new structured representations and tasks. We therefore investigate extensions to recover directed acyclic graphs which can represent various notions of predicate-

argument structure and use this to experiment with frame-semantic formalisms in the context of sentence compression. In addition, we generalize the compression approach to accommodate multiple input sentences for the sentence fusion problem and construct a new dataset of natural sentence fusions which permits an examination of challenges in automated content selection. Finally, the notion of multi-structured inference is considered in a different context—that of monolingual phrase-based alignment—where we find additional support for a holistic approach to structured text representation.

Table of Contents

List of Figures	vi
List of Tables	ix
Acknowledgments	xvi
1 Introduction	1
1.1 Multi-Structured Inference	3
1.2 Tasks	5
1.3 Broader Impact	6
1.4 Contributions	8
1.5 Overview	8
2 Background on Tasks	11
2.1 Sentence Compression	12
2.1.1 Related work	13
2.2 Sentence Fusion	15
2.2.1 Related work	16
2.3 Text Alignment	17
2.3.1 Related work	19
2.4 Other Related Tasks	20
2.4.1 Paraphrase generation	20
2.4.2 Sentence simplification	20
2.4.3 Title generation	21

2.4.4	Machine translation	22
3	Multi-Structured Compression	23
3.1	Compression Corpora	24
3.1.1	Corpus analysis	26
3.2	Multi-Structured Compression	30
3.2.1	Compression as linear optimization	31
3.2.2	Multi-structured objective	32
3.3	Compression via Integer Linear Programming	34
3.3.1	Enforcing tree structure	36
3.3.2	Assembling valid n-gram factorizations	41
3.3.3	Enforcing a compression rate	44
3.4	Features	46
3.4.1	Feature categories	46
3.4.2	Token features	48
3.4.3	n-gram features	49
3.4.4	Dependency features	49
3.5	Parameter Estimation	50
3.5.1	Structured perceptron	51
3.5.2	Deriving features for reference compressions	51
3.6	Experiments	54
3.6.1	Joint inference	56
3.6.2	Content-bearing words	60
3.6.3	Example output	62
3.6.4	Varying the compression rate	65
3.6.5	Higher-order n-grams	68
3.6.6	Subtree deletion	70
3.7	Remarks	72
4	Approximation Strategies for Compression	74
4.1	Compression via Lagrangian Relaxation	75

4.1.1	Decomposing the inference task	76
4.1.2	Bigram paths	78
4.1.3	Dependency subtrees	79
4.1.4	Scoring approximate solutions	81
4.2	Experiments	82
4.2.1	Tightness of approximations	82
4.2.2	Tradeoff between structural solutions	85
4.2.3	Compression quality	87
4.2.4	Timing	90
4.3	Remarks	94
5	Efficient Compression via Dynamic Programming	96
5.1	Compressive Parsing	97
5.1.1	Edge-factored parsing	99
5.1.2	Bigram-factored compressions	100
5.1.3	Second-order parsing	103
5.1.4	Enforcing compression rates	106
5.2	Features	110
5.2.1	Second-order dependency features	111
5.3	Experiments	111
5.3.1	Compression quality	111
5.3.2	Timing	113
5.3.3	Second-order dependencies	116
5.4	Remarks	118
6	Compression over Predicate-Argument Structures	120
6.1	Structured Semantic Graphs	122
6.1.1	Multi-structured objective	124
6.1.2	Enforcing DAG structure	125
6.1.3	Constraining concept lexicons	128
6.1.4	Preserving frame semantics in compression	129

6.2	Features	132
6.2.1	Frame features	132
6.2.2	FE features	132
6.3	Experiments	133
6.3.1	Compression quality	133
6.3.2	Frame-semantic integrity	137
6.4	Remarks	138
7	Multi-Structured Sentence Fusion	140
7.1	Pyramid Fusion Corpus	142
7.2	Multi-Structured Fusion	144
7.2.1	ILP formulation	145
7.2.2	Redundancy	146
7.2.3	Dependency orientation	149
7.3	Features	151
7.3.1	Token features	151
7.3.2	Bigram and dependency features	152
7.3.3	Deriving features for reference fusions	152
7.4	Experiments	153
7.4.1	Fusion quality	154
7.4.2	Example output	157
7.4.3	Content selection	162
7.4.4	Dependency orientation	164
7.5	Remarks	165
8	Multi-Structured Monolingual Alignment	167
8.1	Aligned Paraphrase Corpus	168
8.1.1	Corpus analysis	170
8.2	Multi-Structured Alignment	172
8.2.1	Alignment as linear optimization	172
8.2.2	Multi-structured objective	173

8.2.3	Inference via ILP	175
8.3	Features	178
8.3.1	Phrase alignment features	178
8.3.2	Edge matching features	180
8.4	Experiments	180
8.4.1	Confident alignments	181
8.4.2	All alignments	182
8.5	Remarks	183
9	Conclusions	185
9.1	Limitations	188
9.1.1	Datasets	188
9.1.2	Features	189
9.1.3	Representations	190
9.1.4	Learning algorithms	191
9.1.5	Evaluation	191
9.2	Future Work	192
9.2.1	Unifying text-to-text operations	192
9.2.2	Direct applications	193
9.2.3	Task-based evaluations	195
9.2.4	Multi-task learning	195
	Bibliography	197
	A Glossary of Symbols	213

List of Figures

2.1	An example of phrase-based monolingual alignment drawn from the aligned paraphrase corpus of Cohn et al. (2008). Solid lines indicate SURE alignments while dashed lines indicate POSSIBLE alignments.	18
3.1	Distribution of instances in the BN training dataset with respect to the number of tokens dropped from the input sentence to produce (a) the longest reference compression, (b) the reference compression of median length, and (c) the shortest reference compression.	27
3.2	Distribution of instances in the WN training dataset with respect to the number of tokens dropped from the input sentence to produce the reference compression.	28
3.3	Dependency commodity values for a flow network accompanying a tree-based compression solution. Dashed lines denote all non-zero flow variables γ_{ij}	38
3.4	An illustrative flow network with edge weights indicating non-zero flow featuring (a) consistent flow and no directed cycles, (b) a cycle that preserves flow but needs multiple incoming edges (c) a cycle with one incoming edge for each node but consequently inconsistent flow.	40
3.5	Adjacency commodity values for a flow network accompanying a path-based compression solution. Dashed lines denote all non-zero flow variables γ'_{ij}	43
3.6	Variation in RASP F_1 with imposed compression rate for the BN corpus. All datapoints plotted at average output compression rates after rounding down to token counts.	67

4.1	An example of a weighted directed graph (a) whose maximum spanning tree (b) does not contain the maximum-weight subtree (c). Missing edges in (a) have weight $-\infty$	80
4.2	Variation in RASP F_1 with ψ on the BN and WN development datasets. As defined in (4.1), high values of ψ amplify the influence of the exact bigram solution \mathbf{y} on the token solution \mathbf{x} while low values of ψ attenuate it in favor of the approximate dependency solution \mathbf{z} . Highlighted datapoints indicate maxima and selected ψ values for later experiments.	86
4.3	Average difference between ILP-2gr-dep and DP-2gr+LP-dep in terms of inference time (above) and RASP F_1 (below) plotted against input sentence length over the BN test dataset.	92
4.4	Average difference between ILP-2gr-dep and DP-2gr+LP-dep in terms of inference time (above) and RASP F_1 (below) plotted against input sentence length over the WN test dataset.	93
5.1	Examples of unlabeled dependency analyses with (a) a non-projective parse drawn from McDonald et al. (2005b), and (b) a projective parse with no crossing edges.	98
5.2	An example derivation for edge-factored compressive parsing. Dependencies are indicated by solid lines and established by \square and \square items while bigrams are indicated by dashed lines and defined whenever \boxtimes items are converted to \triangleleft items.	102
5.3	An example derivation for second-order compressive parsing. Dependencies are indicated by solid lines and established by \square and \square items while bigrams are indicated by dashed lines and defined whenever \boxtimes items are converted to \triangleleft or \square items.	105
6.1	A frame-semantic parse of an example sentence using frames from the FrameNet lexicon (Fillmore et al., 2003). Boldfaced words evoke frames through dotted lines. Frame elements (FEs) are denoted with labeled arrows to head words of the corresponding text spans.	122

6.2	An illustrative flow network with edge weights indicating non-zero flow (a) permitting an undesired cycle when imposing the §3.2 constraint $\sum_i \gamma''_{ij} - \sum_k \gamma''_{jk} = x_j^+$ for all nodes, (b) & (c) constrained to acyclic structures when using revised constraints (6.6)–(6.7).	126
6.3	Frame-semantic relations produced by the SEMAFOR semantic parser (Das et al., 2013) over an example sentence and a possible compression. Dotted edges indicate the lexical units which evoke each frame while frame elements (FEs) are denoted by labeled edges to head words of the corresponding lexical units.	131
7.1	Distribution of instances in the pyramid fusion corpus constructed according to §7.1 with respect to the DUC or TAC dataset that they were drawn from.	145
7.2	An illustration of adjacency flow values for a fusion of two input sentences using the ILP from §3.3.2. Dashed lines denote all non-zero flow variables. .	146
8.1	Two examples of human-authored phrase alignments between the sentences “ <i>They discussed the aspects in detail and reached an extensive agreement.</i> ” and “ <i>Both parties discussed the specific issues and arrived at a general consensus.</i> ” drawn from Cohn et al. (2008).	169
8.2	Alignment grid for a training example from the Edinburgh corpus with annotated alignments—both SURE and POSSIBLE—in black as well as Meteor alignments in red.	170
8.3	Highlighted tokens indicate recall errors for Meteor which are supported by easily-aligned Stanford dependencies in the alignment example from Figure 8.2.	171
8.4	A potential phrase-based alignment solution \mathbf{v} to the example from Figure 8.1. Faded tokens do not participate in the alignment structure.	174
8.5	A potential dependency alignment solution \mathbf{w} to the example from Figure 8.1 which is consistent with the phrase-based alignment solution from Figure 8.4. Faded tokens and edges do not participate in the dependency alignment. . .	175

List of Tables

2.1	Illustrative examples of some text-to-text operations.	12
2.2	An example of human-generated extractive sentence compression from the compressed news corpus of Clarke and Lapata (2007).	13
2.3	An example of generic sentence fusion from the corpus described in §7.1.	16
3.1	Examples of extractive sentence compression from the broadcast news (BN) corpus from Clarke and Lapata (2006b) and the written news (WN) corpus from Clarke and Lapata (2007).	25
3.2	Percentage of content words and all words that are preserved in compressed sentences by human annotators in the training portion of the BN corpus (Clarke and Lapata, 2006b).	29
3.3	Fraction of dependencies in Stanford parses of input sentences which are also present in parses of reference compressions alongside the fraction of <i>reachable</i> instances—reference parses contained entirely within input parses—from the BN and WN training datasets.	30
3.4	An example of mapping ambiguity between the tokens in an input sentence and its reference compression from the WN corpus of Clarke and Lapata (2006b). Faded tokens in the input sentence indicate those dropped from the reference compression. The circled comma in the reference compression may be resolved to any of three commas in the input sentence.	54

3.5	Experimental results for the BN corpus averaged over 3 reference compressions per instance. All systems were restricted to compress to the size of the median reference compression yielding an average compression rate of 77.26%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test and the paired t-test ($p < 0.05$).	58
3.6	Experimental results for the WN corpus with all systems compressing to the size of the reference compression, yielding an average compression rate of 70.24%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$) and † indicates entries with a different outcome under the paired t-test ($p < 0.05$).	59
3.7	Precision and recall of content-bearing words with respect to reference compressions for the BN corpus. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	61
3.8	Precision and recall of content-bearing words with respect to reference compressions for the WN corpus. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	61
3.9	Examples of system compressions for instances from the BN test dataset.	63
3.10	Examples of system compressions for instances from the WN test dataset.	64
3.11	Experimental results for the BN corpus with all systems restricted to compress to the size of the <i>shortest</i> reference compression, yielding an average compression rate of 66.81%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	65

3.12	Experimental results for the BN corpus with all systems restricted to compress to the size of the <i>longest</i> reference compression, yielding an average compression rate of 86.01%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	66
3.13	Performance variation with n-gram size for the BN corpus with all systems restricted to compress to the size of the median reference compression. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	69
3.14	Performance variation with n-gram size for the WN corpus with all systems restricted to compress to the size of the reference compression. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	70
3.15	Evaluation against subtree deletion over the BN corpus with all systems restricted to compress to the size of the median reference compression. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	71
3.16	Evaluation against subtree deletion over the WN corpus with all systems restricted to compress to the size of the reference compression. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	72
4.1	Empirical tightness of LP- dep and proportion of integral variables among all non-zero variables in each instance averaged over the BN and WN development corpora.	84
4.2	Optimality of output compressions from LP- dep →MST and proportion of correct non-zero variables when compared to ILP- dep for each instance, averaged over the BN and WN development corpora.	84

4.3	Experimental results for the BN corpus averaged over 3 reference compressions per instance. All systems were restricted to compress to the size of the median reference compression yielding an average compression rate of 77.26%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$). . .	88
4.4	Experimental results for the WN corpus with all systems compressing to the size of the reference compression, yielding an average compression rate of 70.24%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$). . .	89
4.5	Time in seconds for inference over the BN and WN test datasets, excluding the time required for initializing and scoring features.	90
4.6	Fraction of instances which converge under dual decomposition and the average number of iterations to convergence for them in the BN and WN test datasets.	91
5.1	Experimental results for the BN corpus averaged over 3 reference compressions per instance. All systems were restricted to compress to the size of the median reference compression yielding an average compression rate of 77.26%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$). . .	113
5.2	Experimental results for the WN corpus with all systems compressing to the size of the reference compression, yielding an average compression rate of 70.24%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$). . .	114
5.3	Fraction of system-generated dependency tree solutions \mathbf{z} which are projective over the BN and WN test datasets.	114
5.4	Time in seconds for inference over the BN and WN test datasets, excluding the time required for initializing and scoring features.	115
5.5	Fraction of instances which converge under bisection and the average number of iterations to convergence for them in the BN and WN test datasets. . . .	116

5.6	Experimental results for the BN corpus averaged over 3 reference compressions per instance. All systems were restricted to compress to the size of the median reference compression yielding an average compression rate of 77.26%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$). . .	117
5.7	Experimental results for the WN corpus with all systems compressing to the size of the reference compression, yielding an average compression rate of 70.24%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$). . .	118
6.1	Effective variant of constraint (6.6) imposed on the ILP for different binary states of a token/concept variable x_j^+ and its incoming semantic relation u_{ij} . The constraint on output flow is only imposed by active incoming semantic relations.	127
6.2	Percentage of SEMAFOR frames and FEs from reference compressions which are present in input sentences alongside the percentage of <i>reachable</i> cases—references with frames or FEs entirely drawn from the input—over the BN and WN training corpora.	130
6.3	Experimental results for the BN corpus with all systems restricted to compress to the size of the median reference compression. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	134
6.4	Experimental results for the WN corpus with all systems restricted to compress to the size of the reference compression. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	135
6.5	Precision and recall of frames and frame elements (FEs) with respect to the reference compressions for the BN corpus. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	137

6.6	Precision and recall of frames and frame elements (FEs) with respect to the reference compression for the WN corpus. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	137
7.1	SCU annotations drawn from DUC 2005–2007 and TAC 2008–2011. Human-annotated contributors to the SCU are indicated as boldfaced spans within the respective source sentences.	142
7.2	An illustration of lexical redundancy in fusion instances from our corpus. Dashed underlining indicates content words—nouns, verbs, adjectives and adverbs—whose stems occur in two input sentences while solid underlining indicates those which occur in all three.	147
7.3	Experimental results over the pyramid fusion corpus with all systems restricted to produce outputs of the same length as reference fusions. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	155
7.4	Results over the pyramid fusion corpus broken down by number of input sentences. Boldfaced entries are statistically undistinguished from the best result within each column <i>and</i> row group under Wilcoxon’s signed rank test ($p < 0.05$).	157
7.5	Examples of reference and system-generated fusions for instances with 2 input sentences from the test partition of the pyramid fusion corpus.	159
7.5	Examples of reference and system-generated fusions for instances with 3 input sentences from the test partition of the pyramid fusion corpus.	160
7.5	Examples of reference and system-generated fusions for instances with 4 input sentences from the test partition of the pyramid fusion corpus.	161
7.6	Experimental results over the pyramid fusion corpus with all systems restricted to produce outputs of the same length as reference fusions. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	162

7.7	Experimental results over the pyramid fusion corpus with all systems restricted to produce outputs of the same length as reference fusions. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	163
7.8	Experimental results for joint models over the pyramid fusion corpus with all systems restricted to produce outputs of the same length as reference fusions. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	165
8.1	Macro-averaged results for prediction of SURE alignments on the test dataset with respect to aligned token pairs, projected dependency alignments over Stanford parses and entire phrase-based alignment configurations. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	182
8.2	Macro-averaged results for prediction of SURE+POSSIBLE alignments on the test dataset with respect to aligned token pairs, projected dependency alignments over Stanford parses and entire phrase-based alignment configurations. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).	183

Acknowledgments

At the culmination of a long and circuitous journey, I cannot help but reflect with gratitude on the people whose guidance and inspiration contributed to this effort.

Foremost, I want to express my deep appreciation to my advisor Kathy McKeown who has been a tremendously understanding and supportive mentor. Kathy gave me the latitude to explore, err and learn for myself, and I gained much from her frank but friendly advice and passion for bold ideas. Many thanks also to the other members of my committee—Mike Collins, Hal Daumé, Julia Hirschberg and Owen Rambow—who offered valuable feedback on this work and have been steady sources of inspiration to me over the years.

I would like to thank my collaborators beginning with Michael White and Alexander Rush, whose ideas and efforts contributed significantly to parts of this dissertation. I'm grateful to Tony Jebara, Yingbo Song, Fadi Biadisy, Dan Bikel, Scott Martin, Yves Petinot, Sara Rosenthal, Jacob Andreas and Mukund Jha for our fruitful research collaborations. A special thanks is owed to Tony, who nurtured my interest in machine learning and gave me many good ideas to chew on in the early years. In addition, I remain indebted to the writings of André F. T. Martins, Ryan McDonald and Mike Collins, whose work has yielded much inspiration for my own.

I spent many a pleasant hour brainstorming with my colleagues at Columbia. Many thanks to my patient officemates—Kevin Lerman, Hila Becker, Yves Petinot, Jenny Finkel, Shay Cohen, Apoorv Agarwal and Or Biran—for all the rambling discussions. At the risk of forgetting someone, I also greatly enjoyed my interactions with Fadi Biadisy, David Elson, Kristen Parton, Daniel Bauer, Bob Coyne, Laura Furst, Ioannis Papparizos, Joshua Gordon, Kevin Mc Inerney, Jessica Ouyang, Karl Stratos, Neeraj Kumar, Ang Cui, Yingbo Song

and Vishakh. I am also very appreciative of all the participants in the first years of the GALE project and would like to single out Sasha Blair-Goldensohn, who was a patient and encouraging mentor and inspired me to consider the doctoral program. And I would be remiss if I didn't acknowledge Mayank Lahiri, who first planted the idea of graduate school in my head many years ago and more recently suggested a notion of life beyond it.

Finally, I thank my family and friends for their support throughout my time at Columbia. My mother Roma has always indulged my academic leanings and provided for an excellent education. My brother Vivek has encouraged me to follow my ideas and supported my choices even when he didn't agree with them. My friendships have been a treasured refuge from the lonely turmoil that comes with academic specialization. And my love and appreciation goes out to Stephanie, to whom this dissertation is dedicated, for seeing this journey through with me through times both lively and lean.

For Steph
and her infinite patience

Chapter 1

Introduction

Although natural language utterances are typically represented as an ordered sequence of words, it is well-established that non-local structural relationships between words and phrases are crucial to recovering the meaning encoded in a particular piece of text. Consider, for example, the sentence: *Alice called Bob in Caracas when she learned that her brother was in trouble*. Various structural annotations that can be drawn over this sentence using standard computational techniques include (a) tag sequences such as the part of speech—noun, past-tense verb, comparative adjective, etc—of each word, (b) dependency trees that capture syntactic relationships between words, e.g., disambiguating whether *Alice called ... in Caracas*, or *Bob [is] in Caracas*, (c) predicate-argument structures such as frame-semantic graphs that might represent *call* as a form of CONTACTING with *Alice* playing the role of COMMUNICATOR, *Bob* as an ADDRESSEE and *her brother was in trouble* as a TOPIC, (d) sets of co-referent entities which might indicate that *Bob* and *her brother* refer to the same entity, (e) relations over entities which can acknowledge a symmetric SIBLING relation between *Alice* and *Bob*, and (f) discourse relations over clauses that would disambiguate whether the *she learned ...* clause has a CAUSAL or TEMPORAL relationship with the *Alice called ...* clause, among others.

Much research in natural language processing focuses on computational techniques to robustly (and efficiently) produce such annotations over text, in part because these structured formalisms—whether motivated by linguistic theory or computational tractability—often yield useful features for downstream text processing tasks. Furthermore, these structures

also serve as practically useful abstractions of information within text for systems that address high-level problems such as text generation (Belz et al., 2010) and machine translation (Koehn et al., 2003; Chiang, 2007; Chiang et al., 2013). However, because different structured representations admit different factorizations over words and therefore different features for statistical learning algorithms, empirical research is required to determine which of these representations is most appropriate for any given problem or domain—a decision that must be revisited as datasets are expanded and tools are refined.

This dissertation aims to explore the notion that the structured abstractions produced by standard natural language analysis tools offer distinct but incomplete perspectives on the meaning encoded within a piece of text, and that automated systems must consider multiple such perspectives in order to capture the complex, multilayered semantics of human language. We formulate a variety of *multi-structured inference*¹ approaches that simultaneously account for different representations of sentences—for instance, as an ordered sequence of words as well as a tree of syntactic dependencies—in high-level problems such as text-to-text generation and text alignment. Our research finds that pairing data-driven supervised learning with the expressive power of multi-structured representations results in rich models that inevitably surpass single-representation techniques in experimental evaluations, even when the latter are aided by hand-crafted linguistic rules. The techniques investigated make minimal assumptions about language and domain and can consequently be extended to additional structured representations or applied to new tasks without significant modification. Moreover, we consider the tradeoff between optimal solutions, runtime efficiency and model flexibility throughout this investigation and develop or describe more efficient variants of multi-structured inference that rely on approximate solutions or mildly constrained interpretations of the problems under study.

¹The *inference*, *decoding* or *argmax* problem refers to the task of generating the best output labeling for a given input under a particular model parameterization. In this work, the input to inference is always text and the output is one or more structured representations of text or an alignment between text segments. See §3.2.1 for details.

1.1 Multi-Structured Inference

Structured prediction problems are ubiquitous in natural language analysis. For instance, the task of identifying the parts of speech—nouns, verbs, adjectives etc—of the words in a sentence is usually treated as a sequence-labeling problem in which the per-word labels are assumed to be interdependent, e.g., the phrase *time flies* could be jointly tagged as either NN VBZ (as in “*Time flies like an arrow*”) or VBP NNS in “*You must time flies in the testing chamber*”) but never as VBP VBZ. Structured output spaces are also integral to syntactic representations which typically take the form of trees representing constituent hierarchies or grammatical dependency relations between words, i.e., a potential dependency relationship between two words cannot be established independently as it must not violate the tree of dependencies between other words. In general, the defining feature of structured prediction problems is the presence of complex output labelings which cannot be conveniently factored into independent classification problems.²

Part-of-speech tag sequences and parse trees are just two forms of linguistic abstraction over a raw stream of words from a vast vocabulary; other possible abstractions include predicate-argument structures, expressions in first-order logic, entity-relation graphs and clusters of coreferent entities and events. Automated systems to generate these annotations over unseen text offer both a window into text understanding and, more concretely, useful features for systems addressing higher-level natural language problems. For example, the sentiment analysis problem is often viewed as a sentence-level classification task in which lexical features take priority, but prediction performance has been shown to improve significantly with models which take syntactic structure into account (Socher et al., 2013).

In this work, we seek to examine the *representational* power of such heterogenous text abstractions by utilizing them for higher-level structured prediction tasks. These structured formalisms, largely motivated by linguistic theories of syntax and semantics, are usually manifested as category labels defined over one or more words in a sentence or as binary relations between pairs of words. Consequently, we direct our attention to jointly modeling

²Despite the importance of structured prediction problems to natural language processing, a formal definition is elusive in the literature. Daumé III (2006) §2.2.1 discusses the conditions under which a problem might fall under the purview of structured prediction.

the various graph structures—paths, trees, directed acyclic graphs, etc—that these abstractions are built on, which in turn inform the types of scoring functions that can be defined over natural language text.

For instance, an obvious structural phenomenon in natural language text is the ordering of words in a sentence. In English, altering the order of words can dramatically affect the meaning and perceived fluency of a given piece of text, even when the rearranged words share the same syntactic role, e.g., the widespread preference for *big red ball* as opposed to *red big ball*. This phenomenon motivates the widespread use of n-gram factorizations—*n*th-order Markov factorizations of a sequence of tokens—which are often used with probabilistic language models (LMs) for scoring and decoding sentences. However, these factorizations depend on unrealistic independence assumptions and cannot adequately account for the non-local syntactic dependencies prevalent in real-world text.

This limitation can be partially overcome by considering text representations built on syntactic structure such as a parse tree consisting of dependency relations between words. An edge factorization of a dependency tree maintains modifier relationships—including long-range dependencies—regardless of their ordering in the input sentence. However, dependency structures by themselves cannot always be deterministically linearized to fluent text and are often paired with LMs for this task (Bangalore and Rambow, 2000). In addition, syntax-based representations require the use of statistical parsing which can be noisy, particularly when working with longer sentences and informal speech.

We hypothesize that both these structural perspectives—n-gram sequences and syntactic trees—as well as other structured representations of text offer complementary views on the meaning embedded within a sentence. Joint inference under these diverse structures might therefore yield more robust and effective solutions to structured prediction problems over natural language—analogueous to the well-known advantages of multi-view learning (Xu et al., 2013). In order to explore these ideas, we develop inference techniques which explicitly account for multiple structural perspectives in the context of natural language tasks which require the assembly of novel sentences and the alignment of related sentences. The following section describes these tasks in detail.

1.2 Tasks

We investigate the application of multi-structured inference in the context of discriminative structured prediction tasks wherein systems take natural language text as input and produce a well-defined structured labeling under some learned model parameters. Specifically, we consider inference for *text-to-text generation*, in which the structured labeling to be generated represents a complete sentence produced by some transformation of input text, as well as *text alignment*, in which the labeling expresses the connection between two related pieces of input text. Both types of problems operate over natural language input and produce outputs that can be directly interpreted by laypersons, thereby providing a useful framework to compare structures used in text representation. In addition, these systems have practical utility in a variety of applications such as abstractive summarization and textual entailment recognition.

Our primary application for these ideas is in text-to-text generation systems that automatically manipulate short pieces of text for various purposes—for instance, the *compression* task requires sentences to be shortened significantly without losing their essential meaning while *fusion* requires the merging of multiple related sentences in order to emphasize common information. The aim of inference for these tasks is the construction of fluent and eloquent sentences from the words in the input text, thereby implicitly raising the question of how sentences should be represented and scored.

Beginning with the well-studied single-sentence compression task, we describe a supervised text-to-text generation system which uses integer linear programming to simultaneously infer an optimal compressed sentence consisting of both a good ordering of input words *and* a good syntactic parse tree over the words. This technique generalizes over many prior approaches to the compression task and experimental evaluations indicate that it achieves state-of-the-art performance without any need for carefully chosen linguistic constraints.

We also confront practical issues in trading off the efficiency and flexibility of inference within the context of sentence compression. In addition to the exact inference approach which uses integer linear programming, we investigate approximate inference algorithms which recover good-enough solutions along with a practical improvement in runtime. When further restrictions are imposed on the ordering and dependency structure of output com-

pressed sentences, the multi-structured inference formulation also admits a dynamic programming approach which offers both asymptotic efficiency and the ability to use higher-order dependency features with no runtime overhead.

In addition, we exploit the flexibility of the basic integer linear programming formulation for compression and extend it to additional structured representations and text-to-text applications. For the former, initial work is presented on incorporating directed acyclic graphs representing predicate-argument structures, allowing us to directly compare the utility of syntactic and semantic representations under a task-based compression evaluation. We further show how this single-sentence compression approach can be straightforwardly extended to multiple input sentences—the sentence *fusion* task—and identify a novel corpus of naturally-occurring sentence fusions which enables experimentation on this problem. Furthermore, we note that this inference approach can also be generalized to other text-to-text problems such as paraphrasing, simplification and error correction, although a deeper consideration of these tasks is left to future work.

Finally, in addition to text-to-text generation, we also examine the utility of multi-structured inference in text *comparison*—specifically, the problem of monolingual text alignment, a supervised task which involves pairing up fragments of text that convey identical meaning in related sentences. Monolingual alignment is often used as a precursor to sentence fusion and has direct application in higher-level pairwise problems like paraphrase recognition and textual entailment recognition. As with the generation problems described previously, we find that joint alignment over phrases in the input sentences *and* over the dependencies in their parse trees is more robust than using either in isolation.

1.3 Broader Impact

The modern information age is marked by an unprecedented availability of vast amounts of data on the World Wide Web that ranges widely in temporal scope (from historical to real-time), subjectivity (from technical to anecdotal), veracity and style. Much of this information exists in the form of unstructured text intended for human eyes, while the sheer volume of this content necessitates the use of automated techniques such as indexing and

keyword search to help users identify relevant content. However, as mobile computing devices have become pervasive in recent years, access to the web is increasingly accomplished through interaction with small screens—often while users are multitasking—thereby emphasizing the need for automated systems that can understand and respond to users in natural language.

Text-to-text generation techniques have the potential to make information access more efficient and responsive, particularly in mobile computing scenarios. Tasks such as sentence compression and sentence fusion are important components of abstractive text summarization with which text can be rewritten at the sentence level to suit the particular information needs of a user. Similarly, paraphrasing and sentence simplification hold promise for automated approaches that adapt output text according to the speaking preferences and language proficiency of the user. A wide variety of applications fall under the paradigm of text-to-text generation, ranging from editorial tasks like grammatical error correction and headline generation to creative projects like poetry generation.

Looking forward, we envision these ideas coalescing into a single framework for text-to-text generation that is capable of addressing many disparate tasks which involve the production of fluent sentences. By pairing supervised learning with expressive multi-structured inference algorithms, a wide variety of features, representations, learning techniques and specific annotation tools can be compared in task-based evaluations. The use of a common architecture also invites the exploration of multi-task learning approaches that utilize annotated data from different tasks in order to learn better models of human language.

Beyond the immediate aim of progress on real-world applications of text-to-text generation and monolingual alignment, our long-term goal is to advance the symbiosis between text production and analysis. We anticipate that errors made by automated text transformation systems will provide insight into the relative benefits and weaknesses of different types of analysis and perhaps even indicate areas in which new tasks or annotation efforts can be focused. Furthermore, we offer a hypothesis that even potentially-noisy generated text and its jointly recovered annotations may comprise useful training data for standard natural language structured prediction tasks, perhaps leading to novel semi-supervised learning scenarios involving text-to-text tasks.

1.4 Contributions

The broad contributions of our work include:

- **Rich text representations:** The combination of standard structured representations such as n-gram factorizations and dependency trees in our inference approaches enables models to consider a rich set of features over phrases, syntactic relations and other similar structures.
- **A unified text-to-text framework:** We present powerful inference techniques that generalize over existing systems for supervised compression and fusion and can also extend to other text-to-text problems with little modification.
- **A multi-structured text aligner:** In addition to text-to-text systems, we also developed a supervised monolingual text alignment system which jointly produces phrase-based and dependency alignments.
- **Approximate inference strategies:** In addition to exact inference approaches, we describe an approximate multi-structured inference strategy which offers practical speedups for applications in both text-to-text generation and alignment.
- **New corpora:** We produce a new, natural dataset for sentence fusion in order to address annotation issues that have previously limited research in this task. In addition, we have developed and released an improved version of the alignment dataset of Cohn et al. (2008) with corrected tokenization and annotations.

1.5 Overview

Our primary goal for this research is to explore rich and flexible inference approaches in problems that bridge the gap between natural language analysis and generation. Chapter 2 provides background information and related work for the applications under study.

An exact multi-structured inference technique is introduced in Chapter 3 for the sentence compression task. We use integer linear programs with flow-based constraints to assemble an output sentence from the words of the input sentence under factorizations of n-gram and dependency tree representations. When evaluating on supervised sentence compression

tasks, we find that using a heterogenous output space to represent output text under both structured representations improves significantly over using either in isolation.

Integer linear programming can be computationally expensive on large problems, raising the question of whether we can sacrifice optimal solutions for a practical improvement in runtime. In Chapter 4, we consider approximation strategies for the same task, using dual decomposition to pair an exact inference approach under n-gram factorization with a linear programming relaxation for syntactic structure. In experiments, this approach produces runtime gains over the exact approach with a small loss in output quality.

Do efficient algorithms exist for multi-structured inference? Chapter 5 describes a dynamic programming approach to recover multi-structured compressions when output structures are restricted to order-preserving bigram factorizations and projective dependency trees. This yields polynomial-time inference for multi-structured compression and consequently significant runtime improvements over the less constrained inference formulations. Furthermore, this approach also permits richer parameterizations over second-order dependencies with no asymptotic overhead in runtime.

Can these techniques also address richer representations of textual meaning? In Chapter 6 we revisit the flexible integer linear programming approach and propose a flow-based constraint framework to recover optimal directed acyclic graphs, which can represent various kinds of predicate-argument structure for inference in text-to-text problems. In addition, we specifically examine the empirical utility of frame-semantic structures in the sentence compression task.

Does this approach generalize to other text-to-text generation tasks? In Chapter 7, we apply our inference framework to the sentence fusion task through additional features and constraints that capture the support of words and concepts across input sentences. In order to train models, we introduce a new corpus of natural sentence fusions drawn from datasets intended for summarization evaluation. Our experiments indicate that multi-structured inference proves advantageous in this setting as well regardless of the mechanism used for content selection.

Are these ideas applicable to other natural language problems? Chapter 8 turns to the problem of monolingual alignment between related fragments of text such as para-

phrases and introduces an inference approach to jointly produce consistent alignments over phrases and dependencies. We observe that the use of multi-structured output space leads to improvements over standard phrase-based techniques on a dataset of manually-aligned paraphrases from various sources.

Finally, we conclude with a discussion of the limitations of this research and directions for future work in Chapter 9.

Chapter 2

Background on Tasks

Monolingual text-to-text generation tasks can broadly be described as tasks that take naturally occurring human language text as input and generate fluent output text, performing some transformations that humans can (and do) perform naturally while writing and editing. This is broadly defined to include problems in the literature which involve:

1. Natural language text as input and output, specifically sentences or documents without additional structure.
2. Transformations that rely only on the information provided in the input and generic linguistic knowledge.

The second point above is needed to distinguish tasks that are normally considered as text-to-text generation from information retrieval (IR) or question answering (QA) tasks. These latter problems involve the retrieval of documents or sentences from a document collection that are relevant to natural language queries or questions respectively. Although the input and output in these problems does consist of natural language text, the document collection must also be implicitly regarded as part of the input along with the query or question; this structural differentiation between input components places these tasks outside our operational definition of text-to-text problems. Under this definition, we would consider *query-based* generation problems such as query-based fusion (Krahmer et al., 2008) to be hybrid text-to-text/IR tasks.

Sentence A	Production was closed down at Ford last night for the Christmas period.
Sentence B	Negotiations with union leaders remained stalled as assembly lines wound down and are scheduled to resume in January.

Compression of A	Production closed at Ford for Christmas.
Paraphrase of A	Ford production was halted yesterday for the holidays.
Fusion of A and B	Negotiations remained stalled as Ford production closed for Christmas.
Simplification of B	{ Negotiations with union leaders remained stalled. They are scheduled to resume in January.

Table 2.1: Illustrative examples of some text-to-text operations.

Prior work in text-to-text generation has largely centered around a few specific problems driven by application needs and data availability. Table 2.1 lists a few frequently studied text-to-text operations. In this dissertation, we focus on inference techniques for sentence compression and sentence fusion along with the related problem of text alignment which, although not a generation task, is frequently used in text-to-text problems for corpus construction, feature generation and evaluation. We describe these tasks and summarize prior research in relevant areas below.

2.1 Sentence Compression

Sentence compression is a popular text-to-text generation problem in which an input sentence must be transformed into a shorter output sentence which accurately reflects the essential meaning of the input and also remains grammatically well-formed. The name derives from an analogy to the general problem of lossy data compression in which the size of the input—quantified here in the number of words or characters in a sentence—must be reduced while preserving its essential characteristics—in this case, the most salient information in the sentence.

The definition of sentence compression is somewhat analogous to document summariza-

Input	For the first time , archaeologists have been able to study in detail the techniques used by post-medieval builders to construct the typical “ cob ” houses for which the West Country is famous .
Compression	For the first time , archaeologists have been able to study the techniques used to construct “ cob ” houses .

Table 2.2: An example of human-generated extractive sentence compression from the compressed news corpus of Clarke and Lapata (2007).

tion defined at the sentence level. Just as summarization is often reduced to a sentence extraction problem in order to sidestep the challenges of fluent text generation, sentence compression is typically formulated as a word deletion task in which an output sentence is constructed by dropping tokens from the input sentence without any paraphrasing or re-ordering. This deletion-based approach is also referred to as *extractive compression* by Cohn and Lapata (2008) & Galanis and Androutsopoulos (2010) following the distinction between extractive and abstractive approaches to document summarization. Table 2.2 contains an example of extractive compression from a corpus of human-generated compressions.

Compression problems have received significant attention in recent years due to their usefulness in document summarization as well as the increasing number of sources of compression data for training and evaluation (Knight and Marcu, 2000; Clarke and Lapata, 2006b; Cohn and Lapata, 2008; Nomoto, 2009; Galanis and Androutsopoulos, 2011; Filipova and Altun, 2013). With a straightforward task definition and minimal domain assumptions, compression is therefore seen as a fundamental problem for data-driven approaches involving language generation.

2.1.1 Related work

Text-to-text generation tasks first emerged as techniques to achieve abstractive summarization. An early notion of text-to-text generation was proposed by Dras (1997) in the form of reluctant sentence paraphrasing under constraints of word length. Jing and McKeown (2000) analyzed human-generated summaries and reported that human summarization

relied heavily on sentence *reduction* (Jing, 2000) and sentence *merging*.

Knight and Marcu (2000) created a dataset of extractive compression problems by aligning news documents to document abstracts from the Ziff-Davis summarization corpus, consequently generating interest in supervised approaches to compression problems (Knight and Marcu, 2002; Riezler et al., 2003; Turner and Charniak, 2005; McDonald, 2006; Unno et al., 2006; Galley and McKeown, 2007; Nomoto, 2007; Cohn and Lapata, 2009). A larger dataset of extractive sentence-level compressions within entire documents was produced by Clarke and Lapata (2006b) for broadcast news transcriptions and later extended to written news stories in Clarke and Lapata (2007); these have been used for most recent work on the sentence compression task (Clarke and Lapata, 2006a; Cohn and Lapata, 2007; Filippova and Strube, 2008a; Nomoto, 2009; Cohn and Lapata, 2009; Galanis and Androutsopoulos, 2010; Napoles et al., 2011a; Qian and Liu, 2014) as well as the experiments presented in this dissertation. Other lines of research have also attempted to broaden the notion of compression beyond mere word deletion to incorporate paraphrases (Cohn and Lapata, 2008; Cohn and Lapata, 2009; Nomoto, 2009; Galanis and Androutsopoulos, 2010; Marsi et al., 2010; Ganitkevitch et al., 2011; Napoles et al., 2011a). Although we focus on the more common extractive setting in this work, many of the inference techniques presented in the following chapters can accommodate paraphrases and therefore remain viable for these abstractive compression problems.

A wide variety of techniques have been explored for extractive sentence compression. Of particular relevance to our work is McDonald (2006) which builds on Knight & Marcu’s (2002) transition-based model and proposes a dynamic programming approach to recover a bigram factorization of a compressed sentence. Clarke and Lapata (2006a) further extends this idea by using integer linear programming to replace pairwise bigrams with trigrams and enable unsupervised compression with a language model. Other recent work (Filippova and Strube, 2008a; Galanis and Androutsopoulos, 2010) avoids n-gram factorizations and instead uses dependency trees as representations of compressed sentences. We propose inference strategies to generalize over both these n-gram and dependency-based techniques in the multi-structured inference approaches presented in the following chapters.

Progress on standalone compression tasks has also enabled document summarization

techniques that address both sentence selection and compression (Daumé and Marcu, 2002; Clarke and Lapata, 2007; Madnani et al., 2007; Zajic et al., 2007; Gillick and Favre, 2009; Liu and Liu, 2009; Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Chali and Hasan, 2012; Genest and Lapalme, 2012; Woodsend and Lapata, 2012; Almeida and Martins, 2013; Li et al., 2013; Molina et al., 2013; Morita et al., 2013; Qian and Liu, 2013; Wang et al., 2013; Kikuchi et al., 2014; Li et al., 2014), with recent work formulating the summarization task as joint sentence extraction and compression and often employing ILPs or Lagrangian relaxation for constrained inference. Although we restrict the scope of this dissertation to sentence-level compression techniques, our proposed approaches are compatible with a number of these systems and we intend to extend them to the summarization setting in future work.

2.2 Sentence Fusion

Sentence *fusion* is the general term applied to tasks which take multiple sentences as input to produce a single output sentence. Just as sentence compression can be thought of as a sentence-level analog to single document summarization, fusion serves as a sentence-level variant of multi-document summarization. However, the term *fusion* has also been applied to different tasks involving sentence combination over the years; for instance, in addition to the combination of related sentences from different documents, it has also been used to refer to the combination of contiguous or narratively cohesive sentences from a single document (Daumé III and Marcu, 2004; Elsner and Santhanam, 2011).

Generic fusion of sentences has been noted to be difficult for humans to annotate consistently, prompting doubt about whether the task is well-defined for automated systems (Daumé III and Marcu, 2004). However, the challenge of identifying salient content in sentences with more than one predicate is also common to other tasks such as sentence compression, as noted by McDonald (2006). This problem can be alleviated by introducing external sources of salience such as in query-based sentence fusion (Krahmer et al., 2008) or by using a stricter definition of the task, e.g., McKeown et al. (2010) find that untrained annotators can consistently construct the *union* of all information in the input sentence.

Input 1	The heavy-metal group Metallica filed a federal lawsuit in 2000 against Napster for copyright infringement, charging that Napster encouraged users to trade copyrighted material without the band’s permission.
Input 2	The heavy metal rock band Metallica, rap artist Dr. Dre and the RIAA have sued Napster, developer of Internet sharing software, alleging the software enables the acquisition of copyrighted music without permission.
Input 3	The heavy-metal band Metallica sued Napster and three universities for copyright infringement and racketeering, seeking \$10 million in damages.
Fusion	Metallica sued Napster for copyright infringement

Table 2.3: An example of generic sentence fusion from the corpus described in §7.1.

Owing in part to the challenges in annotation, research in standalone sentence fusion has been hampered by the lack of common datasets for the task. The crowdsourced fusion datasets produced by McKeown et al. (2010) suffer from annotator errors while the Reuters news dataset used for training the models of Elsner and Santhanam (2011) cannot be distributed; moreover, both are limited to a few hundred instances and thus insufficient for training models with rich sparse features. In §7.1, we outline the construction of a new dataset of *natural* sentence fusions from summarization evaluation data, an example from which is provided in Table 2.3.

2.2.1 Related work

Sentence fusion was introduced by Barzilay et al. (1999) and Barzilay and McKeown (2005) in the context of multi-document summarization as a way to better capture the information in a cluster of related sentences than just using the sentence closest to the centroid. This approach to merging sentences follows human strategies for summarization as per the analysis of human-generated summaries presented in Jing and McKeown (2000). The definition of fusion has since expanded to encompass other forms of sentence combination such as the combination of two sentences—usually contiguous—from a single document (Daumé III and Marcu, 2004; Elsner and Santhanam, 2011). In addition, although work on standalone

sentence fusion problems has focused on a pairwise setting in which only two sentences must be merged, research on combining sentence clusters has also been revisited under the term *multi-sentence compression* (Filippova, 2010; Boudin and Morin, 2013; Tzouridis et al., 2014). In addition, Cheung and Penn (2014) proposes a formulation for *sentence enhancement* problems in which information from dissimilar sentences is used to augment a sentence using techniques similar to sentence fusion.

Other variations on the fusion task include the set-theoretic notions of *intersection* and *union* (Marsi and Krahmer, 2005; McKeown et al., 2010), which forego the problem of identifying relevance and are thus less dependent on context. Query-based versions of these tasks have been studied by Krahmer et al. (2008) and have produced better human agreement in annotation experiments than generic sentence fusion (Daumé III and Marcu, 2004). McKeown et al. (2010) produced a crowdsourced corpus of intersections and unions which we employed in preliminary experiments on inference for sentence intersection (Thadani and McKeown, 2011b); however, the size and accuracy of this corpus was not sufficient for supervised models studied here.

A popular strategy for fusion relies on merging the dependency trees of input sentences to produce a tree-structured representation of the output sentence that must then be linearized in a separate stage (Barzilay and McKeown, 2005; Filippova and Strube, 2008b; Elsner and Santhanam, 2011; Cheung and Penn, 2014). In contrast, multi-sentence compression techniques for sentence clusters generate output sentences as high-scoring paths in a weighted bigram graph (Filippova, 2010; Boudin and Morin, 2013; Tzouridis et al., 2014). As with sentence compression, we generalize over both n-gram and dependency-factored output spaces in our inference strategy for sentence fusion.

2.3 Text Alignment

Textual alignment problems involve the identification of links between text fragments which are effectively semantically equivalent in their respective sentences. Alignment tasks are differentiated by the form of the text fragments that must be paired up: for instance, a word alignment produces a matching over individual words whereas a phrase alignment

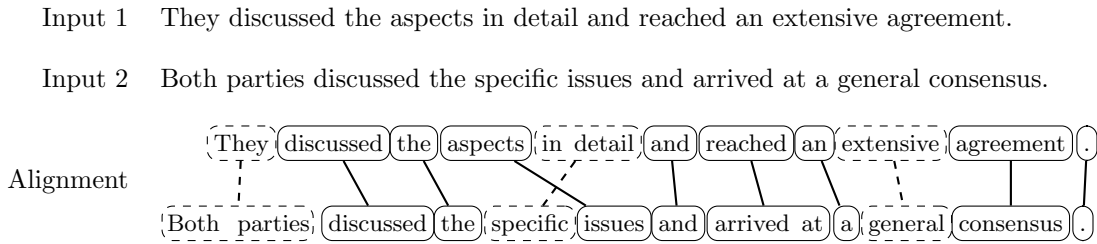


Figure 2.1: An example of phrase-based monolingual alignment drawn from the aligned paraphrase corpus of Cohn et al. (2008). Solid lines indicate SURE alignments while dashed lines indicate POSSIBLE alignments.

produces links between contiguous non-overlapping phrases in the input text as seen in Figure 2.1. Although alignment problems are usually studied in the context of automated machine translation (MT), monolingual alignment is useful in natural language problems which involve pairs or groups of related sentences such as paraphrase and textual entailment recognition.

Unlike compression and fusion, alignment is not a text-to-text generation problem but often finds application in these tasks. For example, many techniques for fusion-like tasks require word alignments to create dependency graphs (Barzilay and McKeown, 2005; Filippova and Strube, 2008b; Filippova, 2010; Boudin and Morin, 2013; Tzouridis et al., 2014; Cheung and Penn, 2014) or generate them during inference (Elsner and Santhanam, 2011). Alignment approaches are also useful in automated corpus construction for text-to-text tasks like sentence simplification (Bott and Saggion, 2011). Despite the utility of monolingual alignment and the ubiquity of tools for the problem, little research has been conducted into the tradeoffs between word or phrase-based alignment representations and syntactic variants such as alignments of dependency edges—aspects that we aim to address by developing a multi-structured inference strategy for text alignment which simultaneously accounts for both these representations.

2.3.1 Related work

Text alignment is a crucial component of machine translation (MT) systems (Vogel et al., 1996; Och and Ney, 2003; Liang et al., 2006b; DeNero and Klein, 2008); however, the general goal of multilingual aligners is the production of wide-coverage phrase tables for translation. In contrast, monolingual alignment is often consumed directly in applications like paraphrasing and textual entailment recognition; this task therefore involves substantially different challenges and tradeoffs.¹ Nevertheless, modern MT evaluation metrics have recently been found to be remarkably effective for tasks requiring monolingual alignments (Bouamor et al., 2011; Madnani et al., 2012; Heilman and Madnani, 2012)—even used off-the-shelf with their default parameter settings.

Monolingual word alignment has been used for many natural language processing applications such as paraphrase generation (Barzilay and Lee, 2003; Quirk et al., 2004) and variants of sentence fusion (Filippova and Strube, 2008b; Filippova, 2010; Boudin and Morin, 2013; Tzouridis et al., 2014; Cheung and Penn, 2014). Token alignment approaches which are constrained by dependency structure have been used in tasks such as sentence fusion (Barzilay and McKeown, 2005; Marsi and Krahmer, 2005) and redundancy removal (Thadani and McKeown, 2008). Joint aligners that simultaneously account for the similarity of tokens and dependency edges have also been explored (Chambers et al., 2007; Chang et al., 2010; Sultan et al., 2014).

Monolingual phrase-based alignment was first tackled by the MANLI system of MacCartney et al. (2008) using simulated annealing for search. We subsequently expanded upon this work in Thadani and McKeown (2011a) to enable exact inference and syntactic constraints through integer programming, leading to the exploration of joint phrase-based and dependency alignment presented in Chapter 8. Recent years have seen further advancements in both word and phrase-based alignments using conditional random fields (Yao et al., 2013a; Yao et al., 2013b) and in joint word and dependency alignment (Sultan et al., 2014).

¹An enumeration of these challenges in the context of textual entailment recognition is provided in MacCartney et al. (2008).

2.4 Other Related Tasks

Although this dissertation considers the application of multi-structured inference for compression, fusion and alignment tasks, our techniques appear pertinent for various additional text-to-text problems previously studied in the literature. A brief overview of the relevant areas follows.

2.4.1 Paraphrase generation

Sentence-level paraphrasing is a task similar to sentence compression in which the constraint of length reduction is replaced with one that mandates that the semantics of the original sentence are preserved. Paraphrase construction generally relies on the application of lexical or lexico-syntactic rules which are devised manually (McKeown, 1983) or harvested from parallel or comparable corpora (Barzilay and McKeown, 2001; Bannard and Callison-Burch, 2005; Callison-Burch, 2008; Ganitkevitch et al., 2013). The application of these resources to generate cohesive sentences has also been extensively studied (Barzilay and Lee, 2003; Pang et al., 2003; Quirk et al., 2004; Zhao et al., 2009; Madnani and Dorr, 2010; Ganitkevitch et al., 2011; Metzler et al., 2011).

Paraphrase generation techniques have application in natural language problems such as question answering (McKeown, 1983) and machine translation (Madnani and Dorr, 2010) and are useful in applications involving the preservation of author anonymity (Narayanan et al., 2012). Although we do not address paraphrase generation directly in this dissertation, all the inference techniques we investigate are capable of supporting simple lexical paraphrases while the approaches based on integer programming can accommodate more complex paraphrasing rules. In future work, we intend to adapt these techniques to standalone paraphrase generation tasks and also incorporate paraphrase resources into compression and fusion problems.

2.4.2 Sentence simplification

While sentence fusion addresses the problem of combining information from different sentences, the goal of sentence simplification is to *decompose* complex sentences into simpler,

single-predicate statements. Such transformations are used to improve content selection tasks (Klebanov et al., 2004; Siddharthan et al., 2004) and also to tailor text according to reading proficiency for the benefit of children and non-native speakers (Carroll et al., 1999).

Much work on simplification relies on manually devised lexico-syntactic rules (Siddharthan, 2006; Siddharthan, 2010; Siddharthan, 2011). However, recent work in this area has leveraged revision data from the Simple English Wikipedia² in order to learn lexical simplifications (Yatskar et al., 2010) or models for simplifying whole sentences using MT systems (Zhu et al., 2010) and synchronous grammars (Woodsend and Lapata, 2011; Angrosh and Siddharthan, 2014; Siddharthan and Angrosh, 2014).

2.4.3 Title generation

The title or headline generation task requires the production of a single sentence-length summary (Banko et al., 2000; Zajic et al., 2002; Dorr et al., 2003; Soricut and Marcu, 2006; Woodsend et al., 2010) to characterize a document or serve as its title. This task is closely related to sentence compression as evidenced by the use of document titles for automatically generating compression targets in the construction of the corpus of Filippova and Altun (2013). Although tasks that take an entire document as input and produce a finite number of sentences as output necessarily put a greater emphasis on content selection over the inference challenge of producing a fluent sentence, the techniques we propose are generic enough to remain applicable in this setting.

Generic text-to-text generation has also been approached in the past with WIDL, a symbolic formalism for phrase selection and reordering (Soricut and Marcu, 2005). This has been successfully applied to MT decoding and headline generation (Soricut and Marcu, 2006) using a log-linear framework in which decoding involves the intersection of a probabilistic finite state automaton with a language model (LM). Although our proposed approaches share the goals of this work in aiming to address generic text-to-text generation tasks, they appear more flexible owing to their capability to incorporate rich structured output including dependency trees and, in some cases, directed acyclic graphs for predicate-argument structure as well as arbitrary global constraints on output text.

²Simple English Wikipedia: <http://simple.wikipedia.org>

2.4.4 Machine translation

Although text-to-text generation addresses strictly monolingual transformations, it shares many challenges with automatic machine translation, one of the oldest and most studied problems in the field of computational linguistics. MT tasks are generally formulated as the problem of transforming a sentence in an input source language to a sentence in a different target language that expresses the same meaning. Research in automated MT systems encounters obstacles in phrase-based decoding and evaluation measures which are common to text-to-text generation tasks such as paraphrase generation. The relative maturity of MT tools has led to their use in monolingual tasks such as paraphrase generation (Quirk et al., 2004; Madnani and Dorr, 2010) and sentence simplification (Zhu et al., 2010; Wubben et al., 2012).

The techniques that we discuss in this work have connections with both traditional statistical MT approaches such as phrase-based MT (Koehn et al., 2003) and syntax-based statistical MT techniques such as tree-to-string translation (Liu et al., 2006; Huang et al., 2006; Huang and Mi, 2010). In particular, a number of approaches which have been proposed to combine phrasal and syntactic models (Huang and Chiang, 2007; Rush and Collins, 2011) *inter alia* offer directions for future research into text-to-text generation. Recent work on efficient graph transducers for semantics-based MT (Chiang et al., 2013) has prompted an annotation effort for graph-structured semantic representations over entire sentences, serving to motivate our initial exploration of semantic structures for multi-structured inference in Chapter 6. Finally, automated evaluation of abstractive generation systems is fraught with the same challenges as in MT and automated MT metrics such as BLEU (Papineni et al., 2002) and NIST (Doddington, 2002) are often used to quantify the performance of text-to-text generation systems.

We now turn to the original research content of this dissertation, beginning with a supervised approach for multi-structured sentence compression.

Chapter 3

Multi-Structured Compression

Even though sentence compression is typically formulated as a token deletion task, it is plainly evident that dropping tokens independently from an input sentence will likely not result in fluent and meaningful compressive text. Tokens in well-formed sentences participate in a number of syntactic and semantic relationships with other tokens, so one might expect that accounting for heterogenous structural relationships between tokens will improve the coherence of the output sentence. For this reason, compression systems often assemble output sentences from larger units of text such as n-grams (McDonald, 2006; Clarke and Lapata, 2008) or dependency relations (Filippova and Strube, 2008a; Galanis and Androutsopoulos, 2010). However, both these approaches depend on different structured output spaces—a sequence of n-grams and a tree of dependency relations respectively—and therefore offer distinct advantages in generating fluent text.

In this chapter, we introduce a novel supervised framework for sentence compression which employs a joint inference strategy to simultaneously generate sentences under *both* structured perspectives—an n-gram sequence as well as a dependency tree.¹ Sentence generation is treated as a discriminative structured prediction task in which rich linguistically-motivated features can be used to predict the informativeness of specific tokens within the input text as well as the fluency of n-grams and dependency relationships in the output text. We then present a novel integer linear program that optimally solves the joint in-

¹This is joint work with Kathleen McKeown. An early version of this research was presented in Thadani and McKeown (2013a).

ference problem by using the notion of *commodity flow* (Magnanti and Wolsey, 1994) to ensure the production of acyclic trees and non-branching n-gram sequences that represent an output sentence. The primary contributions in this chapter include:

- An inference formulation to recover the optimal compressed dependency tree in an arbitrary directed graph.
- An inference formulation to recover the optimal n-gram-factored compression with any n-gram size and arbitrary reordering of input tokens, unlike previous work which is restricted to order-preserving bigrams and trigrams.
- A combination of the dependency-based and n-gram based inference techniques which enables rich features for supervised compression that factor over both n-grams and dependency relations.

These expressive models offer additional flexibility when compared to existing models that compress via n-gram or dependency factorizations (McDonald, 2006; Clarke and Lapata, 2008; Filippova and Strube, 2008a; Galanis and Androutsopoulos, 2010), permitting both arbitrary reordering as well as lexical substitution for paraphrasing. However, we do not evaluate token reordering or paraphrasing here as the corpus described in the following section features human-generated extractive compressions that preserve token ordering.

3.1 Compression Corpora

The first widely used corpus for research in sentence compression was constructed through automated alignments of sentences within documents to corresponding abstracts in the Ziff-Davis collection (Knight and Marcu, 2000; Knight and Marcu, 2002). However, some properties of this corpus have proved vexing for researchers interested in exploring compression—most notably, the small size of the test set (32 compression instances), the lack of document context for salience judgments in human evaluations and an aggressive compression rate (47%) for the abstract sentences that fulfilled the criteria for inclusion in the corpus.

To address these limitations, Clarke and Lapata (2006b) employed human annotators to produce compressions of all sentences within in a document. Annotators were restricted to drop words when compressing text so the resulting sentences do not feature any reordering

(BN)	<p>Input The White House says it was a bureaucratic mistake that occurred in 1993 when an Army staffer was detailed to the White House and he was updating the list of people with clearance to enter the grounds here .</p> <p>Compression 1 The White House says it occurred in 1993 when an Army staffer was detailed to the White House updating the list of people with clearance to enter the grounds .</p> <p>Compression 2 The White House says it was a mistake that occurred in 1993 when an Army staffer was updating the list of people with clearance to enter the grounds .</p> <p>Compression 3 A mistake occurred in 1993 when an Army staffer was updating the list of people with clearance to enter the grounds .</p>
------	--

(WN)	<p>Input For the first time , archaeologists have been able to study in detail the techniques used by post-medieval builders to construct the typical “ cob ” houses for which the West Country is famous .</p> <p>Compression For the first time , archaeologists have been able to study the techniques used to construct “ cob ” houses .</p>
------	--

Table 3.1: Examples of extractive sentence compression from the broadcast news (BN) corpus from Clarke and Lapata (2006b) and the written news (WN) corpus from Clarke and Lapata (2007).

or paraphrasing of the words in the input, i.e., a compressed sentence is a subsequence of its input sentence. Compressions were produced in this manner for the sentences in 50 broadcast news (BN) stories drawn from the HUB-4 1996 English Broadcast News Speech corpus.² In Clarke and Lapata (2007), this annotation effort was extended to 82 written news (WN) stories from the British National Corpus³ and the LA Times and Washington Post articles from the North American News Text Corpus.⁴ We use these datasets⁵ for all compression experiments in this dissertation.

²HUB-4 1996 English Broadcast News Speech corpus: <https://catalog.ldc.upenn.edu/LDC97S44>

³British National Corpus: <http://www.natcorp.ox.ac.uk/>

⁴North American News Text Corpus: <https://catalog.ldc.upenn.edu/LDC95T21>

⁵The BN and WN compression datasets are hosted at <http://jamesclarke.net/research/resources>.

Each compression instance in these datasets consists of an input sentence from a document accompanied by the human-authored compressions—three per instance for the BN corpus and one each for the WN corpus. An example from each corpus is provided in Table 3.1. We filtered both corpora to eliminate instances in which input sentences had less than 2 tokens or more than 110 tokens⁶ in order to avoid parser failures. Using the same training/development/test splits as the evaluations in Clarke and Lapata (2008), this yields 880/78/404 compression instances respectively for the BN corpus and 953/63/603 for the WN corpus. Finally, because syntactic structure is not annotated in this corpus, we run the Stanford dependency parser⁷ over all sentences in the corpus and their reference compressions as a surrogate for gold-standard dependency parse trees.

3.1.1 Corpus analysis

As a consequence of the variation between spoken and written language, the two corpora under consideration pose distinct challenges for compression systems. The BN training dataset features sentences containing 20.4 tokens on average, with short utterance fragments including interruptions and greetings balanced by lengthy comma splices and run-on sentences. In contrast, the WN corpus consists of formal, edited language but longer sentences—28.0 tokens on average—which are more liable to cause errors in statistical parsing as well as data-driven compression. This motivates the use of separate compression models for these two corpora.

Figure 3.1 shows the distribution of instances with respect to the number of tokens dropped in the different reference compressions for the BN corpus and illustrates the variation in annotator decisions on compression over the same corpus. Nearly a quarter (23.6%) of the longest references in each instance feature no compression at all,⁸ although two-thirds of these instances were successfully compressed by at least one other annotator. We average

⁶These large sentences are rare and appear to originate from erroneous sentence segmentation in the original datasets supplied to annotators.

⁷Stanford dependency parser: <http://nlp.stanford.edu/software/>

⁸Paraphrasing Knight and Marcu (2002), we retain these instances in the corpora for training and testing in order that our systems learn not only *what* to compress but also *when* to compress.

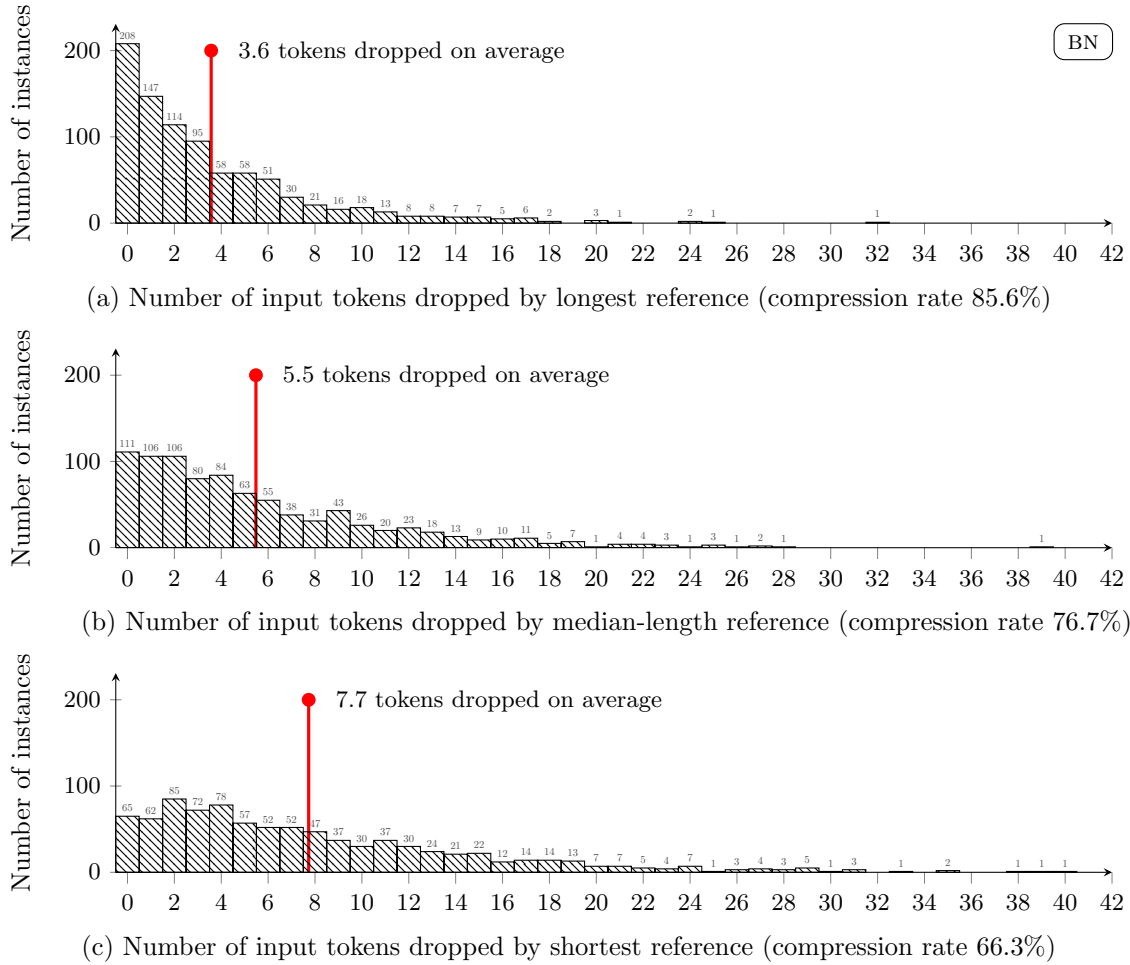


Figure 3.1: Distribution of instances in the BN training dataset with respect to the number of tokens dropped from the input sentence to produce (a) the longest reference compression, (b) the reference compression of median length, and (c) the shortest reference compression.

our evaluation measures across all references provided by annotators in order to account for the diversity of opinions regarding sentence compressibility.

Interestingly, the distribution of dropped tokens for the WN corpus in Figure 3.2 appears to follow that of the shortest reference from the BN corpus indicating that compression over written news text is relatively aggressive. This is also indicated by the average number of tokens dropped by reference compressions in the WN corpus (8.39) which is similar to the number of tokens dropped by the shortest BN references (7.73). The average human compression rate for a sentence is uncorrelated with the length of aggressively compressed

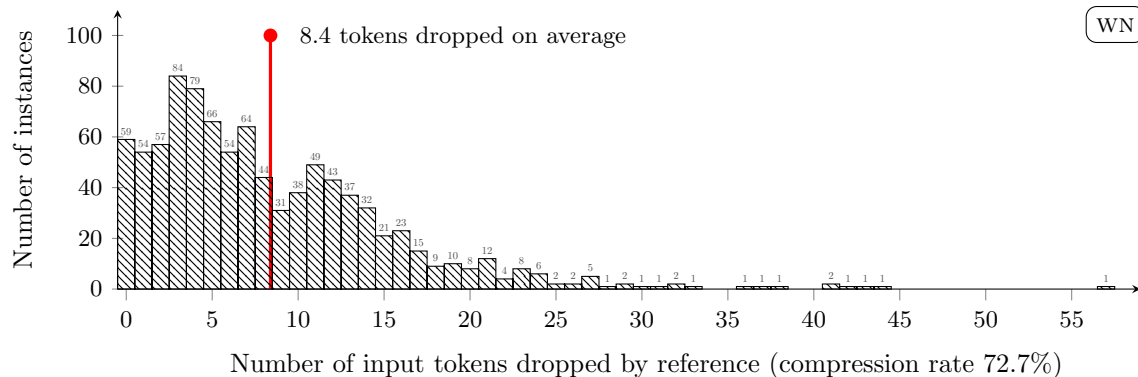


Figure 3.2: Distribution of instances in the WN training dataset with respect to the number of tokens dropped from the input sentence to produce the reference compression.

sentences such as the WN references (Pearson’s $r = 0.03$) and the shortest BN references ($r = -0.02$), mildly anticorrelated with the length of weaker compressions such as the longest ($r = -0.17$) and median-length ($r = -0.1$) BN references, and moderately anticorrelated with the length of input sentences from both the BN corpus ($r = -0.43$) and the WN corpus ($r = -0.33$).⁹

As the BN corpus has multiple references, we can also examine the agreement between human-generated compressions. Table 3.2 enumerates the rate of preservation of content words (nouns, verbs, adjectives, adverbs) and all words in reference compressions. We observe that the majority of preserved words are shared across all reference sentences, with more than 80% of nouns and verbs present in at least two. Annotators appeared to disagree more frequently on adjectives and prepositions (60% present in at least two references) and much more so with adverbs, which were dropped 40% of the time. Notably, at least two references agree on their syntactic roots in 97% of cases, in part because head verbs of the input sentence other than reporting verbs are typically preserved in compression. This appears to indicate that the reference sentences of the BN corpus broadly agree on the general information content to be preserved but also exhibit diversity when it comes to preserving modifiers which encode finer-grained aspects of information.

Because the human-authored compressions in these corpora do not involve word reordering or paraphrasing, many prior compression techniques assume that the syntactic structure

⁹All correlations are reported at $p < 0.05$ where the null hypothesis assumes no correlation.

	Words preserved in reference compressions %					
	Head verbs	Verbs	Nouns	Adjectives	Adverbs	All
All 3 references	72.59	65.08	61.29	43.08	26.89	55.83
2/3 references	11.06	16.33	19.71	21.87	16.41	18.80
1/3 references	7.48	9.91	11.27	18.05	17.28	12.72
No references	8.88	8.68	7.72	17.00	39.42	12.66

Table 3.2: Percentage of content words and all words that are preserved in compressed sentences by human annotators in the training portion of the BN corpus (Clarke and Lapata, 2006b).

of output compressions remains consistent with that of input sentences. For example, some inference formulations draw the dependencies of output compressions from the dependency tree of the input sentence (Martins and Smith, 2009; Filippova and Altun, 2013) while others rely on a subtree-deletion model in which compressions are achieved solely by pruning subtrees from a constituent parse tree of the input sentence (Berg-Kirkpatrick et al., 2011; Qian and Liu, 2013).

To test this hypothesis, we compared the Stanford dependency parses of input sentences to those of their human-authored compressions in the training portions of the Clarke and Lapata (2006b) corpora. The results are enumerated in Table 3.3. For the WN corpus, 86.4% of all dependency edges from the dependency trees of reference compressions were found to correspond to dependency edges in the input parse after normalizing for the length of the compressed sentence. However, only 319/953 sentences had their reference dependency trees wholly contained within the dependency tree of the input sentence, thereby implying that only 33.5% of the instances were reachable under a subtree deletion model.

Similar results were seen for the BN corpus which has three human compressions—often of different lengths—for every input sentence. 85.7% of dependency edges from the Stanford dependency trees over reference compressions overlap with edges from the trees over the corresponding input sentence after normalizing for sentence length. However, only 1270/2640 or 45.7% of the reference dependency trees were reachable using the edges in input trees.¹⁰ When this measure is averaged over the three reference compressions for each

¹⁰These reachability statistics are commensurate with those reported by Qian and Liu (2014) who ran the

	Dep overlap %	Reachability %
BN	85.65	30.00
WN	86.35	33.47

Table 3.3: Fraction of dependencies in Stanford parses of input sentences which are also present in parses of reference compressions alongside the fraction of *reachable* instances—reference parses contained entirely within input parses—from the BN and WN training datasets.

instance, only 30% of the reference dependency trees are reachable using the input trees.

These observations motivate a less restrictive approach to syntax in the compression task. The multi-structured inference strategy presented in the following sections therefore considers all possible dependency relations for an output compression alongside other structural representations of text.

3.2 Multi-Structured Compression

The text-to-text framework proposed here is motivated by the hypothesis that generating fluent text involves a consideration of the diverse structural relationships between tokens in both input and output sentences. Models for sentence compression often compose text from units that are larger than individual tokens such as n-grams which must be assembled into a valid factorization of a token sequence or dependency relations which are typically organized as a tree. However, our investigation is motivated by the notion that *both* these representations of a sentence—a sequence of tokens and a tree of dependency relations—may be meaningful when considering its underlying fluency and integrity. In other words, an approach for compressing a token sequence would benefit from simultaneously considering the compression of its dependency representation and vice versa.

In this section, we discuss the problem of recovering an optimal compression from a sentence as a linear optimization problem (cf. §3.2.1) over heterogenous structured outputs (cf. §3.2.2) that can be assembled into a consistent representation of a sentence (cf. §3.3). In

non-projective MST parser (McDonald et al., 2005b) over the entire (Clarke and Lapata, 2006b) compression corpus and noted that 47.6% the reference compressed sentences violated the subtree-deletion assumption.

the following sections, we consider rich linguistically-motivated features over these substructures (cf. §3.4) for which parameters can be estimated via supervised structured prediction (cf. §3.5). We then employ a widely-used dataset of sentence compressions to conduct an experimental evaluation of this framework (cf. 3.6).

3.2.1 Compression as linear optimization

We begin with some notation. Consider a compression problem involving a source sentence S from which compressions C can be constructed. The *maximum a posteriori* (MAP) inference problem is the task¹¹ of recovering the compressed sentence \hat{C} which is the most likely compression of S under some model parameters θ .

$$\hat{C} \triangleq \arg \max_C p_{\theta}(C|S) \quad (3.1)$$

In a discriminative setting, the conditional probability above is replaced with a linear or log-linear scoring function which is computed via features defined over S and C . Adopting a linear scoring function denoted by Δ , we rewrite (3.1) as

$$\begin{aligned} \hat{C} &\triangleq \arg \max_C \Delta(S, C) \\ &= \arg \max_C \theta^{\top} \phi(S, C) \end{aligned} \quad (3.2)$$

where we use $\phi(S, C)$ to denote some feature map over S and C , parameterized by a vector of learned weights θ .

Let $\mathcal{T} \triangleq \{t_i : 1 \leq i \leq n\}$ represent the set of tokens (including repetitions) in S and let $x_i \in \{0, 1\}$ represent a token indicator variable whose value corresponds to whether token t_i is present in the output sentence C . Collectively, these indicator variables comprise the incidence vector $\mathbf{x} \triangleq \langle x_1, \dots, x_n \rangle^{\top}$ which specifies an output token configuration equivalent to some subset of \mathcal{T} , i.e., the set of values of $\mathbf{x} \in \{0, 1\}^n$ has a bijection with $2^{\mathcal{T}}$.

Tractability in structured prediction problems involving text is generally achieved through strong factorization assumptions. For instance, if we were to consider a simplistic bag-of-tokens scenario in which sentences are viewed as collections of unordered tokens, we

¹¹More generally, the goal of MAP inference—a form of Bayesian inference—is to identify the assignment to every non-evidence random variable (given the evidence variables) that jointly maximizes the probability defined by the model.

can define a feature map ϕ —and consequently a compression score Δ —which factors over the input tokens. The highest-scoring compression under (3.2) can therefore be expressed as the token configuration that maximizes a linear combination of per-token scores, i.e., $\sum_i x_i \cdot \Delta_{\text{tok}}(S, i)$ where $\Delta_{\text{tok}}(S, i) \triangleq \boldsymbol{\theta}_{\text{tok}}^\top \boldsymbol{\phi}_{\text{tok}}(S, i)$ denotes a feature-based scoring function which measures the relative value of retaining token t_i in a compression of S .

One consequence of these strong independence assumptions is efficient MAP inference. An optimal token-factored compression can be trivially recovered in $O(n)$ time by dropping all input tokens t_i which incur negative scores $\Delta_{\text{tok}}(S, i)$. If output compressions are required to obey a predetermined compression rate,¹² the highest-scoring compression can be retrieved in $O(n \log n)$ time by sorting the input tokens by their scores. However, regardless of efficiency, the strong independence assumption used in this scenario is clearly unrealistic: a model that does not consider any relationship between output tokens cannot explicitly account for the ordering of output tokens or ensure that the resulting sentence remains grammatical.

3.2.2 Multi-structured objective

A natural solution to this problem is the inclusion of higher-order factorizations of linguistic structures such as n-grams in the scoring objective from (3.2). For clarity of exposition, we assume the use of trigrams without loss of generality. Let \mathcal{Y} represent the set of all possible trigrams that can be constructed from the tokens of S ; in other words $\mathcal{Y} \triangleq \{\langle t_i, t_j, t_k \rangle : t_i \in \mathcal{T} \cup \{\text{START}\}, t_j \in \mathcal{T}, t_k \in \mathcal{T} \cup \{\text{END}\}, i \neq j \neq k\}$ where the tokens in the output sentence are preceded by the special token START and followed by the special token END. When the order of tokens in the input must be preserved, the definition of \mathcal{Y} can be revised to include the constraint $i < j < k$ for every potential trigram $\langle t_i, t_j, t_k \rangle$.¹³ Following the notation for token indicators, let $y_{ijk} \in \{0, 1\}$ represent a trigram indicator variable for whether the

¹²The compression rate is the length of an output sentence normalized by the length of the input sentence, generally expressed as a percentage. Fixed compression rates are often imposed in sentence compression problems in order to avoid degenerate solutions (Clarke and Lapata, 2008) and have been shown to be critical in fair evaluations of compression systems (Napoles et al., 2011b). See §3.3.3 for more details.

¹³This condition is imposed for all compression problems in this dissertation.

sequence of tokens $\langle t_i, t_j, t_k \rangle$ is contiguous in the output sentence. The incidence vector $\mathbf{y} \triangleq \langle y_{ijk} \rangle_{\langle t_i, t_j, t_k \rangle \in \mathcal{Y}}$ thus represents some subset of the trigrams in \mathcal{Y} .

Similarly, let \mathcal{Z} represent the set of all possible dependency edges that can be established among the tokens of S , i.e., $\mathcal{Z} \triangleq \{\langle t_i, t_j \rangle : t_i \in \mathcal{T} \cup \{\text{ROOT}\}, t_j \in \mathcal{T}\}$ where the special token ROOT governs the head of the output dependency parse. As before, $z_{ij} \in \{0, 1\}$ represents a dependency indicator variable indicating whether t_j is a direct dependent of t_i in the dependency structure of the output sentence, and the corresponding incidence vector $\mathbf{z} \triangleq \langle z_{ij} \rangle_{\langle t_i, t_j \rangle \in \mathcal{Z}}$ represents a subset of the edges from \mathcal{Z} .

Using this notation, any output sentence C can now be expressed as a combination of some token, trigram and dependency configurations $\langle \mathbf{x}, \mathbf{y}, \mathbf{z} \rangle$. Defining Δ_{ngr} and Δ_{dep} analogously to Δ_{tok} for trigrams and dependency edges respectively, we rewrite (3.2) as

$$\begin{aligned} \widehat{C} &= \arg \max_{\mathbf{x}, \mathbf{y}, \mathbf{z}} \sum_{i: t_i \in \mathcal{T}} x_i \cdot \Delta_{\text{tok}}(S, i) \\ &\quad + \sum_{\substack{i, j, k: \\ \langle t_i, t_j, t_k \rangle \in \mathcal{Y}}} y_{ijk} \cdot \Delta_{\text{ngr}}(S, i, j, k) \\ &\quad + \sum_{\substack{i, j: \\ \langle t_i, t_j \rangle \in \mathcal{Z}}} z_{ij} \cdot \Delta_{\text{dep}}(S, i, j) \\ &= \arg \max_{\mathbf{x}, \mathbf{y}, \mathbf{z}} \mathbf{x}^\top \mathbf{\Delta}_{\text{tok}} + \mathbf{y}^\top \mathbf{\Delta}_{\text{ngr}} + \mathbf{z}^\top \mathbf{\Delta}_{\text{dep}} \end{aligned} \quad (3.3)$$

where $\mathbf{\Delta}_{\text{tok}} \triangleq \langle \Delta_{\text{tok}}(S, i) \rangle_{t_i \in \mathcal{T}}$ is used to compactly denote the vector of token scores for all tokens $t_i \in \mathcal{T}$ and $\mathbf{\Delta}_{\text{ngr}}$ and $\mathbf{\Delta}_{\text{dep}}$ represent similar vectors of scores for all trigrams and dependencies in \mathcal{Y} and \mathcal{Z} respectively.

The joint objective in (3.3) is an appealingly straightforward and yet fairly general formulation for the compression task. An n-gram factorization \mathbf{y} can be scored under probabilistic n-gram LMs as in Clarke and Lapata (2008) while a compressed dependency tree \mathbf{z} can be scored using a rich set of syntactic features including dependency labels and part-of-speech tags, similar to Filippova and Strube (2008a). However, unlike the bag-of-tokens scenario described previously, optimal solutions for \mathbf{y} and \mathbf{z} cannot be recovered efficiently due to their interdependence and the global nature of their underlying structures.¹⁴ Specifi-

¹⁴We discuss one particular constrained formulation for which efficient inference is possible in Chapter 5.

cally, we need to enforce the following conditions in order to obtain a usable token sequence from \mathbf{y} :

- Trigram variables y_{ijk} must be non-zero if and only if their corresponding word variables x_i , x_j and x_k are non-zero.
- The non-zero y_{ijk} must comprise a valid trigram factorization of a sentence in which the underlying token ordering avoids disjoint, cyclic or branching structures.

Similarly, a well-formed dependency tree \mathbf{z} will need to satisfy the following conditions:

- Dependency variables z_{ij} must be non-zero if and only if the corresponding word variables x_i and x_j are.
- The non-zero z_{ij} must form a directed rooted tree with no cycles and all edges oriented away from the root.¹⁵

Although we require that dependency variables z be consistent with trigram variables y , we do not require constraints over both of them as long as they agree on the token configuration in the output. The following section introduces our initial approach for recovering exact solutions to this problem through the use of integer linear programming.

3.3 Compression via Integer Linear Programming

A linear program (LP) is an optimization problem of the form

$$\begin{aligned} \max_{\mathbf{x} \in \mathbb{R}^d} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{Ax} \leq \mathbf{b} \end{aligned}$$

where \mathbf{x} is a vector of real-valued decision variables of interest, \mathbf{c} is a vector of corresponding coefficients, and the matrix \mathbf{A} and vector \mathbf{b} impose arbitrary linear coefficients on the

¹⁵Such a structure is variously referred to as an *out-tree*, *arborescence* or *branching* in the graph theory literature. However, the distinction between these directed structures and regular undirected trees is not critical here owing to the rarity of the latter in natural language representations. We therefore follow the terminology of the parsing community and simply use the more general term *tree* to refer to these directed structures hereafter.

permitted values of \mathbf{x} . If the problem is feasible and bounded, the optimum is attained at some vertex of the polyhedron that defines the constraint space. This property is exploited by a number of well-known techniques to solve LPs including the simplex algorithm as well as interior point methods which exhibit worst-case polynomial time complexity in the number of variables and constraints.

If decision variables are restricted to integer values, i.e., $\mathbf{x} \in \mathbb{Z}^d$, the problem is referred to as an integer linear program (ILP).¹⁶ Solving an ILP is NP-complete in the general case although there are some notable special cases: for instance, when the constraint matrix \mathbf{A} is totally unimodular¹⁷ and \mathbf{b} is integer valued, the vertices of the constraining polyhedron are all integral. In this case, the solution of the ILP is identical to that of its LP relaxation—the LP formed by omitting the integer constraints on \mathbf{x} —and this LP relaxation is said to be *tight*.¹⁸

Despite their worst-case exponential time complexity, ILPs are extensively used in practice to model a wide range of real-world optimization problems. High-performance ILP solvers—both free and commercial—are widely available and capable of recovering fast solutions to large ILPs, generally by first solving an LP relaxation and then searching for an integral solution using the branch-and-bound algorithm. Furthermore, the worst-case complexity of ILPs is less prohibitive for sentence-level natural language problems in which the number of variables and constraints is described by a low-order polynomial over the length of a sentence. Consequently, recent years have seen ILPs used in many structured natural language processing applications including dependency parsing (Riedel and Clarke, 2006; Martins et al., 2009), text alignment (DeNero and Klein, 2008; Chang et al., 2010; Thadani and McKeown, 2011a), multi-document summarization (McDonald, 2007; Lin and Bilmes, 2010) and a number of previous approaches to text-to-text generation including

¹⁶When *some* but not all of the decision variables are constrained to integer values, the problem is sometimes referred to as a mixed integer linear program (MILP) in the optimization literature. We include this class of problems in our usage of the term ILP.

¹⁷A matrix is said to be totally unimodular if the determinant of every square submatrix is in $\{-1, 0, 1\}$.

¹⁸Clarke and Lapata (2008) build on this notion by conjecturing that an ILP with a constraint matrices with entries confined to $\{-1, 0, 1\}$ —a necessary but not sufficient condition of total unimodularity—might be more likely to yield integral solutions.

paraphrasing (Dras, 1998), sentence compression (Clarke and Lapata, 2008; Filippova and Strube, 2008a; Filippova and Altun, 2013), document compression (Martins and Smith, 2009; Clarke and Lapata, 2010; Berg-Kirkpatrick et al., 2011; Chali and Hasan, 2012; Woodsend and Lapata, 2012), sentence fusion (Filippova and Strube, 2008b; Elsner and Santhanam, 2011; Thadani and McKeown, 2011b), sentence simplification (Woodsend and Lapata, 2011; Angrosh et al., 2014) and other similar tasks (Woodsend and Lapata, 2010; Woodsend et al., 2010; Cheung and Penn, 2014).

3.3.1 Enforcing tree structure

We begin by considering the problem of defining constraints to ensure that the structure specified by the dependency configuration \mathbf{z} represents a valid tree and remains consistent with the token configuration \mathbf{x} . The following conditions must hold for any directed graph structure to be a valid tree:

1. There is a single node at the root of the tree with no incoming edges.
2. Every non-root node must have exactly one incoming edge.
3. The structure has no cycles.

To satisfy the first condition and designate some token from \mathcal{T} as the root of an output tree, we include a special node—the `ROOT` from the definition of \mathcal{Z} in §3.2.2—which will take it as a dependent. We address `ROOT` as an auxiliary token $t_0 \notin \mathcal{T}$ in the constraints below. Ensuring that the output dependency configuration \mathbf{z} is rooted by a single token from \mathcal{T} can be accomplished with a simple constraint over all dependencies z_{0j} governed by `ROOT`, restricting only one of them to be *active*¹⁹ in any solution to the ILP.

$$\sum_j z_{0j} = 1 \tag{3.4}$$

The second condition is similarly local to every token variable and its incoming dependency edges. Each active token x_j must be accompanied by exactly one active dependency z_{ij}

¹⁹A binary variable is *active* when assigned a value of 1 in the optimal solution to the ILP and *inactive* when assigned a value of 0.

while inactive tokens must have no active dependencies.

$$x_j - \sum_i z_{ij} = 0, \quad \forall 1 \leq j \leq n \quad (3.5)$$

The above constraints do not preclude directed cycles in \mathbf{z} . The final condition that output structures be acyclic does not seem immediately enforceable due to its non-local nature, i.e., the assignment of each dependency variable appears dependent on the remaining dependency structure. This condition is equivalent to requiring that the structure be fully connected, as noted by Magnanti and Wolsey (1994) who proposed LPs and ILPs which recover optimal spanning trees in directed graphs by enforcing connected *commodity flow* between all pairs of tokens. Martins et al. (2009) have exploited these flow formulations in ILPs to recover non-projective and nearly-projective spanning trees for dependency parsing.

A similar intuition informs this work. Auxiliary variables γ_{ij} are defined to carry some real-valued *dependency commodity* between all pairs of tokens $\langle t_i, t_j \rangle$ where $t_i \in \mathcal{T} \cup \{\text{ROOT}\}$, $t_j \in \mathcal{T}$ and $i \neq j$. The flow network is made to correspond to the active dependency configuration by constraining these variables to be zero whenever t_j is not a dependent of t_i in the output sentence.

$$\gamma_{ij} \geq 0, \quad \forall 0 \leq i \leq n, \quad (3.6)$$

$$1 \leq j \leq n, i \neq j$$

$$\gamma_{ij} - \Gamma_{\max} z_{ij} \leq 0, \quad \forall 0 \leq i \leq n, \quad (3.7)$$

$$1 \leq j \leq n, i \neq j$$

where Γ_{\max} is the maximum amount of flow that the γ_{ij} variables may carry and serves as an upper bound on the number of tokens in the output sentence. Because we use flow to avoid cyclical structure and not to specify spanning trees, Γ_{\max} can simply be set to an arbitrary large value.

In typical flow networks, nodes *consume* a fixed quantity of flow from their incoming flow variables and transmit the remainder to their outgoing flow variables. Structural connectivity can then be established by constraining flow to originate from a single source, i.e., the root of the tree. However, unlike the spanning trees required in dependency parsing (Martins et al., 2009), we cannot assume that all input tokens will participate in a compression solution—indeed, such a scenario would imply a total absence of compression

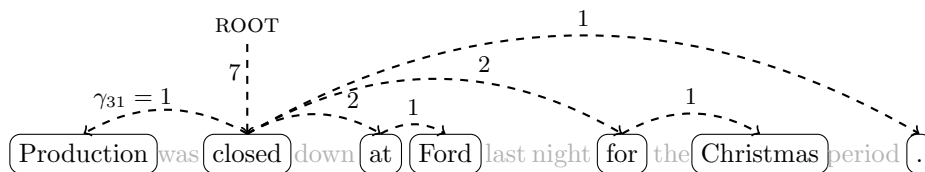


Figure 3.3: Dependency commodity values for a flow network accompanying a tree-based compression solution. Dashed lines denote all non-zero flow variables γ_{ij} .

in the output sentence. We therefore introduce per-token constraints to make only active tokens consume flow from their incoming flow variables while ignoring inactive tokens.

$$\sum_i \gamma_{ij} - \sum_k \gamma_{jk} = x_j, \quad \forall 1 \leq j \leq n \quad (3.8)$$

Figure 3.3 illustrates a flow network corresponding to a compression of the input sentence “*Production closed down at Ford last night for the Christmas period.*” The seven active output tokens must each consume a single unit of flow according to (3.8); therefore, seven units of flow are drawn from ROOT and routed to the active tokens through a connected structure. Moreover, since (3.7) makes this flow structure isomorphic to the dependency configuration \mathbf{z} , the dependency constraints (3.4)–(3.5) preclude re-entrant edges and further constrain the structure to be a directed tree. This is elaborated in the following propositions.

Proposition 3.3.1 *A dependency variable z_{ij} is active iff the corresponding $\gamma_{ij} > 0$.*

Proof By (3.6) and (3.8), every active x_j requires a positive total incoming flow $\sum_i \gamma_{ij}$.

Because (3.7) ensures that each positive-valued γ_{ij} is accompanied by an active dependency variable z_{ij} and (3.5) requires an active token x_j to have exactly one active incoming dependency, it follows that z_{ij} is active iff $\gamma_{ij} > 0$. ■

Proposition 3.3.2 *The token configuration represented by \mathbf{x} is fully connected by \mathbf{z} .*

Proof For every flow variable $\gamma_{jk} > 0$, either $t_j \in \mathcal{T}$ —in which case there exists some incoming flow variables with flow $\sum_i \gamma_{ij} > \gamma_{jk}$ —or t_j represents ROOT which, by definition, has no incoming flow. The auxiliary ROOT is thus the only node in the network with no constraint on outgoing flow (other than the Γ_{\max} limit). By (3.8), active nodes reduce flow while inactive nodes don’t impact flow, so ROOT is the only

possible source of flow for the network. Active tokens in \mathbf{x} must therefore draw a directed path of positive flow from ROOT.

Proposition 3.3.1 requires positive flow variables to be accompanied by active dependencies so \mathbf{z} must connect all active tokens in \mathbf{x} to ROOT. Moreover, since (3.4) restricts ROOT to have only one outgoing dependency, \mathbf{x} remains connected by \mathbf{z} when ROOT is disregarded. ■

Proposition 3.3.3 *The dependency structure specified by \mathbf{z} contains no cycles.*

Proof Assume that there exists an active cycle \mathcal{C} composed of some subset of dependencies which are active in \mathbf{z} . We use $\mathcal{T}_{\mathcal{C}} \subseteq \mathcal{T} \cup \{\text{ROOT}\}$ to represent the tokens which participate in the cycle.

If \mathcal{C} is a directed cycle (also known as a *circuit*) with all participating edges oriented in the same direction, every token in $\mathcal{T}_{\mathcal{C}}$ has exactly one active incoming dependency edge and thus $\text{ROOT} \notin \mathcal{T}_{\mathcal{C}}$. By (3.5), these tokens cannot also have incoming dependencies governed by tokens outside $\mathcal{T}_{\mathcal{C}}$. The cycle thus remains disconnected from ROOT, violating Proposition 3.3.2.

If \mathcal{C} is not a directed cycle, at least one token in $\mathcal{T}_{\mathcal{C}}$ would have more than one incoming dependency and constraint (3.5) would be violated. Thus, \mathcal{C} cannot exist as a directed or undirected cycle and \mathbf{z} is acyclic. ■

Figure 3.4 represents this property visually. A cycle cannot be introduced through multiple positive incoming flow variables on any token node because Proposition 3.3.1 ties those flow variables to active dependencies and constraint (3.5) permits only one of these to be active per token. Cycles that satisfy this constraint must be in violation of (3.8) which requires all active tokens to consume flow. More generally, the following condition is sufficient to prevent *only* directed cycles in flow networks regardless of whether multiple incoming dependencies—and therefore undirected cycles—are permitted in the output.

Lemma 3.3.1 (Directed acyclicity) *A flow network specified by constraints (3.6) and (3.8) has no directed cycles if for all nodes $1 \leq j \leq n$*

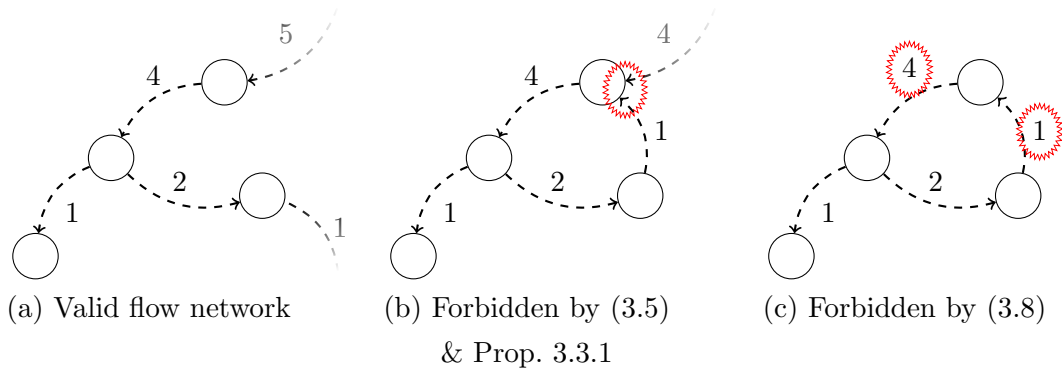


Figure 3.4: An illustrative flow network with edge weights indicating non-zero flow featuring (a) consistent flow and no directed cycles, (b) a cycle that preserves flow but needs multiple incoming edges (c) a cycle with one incoming edge for each node but consequently inconsistent flow.

$$\min_{i:\gamma_{ij}>0} \gamma_{ij} \begin{cases} \geq \max_k \gamma_{jk} + x_j, & \exists i \text{ s.t. } \gamma_{ij} > 0 \\ = 0, & \text{otherwise} \end{cases} \quad (3.9)$$

Proof This condition holds trivially for inactive nodes because they have zero incoming and outgoing flow according to (3.8). For any active node x_j , the directed acyclicity condition simply implies that every outgoing flow variable γ_{jk} will contain a smaller amount of flow than *any* of its positive-valued incoming flow variables γ_{ij} . In other words, the amount of flow must strictly decrease as we traverse the flow variables which a cycle, which is not possible for any cycle of finite length. ■

Because (3.5) restricts every node to have only one incoming flow variable with positive flow, the directed acyclicity condition in (3.9) is imposed on every token by constraint (3.8). We exploit this property in Chapter 6 when extending the ILP to support semantic parse structures which take the form of directed acyclic graphs.

In summary, the constraints (3.4)–(3.8) ensure that the dependency configuration \mathbf{z} specifies a rooted dependency tree. The resulting ILP remains fairly compact and requires only $O(n^2)$ variables and constraints for an input sentence with n tokens. Additional constraints can be introduced easily and enable various extensions ranging from the linguistically-motivated rules of Clarke and Lapata (2008) to richer parameterizations, e.g., §7.2.3 demonstrates that left and right attachments in the output can be parameter-

ized separately without increasing the asymptotic size of the program. This ILP can also straightforwardly be extended to produce *labeled* dependencies z_{ij}^b , where $b \in \mathcal{B}$ denotes a dependency label by using $O(n^2|\mathcal{B}|)$ variables and constraints.

3.3.2 Assembling valid n-gram factorizations

We now turn to the problem of enforcing structure in the n-gram configuration \mathbf{y} . In the first-order (bigram) scenario, this task reduces to ensuring that the output sentence forms a directed *path*²⁰ in an adjacency graph where each edge represents a bigram of adjacent words in an output sentence. The following conditions would hold for any directed graph structure to form a valid path.

1. There is a single start node with no incoming edges and one outgoing edge.
2. There is a single end node with one incoming edge and no outgoing edges.
3. Every non-terminal node must have exactly one incoming and outgoing edge.
4. The structure has no cycles.

Because a directed path can be viewed simply as a directed tree in which every node has exactly one outgoing edge, we can employ flow variables and constraints similar to those described previously in order to ensure that the output structure is acyclic. The remaining conditions can then be generalized to n-gram variables of any order as long as all active adjacent token nodes and the edges linking them form a path in the flow network. We illustrate this approach with trigram variables y_{ijk} .

The first and second conditions above express the notion that a valid output sentence will begin and terminate with exactly one token. We can enforce this through special nodes—the START and END from the definition of \mathcal{Y} in §3.2.2—that precede the first token and follow the final token of a sentence respectively. In the constraints below, we treat START as an auxiliary token $t_0 \notin \mathcal{T}$ ²¹ and END as an auxiliary token $t_{n+1} \notin \mathcal{T}$. The conditions

²⁰We assume all paths are *simple*, i.e., they consist of an open walk in a graph with no repeated vertices and edges.

²¹Although the ROOT from §3.3.1 also uses the token index 0, it is of little consequence as START and ROOT never interact directly in constraints; they can be viewed as different names for the same node.

can then be expressed by constraints that require exactly one active sentence-initial n-gram variable y_{0jk} and sentence-final n-gram variable $y_{ij(n+1)}$ in the output.

$$\sum_{j,k} y_{0jk} = 1 \quad (3.10)$$

$$\sum_{i,j} y_{ij(n+1)} = 1 \quad (3.11)$$

We generalize the third condition to n-grams of order μ by specifying that a token can only be active in the solution when, for $1 \leq p \leq \mu$, there is exactly one active n-gram in the solution which contains this token in position p . Note that this does not always hold for n-grams of order $\mu > 2$ due to the way terminal n-grams featuring START and END are defined. For instance, in an output sentence “*The cat sat on the mat.*” the first word *The* is in position 1 for the trigram *The cat sat* and in position 2 for the trigram START *The cat* but does not appear in position 3 for any trigram. More generally, in a valid linear ordering of tokens and $\forall p \in 1 \dots \mu - 2$, there can be no n-grams that feature the last $\mu - p - 1$ tokens in position p or the first $\mu - p - 1$ tokens in position $(\mu - p + 1)$. However, this is easily tackled computationally by assuming that the terminal n-gram replaces these missing n-grams for near-terminal tokens, e.g., that the trigram START *The cat* features the word *The* in both positions 2 and 3 for the example above. In the trigram case, this leads to the following constraints for each token variable x_j .

$$x_j - \sum_i y_{ij(n+1)} - \sum_{k,l} y_{jkl} = 0, \quad \forall 1 \leq j \leq n \quad (3.12)$$

$$x_j - \sum_{i,k} y_{ijk} = 0, \quad \forall 1 \leq j \leq n \quad (3.13)$$

$$x_j - \sum_k y_{0jk} - \sum_{h,i} y_{hij} = 0, \quad \forall 1 \leq j \leq n \quad (3.14)$$

The final condition which requires \mathbf{y} to be acyclic can be enforced with a first-order flow network identical to the one used to enforce a dependency tree for \mathbf{z} . Auxiliary variables γ'_{ij} are defined to carry some real-valued *adjacency* commodity between all pairs of tokens $\langle t_i, t_j \rangle$ where $t_i \in \mathcal{T} \cup \{\text{START}\}$, $t_j \in \mathcal{T} \cup \{\text{END}\}$ and $i \neq j$. The flow network is made to correspond to active n-gram variables by constraining adjacency flow to be zero whenever

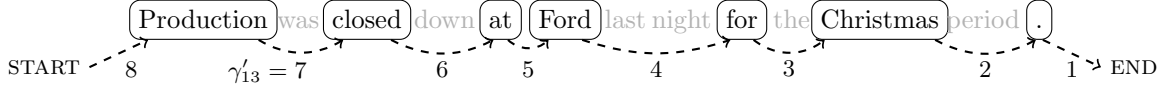


Figure 3.5: Adjacency commodity values for a flow network accompanying a path-based compression solution. Dashed lines denote all non-zero flow variables γ'_{ij} .

t_j does not immediately follow t_i in the output sentence.

$$\gamma'_{ij} \geq 0, \quad \forall 0 \leq i \leq n, \quad (3.15)$$

$$1 \leq j \leq n+1, \quad i \neq j$$

$$\gamma'_{ij} - \Gamma'_{\max} \sum_k y_{ijk} \leq 0, \quad \forall 0 \leq i \leq n, \quad (3.16)$$

$$1 \leq j \leq n, \quad i \neq j$$

$$\gamma'_{jk} - \Gamma'_{\max} \sum_i y_{ijk} \leq 0, \quad \forall 1 \leq j \leq n, \quad (3.17)$$

$$1 \leq k \leq n+1, \quad j \neq k$$

where Γ'_{\max} is an arbitrarily large limit on the flow that the γ'_{ij} variables may carry. Finally, active tokens consume adjacency flow in the same manner as they do dependency flow, thereby ensuring that the n-gram solution \mathbf{y} and the dependency solution \mathbf{z} agree on the tokens in the output compression.

$$\sum_i \gamma'_{ij} - \sum_k \gamma'_{jk} = x_j, \quad \forall 1 \leq j \leq n \quad (3.18)$$

Figure 3.5 illustrates an adjacency flow network corresponding to the compression example from the previous section. The flow constraints (3.15) and (3.18) produce a connected structure over \mathbf{x} which is tied to the n-gram configuration \mathbf{y} by (3.16)–(3.17) and consequently coerced into a path by (3.10)–(3.14). This is demonstrated by the following propositions.

Proposition 3.3.4 *An n-gram variable y_{ijk} is active iff the corresponding $\gamma'_{ij} > 0$ and $\gamma'_{jk} > 0$.*

Proof By (3.15) and (3.18), every active x_j requires a positive total incoming flow $\sum_i \gamma'_{ij}$.

Since (3.16) ensures that each positive-valued γ'_{ij} is accompanied by an n-gram variable y_{ijk} and (3.13) requires an active token x_j to constitute the middle token in exactly one active n-gram variable y_{ijk} , it follows that any y_{ijk} is active iff $\gamma'_{ij} > 0$.

The same reasoning can be employed for the final token x_k of an active n -gram variable y_{ijk} via (3.14) to complete the proof. ■

Proposition 3.3.5 *The n -gram configuration \mathbf{y} forms a tree rooted at START.*

Proof This follows via analogy to Proposition 3.3.3 which establishes that the dependency configuration \mathbf{z} forms an acyclic structure that is rooted at ROOT using a similar flow network. The constraints (3.13) and (3.14) are equivalent to (3.5) as they restrict tokens to participate in only one incoming adjacency relationship. By substituting START for ROOT and n -grams \mathbf{y} for dependencies \mathbf{z} , Proposition 3.3.3 can be used to establish that \mathbf{y} forms an acyclic tree rooted at START. ■

Proposition 3.3.6 *The adjacency structure underlying \mathbf{y} has no branches.*

Proof Assume there is a branching structure in \mathbf{y} and let x_j denote the terminal node in a branch off the main path (which terminates in END). By definition, there is no active n -gram $y_{ij(n+1)}$ such that x_j is followed by END otherwise x_j would be part of the primary path and not a branch. Now, by (3.13), there must be some active n -gram y_{ijk} in which x_j occupies the central position. Therefore, x_j cannot be a terminal node in the branch and \mathbf{y} must consist of a single non-branching path terminating in END. ■

The constraints (3.10)–(3.18) therefore ensure that the \mathbf{y} represents a valid n -gram factorization of a token sequence, or, more plainly, a sentence. For an input sentence with n tokens, the resulting ILP requires $O(n^2)$ constraints over $O(n^3)$ variables—or more generally, $O(n^{\max(2,\mu)})$ variables when working with n -grams of order μ . We also note that the notion of pairing a first-order flow network to higher-order n -gram variables can be extended to support higher-order dependency variables in \mathbf{z} as well.

3.3.3 Enforcing a compression rate

A crucial measure of automated compression approaches is the degree of aggressiveness with which they reduce sentence content. Moreover, statistical approaches to the task often admit explicit restrictions on the size of output sentences—a notion introduced in early

efforts toward text-to-text generation (Dras, 1997; Dras, 1998). These restrictions can sometimes be critical to compression systems; for instance, compression techniques based on probabilistic language models rely on a lower-bound on output sentence size to avoid producing empty compressed sentences (Clarke and Lapata, 2008).

The *compression rate* of an output compression is the ratio of the number of output tokens or words to the number of input tokens,²² usually expressed as a percentage. Following the variable notation above and using $\mathbf{1}$ to denote a vector of all ones, the compression rate ω for an input sentence with n tokens is

$$\omega \triangleq \frac{\mathbf{1}^\top \mathbf{x}}{n} \quad (3.19)$$

Until recently, it has been commonplace to simply report the intrinsic compression rates of automated systems in task evaluations rather than explicitly restrict the length of output sentences. However, a recent analysis of approaches to evaluating automated compression systems (Napoles et al., 2011b) has shown a strong correlation between system-imposed compression rates and human judgments of compression quality, thereby concluding that comparisons of systems which compress at different rates are unreliable. Consequently, the imposition of an extrinsic compression rate must be accounted for in any inference algorithm for sentence compression.

In the ILP formulation described previously, the compression rate of a sentence can be bounded to the range $(\omega_{\min}, \omega_{\max})$ via global constraints on the active token indicators.

$$\sum_i x_i \geq n \cdot \omega_{\min} \quad (3.20)$$

$$\sum_i x_i \leq n \cdot \omega_{\max} \quad (3.21)$$

The flow networks also implicitly impose an upper bound for output sentence lengths via the coefficients Γ_{\max} and Γ'_{\max} which respectively limit the capacity of a dependency flow variable in (3.7) and that of an adjacency flow variable in (3.16)–(3.17). Because only active

²²This definition is equivalent to the reciprocal of the familiar *compression ratio* from the information theory literature and is consequently an occasional source of confusion. It should be noted that systems which compress text at a *higher* compression rate produce longer output sentences and thereby induce a *smaller* degree of compression than otherwise.

tokens consume flow and all terminal nodes—the leaves of \mathbf{z} and END respectively—must have zero outgoing flow, the assignments

$$\Gamma_{\max} = \Gamma'_{\max} - 1 \triangleq n \cdot \omega_{\max} \quad (3.22)$$

are equivalent to the constraint (3.21) above.

3.4 Features

As seen in §3.2.2, the scoring function that guides inference for compression is a linear function over the feature map ϕ which is defined as a concatenation of feature maps for each substructure, i.e., token features ϕ_{tok} , n-gram features ϕ_{ngr} and dependency edge features ϕ_{dep} . We expect that features in text-to-text generation problems will be broadly similar and enumerate the potential categories that they might fall into below.

3.4.1 Feature categories

1. **Saliency:** Good compressions might require specific words or relationships between words to be preserved, highlighted, or perhaps explicitly rejected. This can be expressed through features on token variables or other substructures that indicate their *a priori* salience for consideration in an output sentence. These can be composed of external indicators of salience such as queries, contextual information such as the *tf*idf* within a document collection or even syntactic information such as part-of-speech (POS) tags or dependency labels; for example, the latter may help statistical models determine whether head verbs are more relevant in compressions than relative clause verbs or auxiliary verbs.
2. **Fluency:** Of paramount importance in any statistical generation task are features that capture how the presence of a given substructure contributes to the overall fluency of a sentence. Syntactic information such as POS context and dependency labels are likely useful for such scenarios. Probabilities and log-likelihoods under LMs are also reasonable candidates for fluency features.
3. **Fidelity:** One might expect that many substructures in the input sentence will appear unchanged in the output sentence, e.g., §3.1.1 indicates that 85% of dependencies in

reference compressions correspond to dependencies in the input sentence. This can be leveraged by including binary features which indicate whether a substructure was seen in the input. More generally, one might consider features that relate any substructure to input substructures, e.g., whether a potential dependency edge links words which were connected via a directed path in the input dependency tree.

4. **Pseudo-normalization:** A drawback of using linear models for generation problems is an inability to normalize potential output structures by their size, as this would lead to a non-linear objective function. However, we can invoke the MT strategy of employing word penalty features for this purpose. These features take the form of simple indicators for each substructure whose learned parameters are intended to balance out the biases in output length which are induced by other features.

We now enumerate the specific feature templates used by the substructure feature maps for the experiments in §3.6. These templates were tuned on the development portions of the datasets described in §3.1 using the structured learning algorithm from §3.5.1. To minimize training overhead and overfitting, we chose to avoid feature templates with a high degree of conjugation and lexical features over open-class words;²³ however, richer or sparser templates may be preferable for different datasets or circumstances.²⁴ Crucially, we made an effort to ensure that the features over n-gram and dependency variables relied on the same low-level signals (POS tags and Stanford dependency labels) and were roughly commensurate in expressiveness so as to ensure a fair comparison between models based on these features for the experiments described in §3.6.

In order to aid generalization, certain groups of fine-grained POS tags (nouns, verb, adjectives, adverbs + particles, and wh-words) also induce coarse POS categories (N, V, J, R and W respectively). Similarly, every dependency label also induces its ancestors in the

²³We identify open-class word by their POS tags—expected to be members of the tag set for nouns (NN, NNS, NNP, NNPS), verbs (VB, VBD, VBN, VBG, VBP, VBZ), adjectives (JJ, JJR, JJS), adverbs (RB, RBR, RBS), cardinal numbers (CD) or interjections (UH).

²⁴As an example, recent work by Qian and Liu (2014) employs rich lexical features with a reimplemention of our multi-structured inference technique.

hierarchy of Stanford dependency labels.²⁵ Furthermore, the value of each scale-dependent feature is recorded absolutely and also normalized by the length of the input sentence. This is done in order to induce some robustness to sentence length variation in the models.

3.4.2 Token features

The feature templates used in the feature map $\phi_{\text{tok}}(S, i)$ for a token t_i consist of the following lexico-syntactic indicators:

- t_j itself if t_j is a function word (i.e., not an open-class word such as nouns, verbs, adjectives and adverbs) for $j \in i - 1, \dots, i + 1$.
- The POS of t_i conjoined with the label of the dependency edge incident on t_i in the Stanford dependency parse of the input sentence.
- The POS tag of t_j for $j \in i - 2, \dots, i + 2$.
- The POS tag sequence of the segment $\langle t_j, t_{j+1} \rangle$ for $j \in i - 2, \dots, i + 1$.
- The POS tag sequence of the segment $\langle t_j, t_{j+1}, t_{j+2} \rangle$ for $j \in i - 2, \dots, i$.

as well as the following fidelity and morphological features:

- Whether t_i appears in the input. This is always 1 for extractive compressions and hence also serves as a token penalty feature for approximate length normalization.
- Whether t_i is capitalized—a crude indication of whether it refers to a proper noun or named entity.
- The relative position of t_i in a sequence of capitalized words if it is capitalized.
- Whether t_i lies within parentheses, a useful indicator of non-salience drawn from a constraint by Clarke and Lapata (2008).
- Whether t_i is a negation such as *not* or *n't*.

Many compression systems (Clarke and Lapata, 2008; Filippova and Strube, 2008a) also use a measure based on $tf*idf$ which derives from the informativeness score of Hori and Furu (2004), but we did not find this measure to be relevant in our development experiments.

²⁵Stanford dependencies manual (de Marneffe and Manning, 2008): http://nlp.stanford.edu/software/dependencies_manual.pdf

3.4.3 n-gram features

We define feature templates that apply to n-grams of any order μ but illustrate them with trigrams here. The feature templates used in the feature map $\phi_{\text{ngr}}(S, i, j, k)$ for a trigram $\langle t_i, t_j, t_k \rangle$ consist of the following lexico-syntactic indicators:

- t_l if t_l is a function word for $l \in \{i, j, k\}$.
- The POS tags of the tokens in the n-gram $\langle t_i, t_j, t_k \rangle$.
- If the n-gram order $\mu > 2$, the POS tags of all contiguous bigrams in the n-gram, e.g., $\langle t_i, t_j \rangle, \dots, \langle t_j, t_k \rangle$.
- The labels of dependency edges incident on t_i, t_j and t_k in the Stanford dependency parse of the input sentence.

as well as the following fidelity and LM features:

- Whether the contiguous sequence $\langle t_i, t_j, t_k \rangle$ appears in the input.
- If $\mu > 2$, whether any of the bigrams $\langle t_i, t_j \rangle, \dots, \langle t_j, t_k \rangle$ appear in the input.
- The probability of $\langle t_i, t_j, t_k \rangle$ under an LM.²⁶

We did not observe improvement when using the log-likelihood of n-grams under an LM as a feature—with or without normalization—despite this yielding the convenient interpretation of scoring the output sentence under an LM.

3.4.4 Dependency features

The feature templates used in the feature map $\phi_{\text{dep}}(S, i, j)$ for a dependency edge $\langle t_i, t_j \rangle$ consist of the following lexico-syntactic and fidelity indicators for both participating tokens:

- t_l if t_l is a function word for $l \in \{i, j\}$ conjoined with a binary indicator for the fidelity of the dependency edge, i.e., whether the edge is present in the dependency tree of the input sentence.

²⁶We use an LM trained with Kneser-Ney smoothing over the Gigaword 3 corpus distributed by the LDC at <https://catalog.ldc.upenn.edu/LDC2007T07> using the SRILM toolkit (Stolcke, 2002) available from <http://www.speech.sri.com/projects/srilm/>.

- The POS tag of t_i conjoined with the POS tag of t_j as well as the fidelity of the edge and its orientation, i.e., $sign(t_j - t_i)$.
- The POS tags of the input tokens lying between t_i and t_j conjoined with the POS tag of the dependent token t_j as well as its fidelity and orientation.
- If the dependency edge $\langle t_i, t_j \rangle$ is present in the parse of the input sentence, the label of that edge.

as well as features to indicate the likelihood of the dependent t_j appearing in the output:

- The POS tag of t_j conjoined with the label of its incoming dependency in the input dependency tree.
- The POS context of t_j indicated by the POS tags of the tuples $\langle t_{j-1}, t_j \rangle$, $\langle t_j, t_{j+1} \rangle$ and $\langle t_{j-1}, t_{j+1} \rangle$.

Various additional features including indicators which considered chunk boundaries and paths between tokens in the input dependency tree offered at best inconsistent improvements and were disregarded in our final experiments.

3.5 Parameter Estimation

We approach the problem of sentence compression as a supervised structured prediction task, assuming the availability of a training dataset \mathcal{D} consisting of tuples $\langle S, C_{\text{ref}} \rangle$ representing an input sentence S and a reference output compression C_{ref} for each compression problem. Let \widehat{C}_{θ} indicate the best compression hypothesis recovered under the linear objective from (3.3) with some parameterization θ . The learning problem is the task of recovering parameters θ^* which minimize the expected loss over \mathcal{D} given an appropriate loss function $\mathcal{L}(S, C_{\text{ref}}, \widehat{C}_{\theta})$. Assuming the loss function is meaningful in penalizing poor compressions and our training sample adequately captures the true distribution of compression problems, learned parameters which minimize this expected loss over the training dataset may also be effective for recovering good compressions of new sentences.²⁷

²⁷These are naturally rather strong assumptions which rarely apply to real-world datasets. A wide variety of statistical learning techniques have been developed with the sole aim of improving generalization when training data is small or noisy, e.g., parameter regularization, max-margin techniques, etc.

3.5.1 Structured perceptron

We rely on a variant of the structured perceptron of Collins (2002) for all experiments described in this dissertation. A sketch of the training procedure is provided in Algorithm 1. The perceptron algorithm of Rosenblatt (1958) was originally developed for binary classification problems and recovers parameters that correctly classify all training examples in a bounded number of iterations if the data is linearly separable in the feature space. Collins (2002) generalizes the perceptron to the structured prediction setting—in which it targets 0/1 loss—and also proposes averaging the parameters from every iteration of perceptron training as an approximation of the voted perceptron (Freund and Schapire, 1999) which takes advantage of data that is linearly separable with large margins. Perceptron-based approaches are relatively simple to implement, can be easily parallelized (McDonald et al., 2010; Zhao and Huang, 2013), require no hyperparameters to be tuned and accommodate convenient optimizations for approximate inference.²⁸ Regardless, the inference techniques we describe can also be used with other structured learners.²⁹

In our implementation of the structured perceptron, training examples are grouped into *minibatches* within which inference can be parallelized—this preserves the mistake bound guarantee and often speeds up convergence (Zhao and Huang, 2013). For the experiments in §3.6, we use minibatches of size 4 in a load-balancing approach, i.e., every minibatch is assigned instances of similar size in order to minimize processor idle time. The learning rate is kept constant and performance is monitored over held-out development corpora in order to tune features and prevent overfitting on the training dataset.

3.5.2 Deriving features for reference compressions

The learning algorithm requires the computation of features $\phi(S, C_{\text{ref}})$ over one or more reference compressions for parameter updates, which in turn requires the substructures—

²⁸When inference takes the form of inexact search, early-update (Collins and Roark, 2004) and max-violation (Huang and Feyong, 2012) strategies can be used to speed up convergence.

²⁹Popular alternatives encountered in the text-to-text generation literature include max-margin techniques such as the structured SVM (Tsochantaridis et al., 2004) and MIRA (Crammer and Singer, 2003; McDonald et al., 2005a), which can also utilize the k -best inference hypotheses for each parameter update.

Algorithm 1 Structured perceptron with parameter averaging (Collins, 2002)

Input: training dataset \mathcal{D} , feature map ϕ , number of epochs e , learning rate schedule $\eta \in \mathbb{R}^e$

Output: vector of learned parameters θ

```

1:  $\theta_{(0)} \leftarrow \mathbb{0}^{|\phi|}$ 
2:  $k \leftarrow 0$ 
3: for epoch  $i$  in  $1, \dots, e$  do
4:   for instance  $\langle S, C_{\text{ref}} \rangle \in \mathcal{D}$  do
5:      $\hat{C} \leftarrow \arg \max_C \theta_{(k)}^\top \phi(S, C)$ 
6:     if  $\hat{C} \neq C_{\text{ref}}$  then
7:        $\theta_{(k+1)} \leftarrow \theta_{(k)} + \eta_i \left( \phi(S, C_{\text{ref}}) - \phi(S, \hat{C}) \right)$ 
8:        $k \leftarrow k + 1$ 
9:   if converged then break

return average weights  $\frac{1}{k} \sum_j \theta_{(j)}$ 

```

tokens, n-grams and dependencies—present in C_{ref} to be resolved to the substructures derived from S , i.e., those indicated by x , y and z variables. In other words, a unique *derivation* of the reference compression is necessary in order for the training procedure to determine the relative merit of every feature in reproducing this compression. However, recovering such a derivation can be challenging because of the ambiguity inherent in relating output tokens to input tokens.

In the general case when unique derivations are not easily recoverable for a reference sentence, the learning algorithm can be revised to incorporate latent variables which indicate the derivation (Liang et al., 2006a). However, in text-to-text generation tasks like sentence compression, a unique derivation can simply be identified by inducing a monolingual token alignment from the output text to the input text and inferring corresponding alignments for larger substructures. Moreover, in an extractive compression setting with no paraphrasing or reordering, we can forgo sophisticated alignment techniques such as those described in §2.3.1 in favor of a simple multi-phase approach which progressively reduces the ambiguity of a token mapping. We consider four phases:

1. **Lexical overlap:** First, we iterate through the reference tokens in C_{ref} and, for each, identify all matching tokens in S . If any reference token has more than one match, the derivation remains ambiguous and the following steps are considered.
2. **Monotonicity:** In the extractive compression scenario, we can assume that there is no reordering of input tokens in the output sentence and can therefore filter out all mappings which violate this assumption. This is accomplished by iterating over all reference tokens and ensuring that each maps to an input token with an index greater than at least one of the input tokens mapped to by the previous reference token.³⁰ This step is skipped in text-to-text problems for which reordering is possible such as the sentence fusion approach discussed in Chapter 7.
3. **Syntactic overlap:** We then iterate over the ambiguous tokens and consider their syntactic governors in their respective dependency trees. If the governors of the reference token and one of its potential input matches are already aligned, the two tokens are also paired up. If this does not resolve all remaining ambiguities, we proceed to the following step.
4. **Contextual overlap:** The remaining ambiguous reference tokens are iterated over once again. For each of these, let $\mathcal{M}_p \subset \mathcal{T}$ indicate the set of input tokens which are already matched unambiguously to reference tokens and appear within a context window of size p around a potential match for some token t_{ref} from C_{ref} . Of all potential input matches for this reference token, the preferred match is the one with the minimum total distance—measured in tokens—between input candidates that match t_{ref} and all aligned tokens in \mathcal{M}_p . This step is repeated for increasing values of context window size $1 \leq p \leq 15$. Like monotonicity, contextual overlap is assumed to be unreliable whenever input tokens may be reordered.

Ambiguous token mappings were rare after these steps. In the training partitions of the Clarke and Lapata (2006b) datasets, the few remaining ambiguous mappings consist of unresolvable punctuation and determiners surrounding dropped phrases as shown in Table 3.4. In these cases, we simply average the features of every possible reference derivation.

³⁰If a particular token mapping is resolved in any step, the participating input token is removed from other ambiguous mappings.

Input	Lawyers representing the nuns from the Our Lady of the Passion Monastery ◉ Daventry ◉ Northamptonshire ◉ were yesterday granted a judicial review of the Ministry of Agriculture order to slaughter the chickens .
Compression	Lawyers representing the nuns from Our Lady of the Passion Monastery ◉ were granted a review of the Ministry of Agriculture order .

Table 3.4: An example of mapping ambiguity between the tokens in an input sentence and its reference compression from the WN corpus of Clarke and Lapata (2006b). Faded tokens in the input sentence indicate those dropped from the reference compression. The circled comma in the reference compression may be resolved to any of three commas in the input sentence.

3.6 Experiments

In order to evaluate the performance of the multi-structured compression framework, we ran compression experiments over the broadcast news (BN) transcriptions and written news (WN) documents collected by Clarke and Lapata (2006b) and Clarke and Lapata (2007) respectively. As described in §3.1, the BN corpus contains 880, 78 and 404 compression instances for training, development and testing respectively, where each instance features 3 human-authored reference compressions of an input sentence after filtering out single-word sentences and sentences longer than 110 words. Similarly, the WN corpus consists of 953, 63 and 603 compression instances respectively after the same filtering procedure, although each of these instances have only one reference compression.

As these corpora do not include paraphrasing or word reordering, we follow evaluations in machine translation as well as previous work in sentence compression (Unno et al., 2006; Clarke and Lapata, 2008; Martins and Smith, 2009; Napoles et al., 2011b) in evaluating system performance with automated measures against reference compressions, specifically F_1 measures over n-grams and of the unlabeled dependency edges in the output dependency tree \mathbf{z} . We also report F_1 of dependency edges produced by parsing system output with the Stanford parser as well as the Robust Accurate Statistical Parsing (RASP) toolkit (Briscoe et al., 2006).

In particular, evaluations based on F_1 over RASP grammatical relations are frequently

used in sentence compression research following the observation by Clarke and Lapata (2006b) that this measure correlates well with human judgments of compression quality. Eliciting 20 judgments per compressed sentence, they report a correlation of 0.532 over a 60-compression sample from the BN corpus (20 instances \times 3 compression techniques) and a correlation of 0.575 on a similarly sized sample from the Ziff-Davis corpus. The result with RASP F_1 strongly outperforms simple string accuracy, an edit-distance measure proposed as a baseline metric for natural language generation (Bangalore et al., 2000) and is relatively close to the correlation between human raters elicited through leave-one-out resampling: 0.746 on the BN corpus and 0.679 on the Ziff-Davis corpus. Clarke and Lapata (2006b) recommend the RASP toolkit because of its ability to parse both full sentences and sentence fragments as well as its robustness in analyzing semi-grammatical compression outputs. Furthermore, unlike n-grams and Stanford dependencies, RASP structures are not used to generate features and are not considered in our inference objectives. We therefore view RASP F_1 as the primary measure of system performance in these evaluations.

Without *a priori* compression rate restrictions, different techniques yield output sentences of different lengths. For comparing different compression systems, the standard practice in early compression literature has been to simply report the average intrinsic compression rate of each system under study. However, in a survey of evaluation methods for sentence compression, (Napoles et al., 2011b) identify a strong correlation between a system compression rate and human judgments of compression quality and conclude that comparisons between systems which compress with different degrees of aggressiveness are not adequate to characterize their relative performance. Consequently, all our experiments impose a restriction on the compression rate of an output sentence to ensure that observed differences between the systems under study are meaningful. Rather than use a fixed compression rate for the corpus, we choose to restrict output sentences to be the same length as the reference compression accompanying each input sentence. For the BN corpus—which contains 3 references per instance—the the median reference compression rate is usually targeted for experiments, although we also examine the shortest and longest reference compression rates in §3.6.4.

The following sections describe specific experiments over these corpora. All ILPs were

solved with Gurobi 6.0,³¹ a high-performance commercial-grade solver under active development. We follow Martins et al. (2009) in using LP relaxations of all ILPs for quick inference during training, assuming algorithmic separability (Kulesza and Pereira, 2007) for these compression problems.

3.6.1 Joint inference

The primary hypothesis we consider in this evaluation is whether the use of multiple structural perspectives and joint inference offers performance gains in the compression task. In order to address this, we use the automated measures and experimental environment described above to compare systems that rely on just n-gram or dependency structures as well as joint models that incorporate both structures. We consider one exemplar of each of these categories as well as an unsupervised baseline as enumerated here.

- **LM-3gr**: A reimplementation of the mostly unsupervised³² ILP of Clarke and Lapata (2008) which infers order-preserving trigram variables parameterized with log-likelihood under an LM and a *tf*idf*-based significance score for token variables inspired by Hori and Furui (2004). In addition, this model incorporates several targeted syntactic constraints based on grammatical relations derived from RASP parses (Briscoe et al., 2006) designed to encourage fluent output.
- **DP-2gr**: A reimplementation of the path-based dynamic program of McDonald (2006), which is described in detail in §4.1.2.³³ As an exact inference technique, this is equivalent to the ILP described in §3.3.2 for bigram variables ($\mu = 2$) albeit with a polynomial runtime bound for inference.

³¹The Gurobi solver is available with restricted and academic licensing at <http://www.gurobi.com>.

³²Although the main components of this system—the LM, a heuristic significance score and linguistically-motivated constraints—are all produced without an annotated corpus, a single parameter is tuned to trade off the LM score with the significance score.

³³For consistent comparisons with the other systems, our reimplementation does not include the *k*-best inference strategy presented in McDonald (2006) for learning with MIRA. We note, however, that ILP-based inference can also be made to yield *k*-best solutions via cutting planes as described by Clarke and Lapata (2008), albeit with runtime increased by a factor of *k*.

- **ILP-dep**: The ILP described in §3.3.1 for inference of edge-factored dependency structure along with constraints for imposing a compression rate from §3.3.3, excluding n-gram variables and their corresponding features.
- **ILP-2gr-dep**: The joint inference approach which combines the constraints from §3.3.1 to produce an edge-factored dependency tree with §3.3.2 to yield a bigram-factored token ordering for the output sentence. We opt to use only bigrams for this evaluation so that the ILP remains compact and scales quadratically with the size of the input sentence.

Table 3.5 contains the results for these systems on the BN test dataset which contains 404 instances, each with three human-authored reference compressions. These reference compressions may be identical or may vary in both length and content as shown by the example in Table 3.1. In order to account for different perspectives on compression, we generate a separate instance from each reference compression when training, i.e., our training dataset consists of 3 x 880 compression instances.³⁴ At test time, we average the per-instance evaluation measures with respect to reference sentences and restrict the output sentence to the median reference sentence length, resulting in an average compression rate of 77.26% over the BN test dataset.

We observe a significant gain for supervised systems under all measures of quality against the mostly unsupervised LM-3gr approach of Clarke and Lapata (2008). The latter approach relies on numerous linguistically-motivated constraints³⁵ to enforce fluency in the output

³⁴In our development tests, an alternative approach in which each original compression instance is considered once, output sentences are restricted to the median reference length and the features of every reference compression are simply averaged for each input sentence during training—disregarding the difference in compression rate across references—performs identically to the results above.

³⁵These linguistically-motivated constraints are defined over the RASP parse of the input sentence and summarized here: (i) an active verb in the solution activates its subject and object and vice versa, (ii) an active non-clausal modifier or determiner activates its head, (iii) negations and possessive modifiers activate their heads and vice versa, (iv) the sentence must contain at least one active verb if the input sentence contains one, (v) the word introducing a prepositional phrase or subordinating clause must be active if at least word within the syntactic constituent is active and vice versa, (vi) active head words which are conjoined activate the coordinating conjunction, (vii) words in parentheses are always dropped, (viii)

BN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		path	LM-3gr (CL08)	74.96	60.60			
DP-2gr (McD06)	82.94		72.84	61.08	52.65	-	70.96	66.34
tree	ILP-dep	82.70	70.05	56.81	47.94	75.76	70.88	65.25
path + tree	ILP-2gr-dep	82.85	72.66	60.87	52.87	75.73	73.82	68.56

Table 3.5: Experimental results for the BN corpus averaged over 3 reference compressions per instance. All systems were restricted to compress to the size of the median reference compression yielding an average compression rate of 77.26%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test and the paired t-test ($p < 0.05$).

sentence. However, these hard constraints were responsible for infeasibility in the ILP for one instance from the BN test dataset and three from the WN test dataset. The significant advantage of supervised approaches such as DP-2gr over LM-3gr when compression rates are fixed has also been observed by Napoles et al. (2011b) in human evaluations over the WN corpus.

Comparing supervised techniques, we find that the multi-structured ILP-2gr-dep system equals or surpasses the single-perspective systems DP-2gr and ILP-dep in all evaluation measures considered. Measures of n-gram F₁ with respect to reference compressions are statistically indistinguishable for DP-2gr and ILP-2gr-dep—the two supervised techniques that parameterize an n-gram factorization of the output compression—under Wilcoxon’s signed rank test and the paired t-test. Similarly, the difference between ILP-dep and ILP-2gr-dep, in terms of F₁ of edges in the system-generated tree with respect to the Stanford dependency parses over reference compressions is also not significant. It therefore appears that the joint approach adopts the advantages of both single-perspective techniques—namely, parameterizations of bigrams and dependency structures—under corresponding evaluation measures.

personal pronouns are always preserved.

WN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		path	LM-3gr (CL08)	66.68	51.59	39.33	30.54	-
DP-2gr (McD06)	75.36 [†]		63.40	52.15 [†]	42.97	-	63.08	59.43
tree	ILP-dep	75.74	60.07	46.84	37.48	69.08	64.33	58.49
path + tree	ILP-2gr-dep	76.15	64.47	53.63	45.15	69.52	67.35	61.82

Table 3.6: Experimental results for the WN corpus with all systems compressing to the size of the reference compression, yielding an average compression rate of 70.24%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$) and † indicates entries with a different outcome under the paired t-test ($p < 0.05$).

This result may appear somewhat unsurprising since n-gram and dependency measures are being directly or indirectly optimized in the inference procedure. However, the joint inference strategy also exhibits advantages when output compressions are parsed by the Stanford and RASP parsers and compared to the corresponding parses of reference sentences. Under F₁ measures over edges for both Stanford dependency trees and RASP grammatical relation graphs, ILP-2gr-dep significantly outperforms both DP-2gr and ILP-dep despite including no additional features beyond the ones employed by these systems. Multi-structured inference therefore appears to have a holistic effect on high-level measures of compression quality.

Table 3.6 contains the results for these systems on the 603 compression instances of the WN test dataset, which features longer sentences—27.7 words on average as opposed to 19.2 words for the BN test dataset—and a single reference compression per instance. We restrict system-generated compressions to be the length of each reference sentence, resulting in an average compression rate of 70.24%. Owing to the longer sentences and more aggressive compression rate, the results on this corpus are noticeably lower across all systems when compared to those on the BN corpus.

This evaluation largely echoes the performance trends observed in the previous experiments over the BN corpus. Here, the multi-structured ILP-2gr-dep approach outperforms all single-perspective systems and statistical significance is more readily observed under

the signed rank test although less so under the paired t-test.³⁶ Interestingly, `ILP-2gr-dep` demonstrates a statistically significant improvement in terms of F_1 over 4-grams over all other systems under both our significance tests—a result that might be attributed to either the diminished efficacy of `DP-2gr` on this corpus or the relative strength of `ILP-dep` as indicated by its strong result for unigram F_1 . As with the BN corpus, `ILP-2gr-dep` produces trees that are statistically similar to `ILP-dep` under dependency F_1 and significantly outperforms all other systems in F_1 of syntactic relations generated by parsing the system-generated compressions with the Stanford and RASP parsers.³⁷

3.6.2 Content-bearing words

As n-gram measures do not distinguish between content-bearing words and function words, we examine the precision and recall of open-class words—nouns, verbs, adjectives and adverbs—as a proxy for the content in compressed output. The results for the BN corpus are presented in Table 3.7. with similar results for the WN corpus in Table 3.8.

This analysis reveals notable disparities among different parts of speech with respect to their accuracy in identification for supervised compression. While precision and recall of nouns and verbs remains high ($> 60\%$) over both corpora, the measures are fairly low ($< 40\%$) for adjectives and even lower ($< 20\%$) for adverbs. Even though lexical features are not used for any of these POS categories (cf. 3.4), it appears that salient nouns and verbs are relatively easy to identify in the context of a sentence while modifiers like adjectives and adverbs remain ambiguous. In our development experiments, introducing lexical features for adjectives and adverbs did not noticeably affect results on these measures or the overall performance measures, perhaps indicating the limited generalizability of such features when

³⁶We defer to the non-parametric Wilcoxon’s signed rank test for which the differences between paired samples of F_1 measures need not be normally distributed—a requirement for the paired t-test—although both tests assume these samples are drawn from a symmetric distribution. The non-parametric test therefore offers a decreased sensitivity to outliers.

³⁷Our RASP F_1 results for Clarke and Lapata (2008) in Table 3.5 outperform their reported F_1 results by about 10% in absolute terms across both BN and WN corpora, even though their systems produce compressions at a slightly favorable average compression rate. We suspect that this discrepancy might stem from differences in our Gigaword-trained LM or improvements in recent versions of RASP.

BN		Nouns		Verbs		Adjectives		Adverbs	
Structures	Inference	P%	R%	P%	R%	P%	R%	P%	R%
path	LM-3gr (CL08)	72.58	65.66	75.44	73.48	31.25	30.32	19.81	20.50
	DP-2gr (McD06)	77.75	80.04	78.62	80.10	36.87	34.95	20.04	17.98
tree	ILP-dep	76.32	81.45	77.51	83.77	34.39	32.55	18.21	16.43
path + tree	ILP-2gr-dep	76.47	78.30	77.87	82.38	36.30	34.49	19.49	18.39

Table 3.7: Precision and recall of content-bearing words with respect to reference compressions for the BN corpus. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

WN		Nouns		Verbs		Adjectives		Adverbs	
Structures	Inference	P%	R%	P%	R%	P%	R%	P%	R%
path	LM-3gr (CL08)	68.83	60.08	79.55	71.54	30.60	30.07	13.26	13.82
	DP-2gr (McD06)	74.71	76.18	81.09	86.97	30.66	27.16	13.63	13.35
tree	ILP-dep	75.19	78.75	81.93	91.26	24.31	19.98	13.82	12.77
path + tree	ILP-2gr-dep	75.51	74.78	82.28	90.61	32.41	29.11	14.94	14.07

Table 3.8: Precision and recall of content-bearing words with respect to reference compressions for the WN corpus. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

training models over fewer than a thousand instances.

Turning to the nouns and verbs, we observe that precision generally remains fairly consistent across systems with DP-2gr yielding a significant advantage for the BN corpus and ILP-2gr-dep producing an insignificant gain for the WN corpus. In both corpora, however, ILP-dep exhibits a significant lead in the recall of nouns and verbs. Most notably, verb recall for ILP-dep improves by 3–5% over DP-2gr, leading us to conjecture that one of the primary contributions of dependency structures in the multi-structured ILP-2gr-dep is in enhancing the recall of verbs in the output compression.

3.6.3 Example output

Tables 3.9 and 3.10 contain examples of system output from the four systems described thus far over the BN test dataset and WN test dataset respectively. We chose examples for which all systems produced different—and usually erroneous—outputs in order to examine the practical effect of different structural approaches to sentence compression. This is therefore not intended to be a representative sample with respect to system performance over the test datasets.

Examining the data, we observe that **LM-3gr** tends to prefer deeply-nested noun phrases and chains of frequently-occurring function words to the exclusion of named entities—rare under an LM—an meaningful syntactic structure despite the imposition of linguistically-motivated constraints. We conjecture that these constraints may be too restrictive when compression is required to be aggressive, e.g., the constraint that a subject and object must accompany their governing verb occasionally results in all three being dropped rather than preserved.

The supervised bigram approach **DP-2gr** performs well in practice but is often observed making errors characteristic of a system with no parameterization of long-range dependencies. For instance, predicates and constituent heads are often dropped unexpectedly and clauses are often terminated abruptly so that the model can preserve the fidelity of high-scoring fragments of text while satisfying the compression budget—a phenomenon which can be seen in examples (a) and (b) from Table 3.9. Nevertheless, this often results in readable compressions as seen in example (c) from Table 3.10.

The dependency-based approach **ILP-dep** unsurprisingly favors the removal of modifiers and whole clauses and consequently often produces output sentences that differ from **DP-2gr**. Since this approach relies on an edge-factored representation, modifier removal introduces errors such as aggressive dropping of determiners and punctuation as in example (b) from Table 3.9 and examples (a) and (b) from Table 3.10. Another source of error is the removal of deeply-nested dependents such as prepositional phrases despite retaining the preposition.

The combination of these two perspectives in the joint inference approach of **ILP-2gr-dep** often results in compressions with improved readability. Although this system is prone to the same errors as **DP-2gr** and **ILP-dep**—especially when the compression rate is low—

BN		
(a)	Input	Now, Panetta says that the White House has taken steps to make sure that something like this never happens again .
Reference	(i)	Panetta says that the White House has taken steps to make sure that this never happens again .
Reference	(ii)	Panetta says that the White House has taken steps to make sure that something like this never happens again .
Reference	(iii)	the White House has taken steps to make sure this never happens again .
LM-3gr		, that the House has taken steps to make sure that something like this never happens .
DP-2gr		Panetta says that the White House has taken steps to make sure something like this never .
ILP-dep		Panetta says that the White House has taken steps to make sure something like never happens .
ILP-2gr-dep		Panetta says that the White House has taken steps to make sure something never happens again .
(b)	Input	He makes a living on the downside of physical fitness , but like most medical experts , Dr. Thomas Branch , of Emory University , firmly believes that the benefits of exercise far outweigh the risk of getting hurt .
Reference	(i)	He makes a living on the downside of physical fitness , but Dr. Thomas Branch , of Emory University , believes that the benefits of exercise outweigh the risk .
Reference	(ii)	like most medical experts , Dr. Thomas Branch , of Emory University , believes that the benefits of exercise outweigh the risk of getting hurt .
Reference	(iii)	He makes a living on the downside of physical fitness , but like most medical experts , Dr. Thomas Branch , of Emory University , firmly believes that the benefits of exercise far outweigh the risk of getting hurt .
LM-3gr		He makes a living on the downside of physical fitness , like most medical experts , , that the benefits of exercise far outweigh the risk of getting hurt .
DP-2gr		physical fitness , but like most medical experts , Dr. Thomas Branch , of Emory University , believes that the benefits of exercise outweigh the risk of getting hurt .
ILP-dep		He makes a living on downside of fitness , but Dr. Thomas Branch , of Emory University believes that the benefits of exercise outweigh the risk of getting hurt .
ILP-2gr-dep		He makes a living on the downside , but Dr. Thomas Branch , of Emory University , believes that the benefits of exercise outweigh the risk of getting hurt .

Table 3.9: Examples of system compressions for instances from the BN test dataset.

WN	
(a) Input	He died last Thursday at his home from complications following a fall , said his wife , author Margo Kurtz .
Reference	He died at his home following a fall , said his wife , author Margo Kurtz .
LM-3gr	He died last Thursday at his home from complications a fall , said his wife , .
DP-2gr	He died his home from complications following a fall , said his wife , Margo Kurtz .
ILP-dep	He died Thursday at from complications following a fall , said his wife author Margo Kurtz .
ILP-2gr-dep	He died last Thursday at his home from complications following a fall , said his wife .
(b) Input	Sir Cyril Taylor , the Government 's adviser on CTCs , who had earlier been succesful in persuading Mr Baker to commit more government funds to the 20 schools , had been hoping to get more money for a new round of schools .
Reference	Sir Cyril Taylor , Government adviser on CTCs , had been hoping to get more money for a new round of schools .
LM-3gr	, on , had been in to more government to the 20 schools , to get money for a new of schools .
DP-2gr	Sir Cyril Taylor , the Government 's adviser on CTCs been succesful in persuading Mr Baker had been hoping to get schools .
ILP-dep	Sir Cyril Taylor the Government 's adviser on CTCs succesful in persuading Mr Baker to commit funds had hoping to get money .
ILP-2gr-dep	Sir Cyril Taylor , the Government 's adviser on CTCs had been hoping to get more money for a round of schools .
(c) Input	Tens of thousands of traditional mud-built cob buildings are still in use throughout Devon and Cornwall .
Reference	thousands of cob buildings are still in use throughout Devon and Cornwall .
LM-3gr	Tens of thousands of traditional cob buildings are in use throughout and .
DP-2gr	Tens of mud-built cob buildings are in use throughout Devon and Cornwall .
ILP-dep	Tens of thousands of mud-built cob buildings are in throughout Devon Cornwall .
ILP-2gr-dep	Tens of thousands of mud-built cob buildings are in use throughout Devon .

Table 3.10: Examples of system compressions for instances from the WN test dataset.

BN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		path	LM-3gr (CL08)	69.65	52.01	37.36	29.77	-
DP-2gr (McD06)	78.92		66.45	53.03	44.02	-	63.58	58.88
tree	ILP-dep	78.49	61.81	46.96	38.06	71.02	66.44	57.76
path + tree	ILP-2gr-dep	79.00	66.54	53.73	45.47	71.48	69.47	62.30

Table 3.11: Experimental results for the BN corpus with all systems restricted to compress to the size of the *shortest* reference compression, yielding an average compression rate of 66.81%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

supervised training produces a system that appears robust to some of the challenges faced by either of the component systems in isolation. Even when outputs are inaccurate or incomplete, the resulting sentence remains often relatively interpretable as seen in example (b) from Table 3.9 and (c) from 3.10.

3.6.4 Varying the compression rate

An issue of particular interest in sentence compression problems is the variation of the quality of a solution under different compression rates. While the evaluations described thus far do not account for model performance outside the reference compression rates chosen by human annotators, performance is known to deteriorate as systems are constrained to compress aggressively. We therefore conduct additional experiments with a range of compression rates in order to examine their effect on output sentence quality. The BN corpus is most appropriate for these experiments as it features three human-authored compressions per instance with varying compression rates (cf. §3.1.1) and thus accommodates diverse interpretations of salient content. System-generated compressions were previously evaluated under the median reference compression rate in Table 3.5; here, we revisit the evaluation using other compression rates for reference compressions.

Table 3.11 contains evaluation results for the BN test dataset when each instance is restricted to compress to the *minimum* reference compression rate while Table 3.12. contains

BN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		path	LM-3gr (CL08)	78.38	66.85	54.07	45.63	-
DP-2gr (McD06)	83.93		76.54	65.90	58.00	-	75.67	71.45
tree	ILP-dep	83.93	75.20	63.85	55.33	77.34	75.27	70.38
path + tree	ILP-2gr-dep	83.99	76.70	65.97	58.14	77.44	76.74	72.40

Table 3.12: Experimental results for the BN corpus with all systems restricted to compress to the size of the *longest* reference compression, yielding an average compression rate of 86.01%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

results under the *maximum* reference compression rate. No new models are trained for these experiments; we simply reuse the models from §3.6.1 which were trained over all reference sentences in the training dataset. For consistency, the evaluation measures in these tables are always averaged over all three reference compressions—regardless of whether the reference sentence abides by the imposed compression rate—and then macro-averaged over the test dataset. Comparing these results alongside Table 3.5 which presents the same evaluation using the *median* reference compression rate, we see a clear trend of improvement for automated metrics as the output compression rate increases. This echoes the observation by Napoles et al. (2011b) that evaluation measures are correlated fairly strongly with output compression rate, thereby motivating the requirement for fixed compression rates when comparing different systems.

These different evaluation scenarios appear fairly consistent in their appraisal of the compression systems under consideration. All supervised techniques continue to outperform LM-3gr which relies largely on a trigram LM and a heuristic significance score. The joint inference approach of ILP-2gr-dep appears to produce compressions that are statistically equivalent or better than those produced by single perspective systems under the corresponding evaluation measures, i.e., similar to the bigram-based DP-2gr under n-gram F₁ with respect to the reference compressions and to the dependency edge-based ILP-dep under dependency edge F₁ with respect to the reference parses.

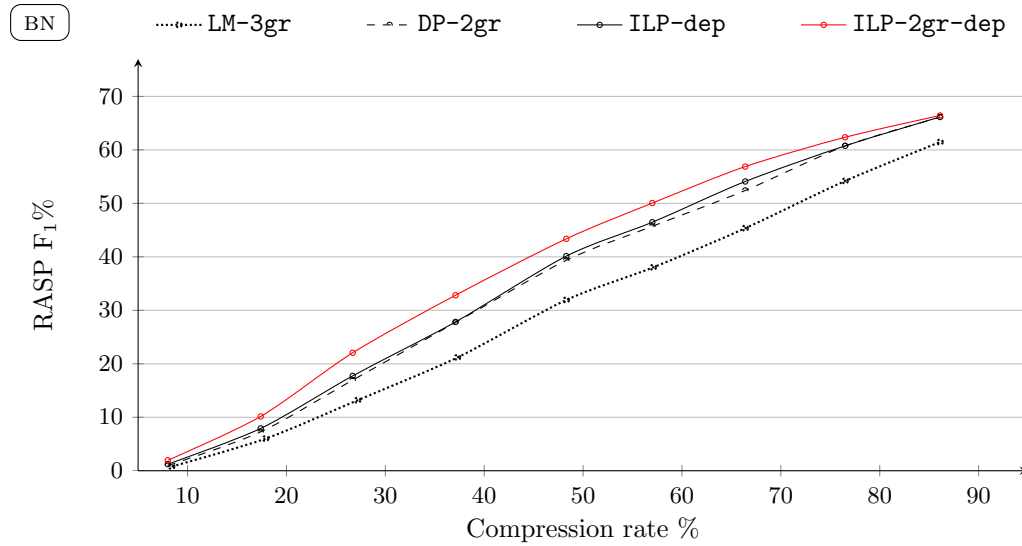


Figure 3.6: Variation in RASP F_1 with imposed compression rate for the BN corpus. All datapoints plotted at average output compression rates after rounding down to token counts.

When considering high-level automated measures involving parsing the system-generated compressions, however, we observe that the results for all systems appear to converge as the average compression rate increases. When considering F_1 of grammatical relations produced by the RASP parser, `ILP-2gr-dep` exhibits a strong absolute improvement of 3.5% over `DP-2gr` when restricted to the minimum compression rate. This advantage shrinks to 2.5% when imposing the median reference compression rate and further to a statistically-insignificant 1% when imposing the maximum reference compression rate. A similar trend is visible when using F_1 over dependencies from the Stanford parser: a strong 6% improvement over `DP-2gr` when using the minimum reference compression rate shrinks to an insignificant 1% improvement when using the maximum rate.

In order to explore this further, we conducted additional evaluations on the same dataset with *fixed* compression rates ranging from 10% to 90% in decile increments. This differs from the previous evaluations in that we can no longer assume that at least one reference sentence is the same length as the output compression; consequently, these evaluation measures may be considered less reliable. However, we assume that averaging over three reference compressions provides a useful approximation of performance and a meaningful relative comparison despite the absence of length-matched references. We focus on RASP F_1 for

this test owing to its widespread use in the literature and the robustness of RASP in the presence of disfluent text (Clarke and Lapata, 2006b).

The results in Figure 3.6 show a fairly consistent trend across compression rates.³⁸ The performance of all systems appears similar at very low output compression rates, diverges noticeably as the rate increases to 20% and begins to converge once again when the rate increases to 80%. This echoes our observations from experiments with reference compression rates and implies that a comparison of sentence compression systems would benefit from datasets produced under an aggressive compression rate.

The advantage of the supervised systems over `LM-3gr` appears to persist even at output compression rates of 90%. Among the supervised systems, the performance of `DP-2gr` and `ILP-dep` is similar in all tests with the latter having a small but sustained advantage, likely due in part to the similarity between RASP grammatical relations and the Stanford dependencies considered in its inference objective. Finally, the joint approach `ILP-2gr-dep` exhibits a clear advantage over the other systems, peaking with an absolute margin of about 5% at an output compression rate of 40%.

3.6.5 Higher-order n-grams

A useful aspect of the inference approach described in §3.2 is the ability to introduce n-gram variables of any size as opposed to just order-preserving bigrams (McDonald, 2006) and trigrams (Clarke and Lapata, 2008). In the models presented thus far, we have used a bigram factorization in order to keep the size of the resulting program quadratic in the size of the input sentence. Here, we examine the performance of trigram variables in supervised compression systems through the following additional models:

- `ILP-3gr`: The ILP described in §3.3.2 for trigram variables ($\mu = 3$) using the additional trigram features described in §3.4.3.
- `ILP-3gr-dep`: The corresponding joint inference approach which combines the constraints from §3.3.1 to produce an edge-factored dependency tree with §3.3.2 to yield

³⁸Since output sentence lengths are chosen by rounding down the imposed compression rate to the nearest token, the observed compression rate across the test dataset is lower than the rate imposed. The former rate is used to plot the data in the chart above.

BN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		path	DP-2gr	82.94	72.84	61.08	52.65	-
ILP-3gr	82.76		73.06	61.88	53.98	-	71.86	67.58
path + tree	ILP-2gr-dep	82.85	72.66	60.87	52.87	75.73	73.82	68.56
	ILP-3gr-dep	83.21	73.67	62.54	54.46	76.40	75.21	69.48

Table 3.13: Performance variation with n-gram size for the BN corpus with all systems restricted to compress to the size of the median reference compression. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

a trigram-factored token ordering for the output sentence.

Evaluation results over the BN corpus are in Table 3.13. We observe a significant improvement for both ILP-3gr and ILP-3gr-dep over their bigram counterparts under trigram and 4-gram F₁ with respect to reference sentences. While the two trigram-based systems are not statistically distinct under n-gram or RASP F₁, ILP-3gr-dep improves significantly over ILP-2gr-dep in F₁ over dependency edges—both in its output tree as well as when output compressions are parsed—thereby reinforcing the marginal utility of edge-factored dependency structures even when paired with higher-order n-grams.

Evaluation results over the WN corpus are presented in Table 3.14. In contrast to the BN results, the joint ILP-3gr-dep improves significantly over the trigram-only ILP-3gr in all measures other than unigram F₁. It also demonstrates significant improvement in bigram, trigram and 4-gram F₁ over the bigram-based ILP-2gr-dep although both joint models perform similarly under F₁ measures over syntactic relations. When considered alongside the results from content word analysis in §3.6.2, this reinforces the notion that models over the WN corpus are more dependent on dependencies while those over the BN corpus are more dependent on n-grams, perhaps because of the longer sentences involved or the more robust reference parses for the written news domain.

More broadly, these results yield a somewhat surprising conclusion: trigram models don’t offer a dramatic advantage over bigram models on the compression task. While sta-

WN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		path	DP-2gr	75.36	63.40	52.15	42.97	-
ILP-3gr	75.07		63.65	52.75	43.88	-	63.11	59.45
path + tree	ILP-2gr-dep	76.15	64.47	53.63	45.15	69.52	67.35	61.82
	ILP-3gr-dep	76.04	64.99	54.47	46.10	69.66	67.82	62.44

Table 3.14: Performance variation with n-gram size for the WN corpus with all systems restricted to compress to the size of the reference compression. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

tistically significant gains are achieved in some measures, unigram and RASP F₁ for the joint models remain similar in both corpora. We conjecture that this is largely due to the relatively small size of the compression datasets under study which leads to sparsity in features for high-order variables and consequently encourages the use of lower-capacity models that avoid overfitting on the training data. Larger compression datasets such as the one proposed by Filippova and Altun (2013) would likely show stronger gains for higher-order models, perhaps making them a worthwhile consideration for applications in environments when output quality takes precedence over computational efficiency and runtime.

3.6.6 Subtree deletion

Finally, we consider the subtree-deletion model that has previously been used in tasks involving joint sentence compression and summarization (Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Qian and Liu, 2013). As indicated in §3.1.1, the assumptions made by this model are not supported by the compression datasets but we seek to examine the impact of unreachable output structures on measures of compression performance. For this purpose, we introduce two additional models:

- **ILP-sub:** A variant of the ILP described in §3.3.1 for dependency edges in which all dependencies in the compressed sentence are drawn only from the set of dependency edges in the input sentence. In other words, we replace \mathcal{Z} in §3.2.2 with $\mathcal{Z}' \triangleq \{\langle t_i, t_j \rangle :$

BN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		tree	ILP-sub	79.39	65.90	52.02	43.28	72.06
ILP-dep	82.70		70.05	56.81	47.94	75.76	70.88	65.25
path + tree	ILP-2gr-sub	79.20	67.58	54.76	46.39	72.08	69.42	63.60
	ILP-2gr-dep	82.85	72.66	60.87	52.87	75.73	73.82	68.56

Table 3.15: Evaluation against subtree deletion over the BN corpus with all systems restricted to compress to the size of the median reference compression. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

$t_j \in \mathcal{T}$ is a dependent of $t_i \in \mathcal{T} \cup \{\text{ROOT}\}$ in S and proceed as before.

- **ILP-2gr-sub:** The corresponding joint inference approach which combines the subtree-deletion variant of §3.3.1 with §3.3.2 to yield a bigram-factored token ordering for the output sentence.

The experimental results for the BN corpus and the WN corpus are presented in Tables 3.15 and 3.16 respectively. In both sets of results, the subtree deletion assumption leads to a dramatic decrease in performance across all evaluation measures. Most relevant among these is dependency F₁ for the output tree, which decreases by 2.7% for the BN corpus and 4.8% for the WN corpus. Similar differences are seen in dependency F₁ after parsing system-generated compressions with the Stanford parser and the results correlate with those using the RASP parser.

The losses incurred by the joint model **ILP-2gr-sub** are similar to those of the standalone subtree-deletion model **ILP-sub** and reinforce the unsuitability of the subtree-deletion assumption for compression tasks. Comparing the joint model to **DP-2gr** from previous evaluations, this demonstrates that combining a subtree-deletion compression technique with a bigram-based approach is significantly worse than just using the bigram-based approach by itself.³⁹ The poor performance of this approach might be consigned to parse errors by

³⁹Naturally, this conclusion is conditioned on the assumption that our dependency features are appropriate for the subtree-deletion task. For instance, different parsers with different dependency label lexicons might

WN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		tree	ILP-sub	72.17	56.36	43.29	34.13	64.23
ILP-dep	75.74		60.07	46.84	37.48	69.08	64.33	58.49
path + tree	ILP-2gr-sub	72.27	59.95	48.72	39.85	64.77	61.78	57.62
	ILP-2gr-dep	76.15	64.47	53.63	45.15	69.52	67.35	61.82

Table 3.16: Evaluation against subtree deletion over the WN corpus with all systems restricted to compress to the size of the reference compression. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

the Stanford parser; however, Qian and Liu (2014) has also examined the subtree-deletion assumption over the BN and WN datasets with the MST parser (McDonald et al., 2005b) and similarly do not find support.

3.7 Remarks

The key contribution of this work is an ILP-based inference approach for sentence compression which jointly considers two structured output spaces—an n-gram factorization and a corresponding dependency tree. This approach unifies many prior techniques in the literature that adopt either bigram, trigram or subtree-deletion formulations of the compression objective. Our flow-based approach imposes no hard limit on factorization and therefore supports higher order n-grams as well as dependencies, while the use of a general-purpose linear programming solver for inference accommodates linguistically-motivated rules and templates as needed for practical applications.

In our evaluation, we find that the joint perspective which pairs syntax and dependency structure offers statistically significant gains over either individual structural perspective. These improvements persist as the compression rate of output sentences is varied although the systems studied converge in performance at very high compression rates. In examining the output, we observe that the output sentences of the joint system avoids the pitfalls of

yield improved results.

bigram-only and dependency-only compression techniques and is often clearly more readable and accurate.

The proposed inference technique enables n-grams of any size to be used in compression problems. In our evaluation, we find that further gains are possible when using trigrams instead of bigrams at the cost of memory and runtime for storing and solving larger ILPs. Here as well, including inference over compressed edge-factored dependency structures offers significant advantages over a trigram-only model although solving these larger ILPs entails a greater computational overhead. Higher-order dependency structures are also feasible in this model but we did not study them with ILP inference; instead, second-order dependencies are investigated in a dynamic program for the joint compression formulation presented in Chapter 5.

The flexibility of linear programs offers many opportunities for building on this work. It is straightforward to extend this inference approach to incorporate paraphrasing and token reordering—we do not evaluate the former in this work but the latter permits us to extend our approach to multiple input sentences (i.e., sentences fusion) in Chapter 7. In addition, Lemma 3.3.1 indicates that other forms of acyclic graph structures can be inferred in a generalization of this formulation; we address this further in Chapter 6 where we consider the inference of structures for high-level relations which specify shallow semantics. Finally, this technique remains compatible with many techniques for document-level abstractive summarization, which remains one of our goals for future research.

Chapter 4

Approximation Strategies for Compression

We have thus far discussed a flexible ILP formulation for text-to-text generation which admits a number of useful extensions, some of which are explored in later chapters. However, it is well-established that the utility of ILP for optimal inference in structured problems is often outweighed by the worst-case performance of ILP solvers on large problems without unique integral solutions. Furthermore, approximate solutions can often be adequate for real-world generation systems, particularly in the presence of restrictions on output text, e.g., linguistically-motivated constraints such as those described by Clarke and Lapata (2008) or pruning strategies such as the use of domain-specific sentence templates.

In this chapter, we develop approximate inference strategies to the joint approach from Chapter 3 which trade the optimality guarantees of exact ILP for faster inference by separately solving the n-gram and dependency subproblems and using Lagrange multipliers to enforce consistency between their solutions.¹ However, while the n-gram problem—restricted to order-preserving bigrams—can be solved efficiently using the dynamic programming approach of McDonald (2006), there are no efficient algorithms to recover maximum weighted non-projective subtrees in a general directed graph.² Maximum spanning tree al-

¹An earlier version of this research was presented in Thadani (2014).

²In contrast to order-preserving projective trees, for which efficient algorithms are provided in Chapter 5.

gorithms, commonly used in non-projective dependency parsing (McDonald et al., 2005b), are not easily adaptable to this task since the maximum-weight subtree is not necessarily a part of the maximum spanning tree.

We therefore consider methods to recover approximate solutions for the subproblem of finding the maximum weighted subtree in a graph, common among which is the use of a linear programming relaxation. This linear program (LP) appears empirically tight for compression problems and our experiments indicate that simply using the non-integral solutions of this LP in Lagrangian relaxation can empirically lead to reasonable compressions. In addition, we can recover approximate solutions to this problem by using the Chu-Liu Edmonds algorithm for recovering maximum spanning trees (Chu and Liu, 1965; Edmonds, 1967) over the relatively sparse subgraph defined by a solution to the relaxed LP. Our proposed approximation strategies are evaluated using automated metrics in order to address the question: under what conditions should a real-world sentence compression system implementation consider exact inference with an ILP or approximate inference? The contributions of this chapter include:

- An empirically useful technique for approximating the maximum-weight subtree in a weighted graph using LP-relaxed inference.
- Multiple approaches to generate good approximate solutions for multi-structured compression, based on Lagrangian relaxation to enforce equality between the path and tree inference subproblems.
- An analysis of the performance tradeoffs incurred by approximate and exact approaches with respect to runtime as well as performance under automated evaluation measures.

4.1 Compression via Lagrangian Relaxation

Dual decomposition (Komodakis et al., 2007) and Lagrangian relaxation in general are often used for solving joint inference problems which are decomposable into individual subproblems linked by equality constraints (Koo et al., 2010; Rush et al., 2010; Rush and Collins, 2011; DeNero and Macherey, 2011; Martins et al., 2011; Das et al., 2012;

Almeida and Martins, 2013). This approach permits sub-problems to be solved separately using problem-specific efficient algorithms while consistency over the solutions produced is enforced through Lagrange multipliers via iterative optimization. Exact solutions are guaranteed when the algorithm converges on a consistent primal solution, although this convergence itself is not guaranteed and depends on the tightness of the underlying LP relaxation (cf. §3.3). The primary advantage of this technique is the ability to leverage the underlying structure of the problems in inference rather than relying on a generic ILP formulation while still often producing exact solutions.

The multi-structured inference problem described in the previous section seems in many ways to be a natural fit to such an approach because output scores factor over different types of structure that comprise the output compression. Even if ILP-based approaches perform reasonably at the scale of single-sentence compression problems, the exponential worst-case complexity of general-purpose ILPs will inevitably pose challenges when scaling up to (a) handle larger inputs, (b) use higher-order structural fragments, or (c) incorporate additional models. In the following section, we propose an alternative formulation that exploits the modularity of the multi-structured objective.

4.1.1 Decomposing the inference task

We begin by revisiting the optimization problem characterized by (3.3) in Chapter 3. Given an input sentence S with n tokens, we seek to recover the highest-scoring compression \hat{C} which maximizes a multi-structured linear objective

$$\hat{C} = \arg \max_{\mathbf{x}, \mathbf{y}, \mathbf{z}} \mathbf{x}^\top \Delta_{\text{tok}} + \mathbf{y}^\top \Delta_{\text{ngr}} + \mathbf{z}^\top \Delta_{\text{dep}}$$

where \mathbf{x} , \mathbf{y} and \mathbf{z} refer to incidence vectors respectively representing a token configuration, an n-gram factorization and a dependency tree while Δ_{tok} , Δ_{ngr} and Δ_{dep} respectively denote vectors of feature-based scores for each corresponding substructure.

The two structural problems that need to be solved in this formulation are the extraction of a maximum-weight n-gram factorization \mathbf{y} of an acyclic path from a lattice of potential adjacency relationships in S and the recovery of a maximum-weight directed subtree \mathbf{z} . Let $\alpha(\mathbf{y}) \in \{0, 1\}^n$ denote the incidence vector of tokens contained in the n-gram path \mathbf{y} and

$\beta(\mathbf{z}) \in \{0, 1\}^n$ denote the incidence vector of words contained in the dependency tree \mathbf{z} . We can now rewrite the objective from (3.3) while enforcing the constraint that the words contained in the path \mathbf{y} are the same as the words contained in the tree \mathbf{z} , i.e., $\alpha(\mathbf{y}) = \beta(\mathbf{z})$, by introducing a vector of Lagrange multipliers $\lambda \in \mathbb{R}^n$. In addition, the token configuration \mathbf{x} can be rewritten in the form of a weighted combination of $\alpha(\mathbf{y})$ and $\beta(\mathbf{z})$ to ensure its consistency with \mathbf{y} and \mathbf{z} . This results in the following Lagrangian:

$$\begin{aligned} L(\lambda, \mathbf{y}, \mathbf{z}) &= \mathbf{y}^\top \Delta_{\text{ngr}} + \mathbf{z}^\top \Delta_{\text{dep}} \\ &\quad + \Delta_{\text{tok}}^\top (\psi \cdot \alpha(\mathbf{y}) + (1 - \psi) \cdot \beta(\mathbf{z})) \\ &\quad + \lambda^\top (\alpha(\mathbf{y}) - \beta(\mathbf{z})) \end{aligned} \tag{4.1}$$

Finding the \mathbf{y} and \mathbf{z} that maximize this Lagrangian above yields a dual objective, and the dual problem corresponding to the primal objective specified in (3.3) is therefore the minimization of this objective over the Lagrange multipliers λ .

$$\begin{aligned} &\min_{\lambda} \max_{\mathbf{y}, \mathbf{z}} L(\lambda, \mathbf{y}, \mathbf{z}) \\ &= \min_{\lambda} \max_{\mathbf{y}} \mathbf{y}^\top \Delta_{\text{ngr}} + (\lambda + \psi \cdot \Delta_{\text{tok}})^\top \alpha(\mathbf{y}) \\ &\quad + \max_{\mathbf{z}} \mathbf{z}^\top \Delta_{\text{dep}} - (\lambda + (\psi - 1) \cdot \Delta_{\text{tok}})^\top \beta(\mathbf{z}) \\ &= \min_{\lambda} \max_{\mathbf{y}} f(\mathbf{y}, \Delta, \lambda, \psi) \\ &\quad + \max_{\mathbf{z}} g(\mathbf{z}, \Delta, \lambda, \psi) \end{aligned} \tag{4.2}$$

This can now be solved with the iterative subgradient algorithm illustrated in Algorithm 2. In each iteration i , the algorithm solves for $\mathbf{y}^{(i)}$ and $\mathbf{z}^{(i)}$ under $\lambda^{(i)}$, then generates $\lambda^{(i+1)}$ to penalize inconsistencies between $\alpha(\mathbf{y}^{(i)})$ and $\beta(\mathbf{z}^{(i)})$. When $\alpha(\mathbf{y}^{(i)}) = \beta(\mathbf{z}^{(i)})$, the resulting primal solution is exact, i.e., $\mathbf{y}^{(i)}$ and $\mathbf{z}^{(i)}$ represent the optimal structures under (3.3).³ Otherwise, if the algorithm starts oscillating between a few primal solutions, the underlying LP must have a non-integral solution. In this case, we opt to identify the solution from the set of oscillating solutions that maximizes the joint score from (3.3) (cf. §4.1.4), although a variety of other techniques can also be employed to recover optimal or near-

³A proof is available in Rush and Collins (2011).

Algorithm 2 Subgradient-based joint inference

Input: scores Δ , hyperparameter ψ , iteration limit i_{\max} , repetition limit J_{\max} ,
learning rate schedule $\eta \in \mathbb{R}^{i_{\max}}$

Output: token configuration \mathbf{x}

```

1: initialize solutions  $\mathcal{J} \leftarrow \emptyset$ , repeated solutions  $\mathcal{J}_{\text{rep}} \leftarrow \emptyset$ 
2:  $\lambda^{(0)} \leftarrow \mathbf{0}^n$ 
3: for iteration  $i$  in  $0 \dots i_{\max}$  do
4:    $\hat{\mathbf{y}} \leftarrow \arg \max_{\mathbf{y}} f(\mathbf{y}, \Delta, \lambda^{(i)}, \psi)$ 
5:    $\hat{\mathbf{z}} \leftarrow \arg \max_{\mathbf{z}} g(\mathbf{z}, \Delta, \lambda^{(i)}, \psi)$ 
6:   if  $\alpha(\hat{\mathbf{y}}) = \beta(\hat{\mathbf{z}})$  then return  $\alpha(\hat{\mathbf{y}})$  // optimal solution
7:   if  $\alpha(\hat{\mathbf{y}}) \in \mathcal{J}$  then  $\mathcal{J}_{\text{rep}} \leftarrow \mathcal{J}_{\text{rep}} \cup \{\alpha(\hat{\mathbf{y}})\}$ 
8:   if  $\beta(\hat{\mathbf{z}}) \in \mathcal{J}$  then  $\mathcal{J}_{\text{rep}} \leftarrow \mathcal{J}_{\text{rep}} \cup \{\beta(\hat{\mathbf{z}})\}$ 
9:   if  $|\mathcal{J}_{\text{rep}}| \geq J_{\max}$  then break // oscillating solutions
10:   $\mathcal{J} \leftarrow \mathcal{J} \cup \{\alpha(\hat{\mathbf{y}}), \beta(\hat{\mathbf{z}})\}$ 
11:   $\lambda^{(i+1)} \leftarrow \lambda^{(i)} - \eta_i (\alpha(\hat{\mathbf{y}}) - \beta(\hat{\mathbf{z}}))$ 
12: if  $|\mathcal{J}_{\text{rep}}| = 0$  then  $\mathcal{J}_{\text{rep}} \leftarrow \mathcal{J}$ 
return  $\arg \max_{\mathbf{x} \in \mathcal{J}_{\text{rep}}} \mathbf{x}^\top \Delta_{\text{tok}} + \alpha^{-1}(\mathbf{x})^\top \Delta_{\text{ngr}} + \beta^{-1}(\mathbf{x})^\top \Delta_{\text{dep}}$  // approximation

```

optimal solutions.⁴ The application of this Lagrangian relaxation strategy is contingent upon the existence of algorithms to solve the maximization subproblems for $f(\mathbf{y}, \Delta, \lambda, \psi)$ and $g(\mathbf{z}, \Delta, \lambda, \psi)$. The following sections discuss our approach to these problems.

4.1.2 Bigram paths

While the ILP approach presented in Chapter 3 permitted the recovery of n-grams of any order, we observe that the use of trigrams does not offer significant advantages over bigrams. For this reason, we confine ourselves to the use of bigrams in \mathbf{y} in this chapter as well as the remainder of this dissertation. Similarly, although the ILP permits the reordering of

⁴Heuristic approaches (Komodakis et al., 2007; Rush et al., 2010), tightening (Rush and Collins, 2011) or branch and bound (Das et al., 2012) can still be used to retrieve optimal solutions, but we did not explore these strategies here.

input tokens, the compression dataset described in §3.1 does not involve reordering. If we were to assume a total ordering over output tokens, we can employ an efficient approach to finding optimal bigram paths.

McDonald (2006) provides a Viterbi-like dynamic programming algorithm to recover the highest-scoring path of order-preserving bigrams from a lattice, either in unconstrained form or with a specific length constraint. The latter requires a dynamic programming table $Q[i][r]$ which represents the best score for a compression of length r ending at token i . The table can be populated using the following recurrence:

$$\begin{aligned} Q[i][1] &= \Delta'_{\text{ngr}}(S, \text{START}, i) \\ Q[i][r] &= \max_{j < i} Q[j][r-1] + \Delta'_{\text{ngr}}(S, i, j) \\ Q[i][m+1] &= Q[i][m] + \Delta'_{\text{ngr}}(S, i, \text{END}) \end{aligned}$$

where $m \triangleq \lfloor n \cdot \omega \rfloor$ is the number of output tokens required to satisfy a compression rate ω and the scoring function Δ'_{ngr} is defined as

$$\Delta'_{\text{ngr}}(S, i, j) \triangleq \Delta_{\text{ngr}}(S, i, j) + \lambda_j + \psi \cdot \Delta_{\text{tok}}(S, j) \quad (4.3)$$

so as to solve $f(\mathbf{y}, \mathbf{\Delta}, \boldsymbol{\lambda}, \psi)$ from (4.2). This approach requires $O(n^2m)$ time and $O(nm)$ space in order to identify the highest scoring path \mathbf{y} and corresponding token configuration $\boldsymbol{\alpha}(\mathbf{y})$. When no specific compression rate is imposed on the output sentence, the table Q can be reduced to a vector whose entries $Q[i]$ represent the best score for a compression of any length ending with token i , reducing runtime to $O(n^2)$ and space to $O(n)$.

4.1.3 Dependency subtrees

The maximum-weight non-projective subtree problem over general graphs is not as easily solved. Although the maximum *spanning* tree in any directed graph can be recovered efficiently, Figure 4.1 illustrates that the maximum-scoring subtree is not necessarily found within it. The problem of recovering a length-constrained maximum-weight subtree in a graph has been shown to be NP-hard even with undirected graphs (Lau et al., 2006).

In order to produce a solution to this subproblem, we consider an LP relaxation of the ILP from §3.3.1 by omitting integer constraints over the token and dependency variables in

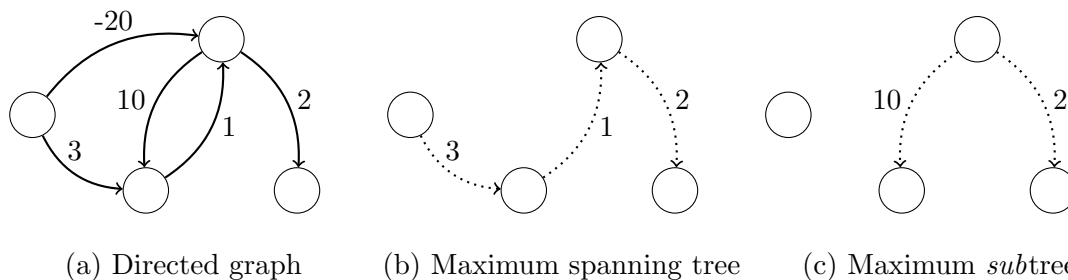


Figure 4.1: An example of a weighted directed graph (a) whose maximum spanning tree (b) does not contain the maximum-weight subtree (c). Missing edges in (a) have weight $-\infty$.

\mathbf{x} and \mathbf{z} respectively. The objective for this LP is given by

$$\max_{\mathbf{x}, \mathbf{z}} \mathbf{x}^\top \Delta'_{\text{tok}} + \mathbf{z}^\top \Delta_{\text{dep}} \quad (4.4)$$

where the vector of token scores is redefined as

$$\Delta'_{\text{tok}} \triangleq (1 - \psi) \cdot \Delta_{\text{tok}} - \lambda \quad (4.5)$$

in order to solve $g(\mathbf{z}, \Delta, \lambda, \psi)$ from (4.2). In the LP relaxation, x_i and z_{ij} are redefined as real-valued variables whose values lie in $[0, 1]$, thereby potentially accommodating fractional values for dependency and token indicators. As a result, the flow network is able to establish connectivity but cannot guarantee a tree structure in the output. For instance, directed acyclic structures are possible and token indicators x_i may be partially assigned to the solution structure. This poses a challenge in implementing $\beta(\mathbf{z})$ which is needed to recover a token configuration from the solution of this subproblem.

We propose two alternative solutions to address this issue in the context of the joint inference strategy. The first is to simply use the fractional token configuration identified by the LP in Algorithm 2, i.e., to set $\beta(\tilde{\mathbf{z}}) = \tilde{\mathbf{x}}$ where $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{z}}$ represent the real-valued counterparts of the incidence vectors \mathbf{x} and \mathbf{z} . As the bigram subproblem is guaranteed to return a well-formed integral solution which obeys the imposed compression rate, we are assured of a source of valid—if suboptimal—solutions in line 10 of Algorithm 2.

We also consider another strategy that attempts to approximate a valid integral solution to the dependency subproblem. In order to do this, we first include an additional constraint in the relaxed LP which restricts the number of tokens in the output to $m \triangleq \lfloor n \cdot \omega \rfloor$

where ω is the required compression rate. This serves to ensure that the resulting token configuration $\tilde{\mathbf{x}}$ has more than m non-zero components, i.e., there are at least as many tokens activated⁵ in a relaxed solution as are required in a valid compression.⁶ We then construct a subgraph $G(\tilde{\mathbf{z}})$ consisting of all dependency edges that were assigned non-zero values in the solution, assigning to each edge a score equal to the score of that edge in the LP as well as the score of its dependent word, i.e., each z_{ij} in $G(\tilde{\mathbf{z}})$ is assigned a score of $\Delta_{\text{dep}}(S, i, j) - \lambda_j + (1 - \psi) \cdot \Delta_{\text{tok}}(S, j)$. Because the flow constraints in (3.6)–(3.7) ensure a connected $\tilde{\mathbf{z}}$, it is therefore possible to recover a maximum-weight spanning tree from $G(\tilde{\mathbf{z}})$ using the Chu-Liu Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967).⁷ Although the runtime of this algorithm is cubic in the size of the input graph, it is fairly speedy when applied on relatively sparse graphs such as $G(\tilde{\mathbf{z}})$.

The resulting spanning tree is a useful integral approximation of $\tilde{\mathbf{z}}$ but, as indicated previously, may contain more nodes than m due to fractional values in $\tilde{\mathbf{x}}$. We therefore prune leaves with the lowest incoming edge weight in the current tree until precisely m nodes remain. The resulting tree is then assumed to be a reasonable approximation of the optimal integral solution to this LP.

4.1.4 Scoring approximate solutions

The Chu-Liu Edmonds algorithm is also employed for another purpose. When the underlying LP for the joint inference problem is not tight—a frequent occurrence in our compression experiments—Algorithm 2 will not converge on a single primal solution and will instead oscillate between primal solutions that are close to the dual optimum. We identify this phenomenon by counting repeated solutions in \mathcal{J}_{rep} and, if they exceed some threshold J_{max} with at least one repeated solution from either subproblem, we terminate the update

⁵In the context of non-binary variables in an LP relaxation, an *activated* variable is one that is assigned a value other than 0.

⁶However, our experiments in §4.2.1 show that this constraint also significantly decreases the proportion of *tight* relaxations, i.e., LPs which produce integral—and therefore optimal—compression solutions.

⁷A detailed description of the Chu-Liu Edmonds algorithm for recovering optimal maximum spanning trees is available in McDonald et al. (2005b).

procedure for Lagrange multipliers and instead attempt to identify a good solution from the repeating ones by scoring them under the joint score from (3.3). This requires the retrieval of bigram and dependency structures for every potential solution, represented by the functions α^{-1} and β^{-1} respectively in the final return statement of Algorithm 2.

Specifically, when considering a dependency solution \mathbf{z} produced by the techniques from §4.1.3, we need to score the token configuration $\mathbf{x} = \beta(\mathbf{z})$ as well as a corresponding bigram path $\alpha^{-1}(\mathbf{x})$. Since we assume that the input ordering of tokens is not altered in the output sentence, the latter term is straightforward to recover from the active tokens in \mathbf{x} .⁸ Similarly, scoring a bigram solution \mathbf{y} produced by the dynamic program from §4.1.2 also requires us to score its token configuration $\mathbf{x} = \alpha(\mathbf{y})$ and a corresponding parse tree $\beta^{-1}(\mathbf{x})$. This can be recovered by constructing a dependency subgraph across only the active tokens in \mathbf{x} and then identifying the maximum spanning tree for this subgraph using the Chu-Liu Edmonds algorithm.

4.2 Experiments

We attempt to characterize the performance and runtime characteristics of compression systems based on approximate solutions to the dependency compression problem through experiments over the BN and WN compression datasets (Clarke and Lapata, 2006b; Clarke and Lapata, 2007) which are described in §3.1. Our experimental environment follows the setup described in §3.6.

4.2.1 Tightness of approximations

We first examine the approximations of the dependency subproblem in isolation before turning to joint models and comparisons with exact systems. Specifically, we consider the following systems:

- **LP-dep**: An LP relaxation of **ILP-dep**, the ILP for inference of optimal edge-factored

⁸Ambiguity in assigning bigrams which involve repeated words in the input is rare. Moreover, since the process of mapping output tokens from reference solutions to input tokens suffers from the same ambiguity, it is of little consequence to the broader learning problem as long as resolution is consistent.

dependency trees described in §3.3.1. While this cannot generate integral compression solutions on its own, it is used in the following systems which can.

- **LP-dep→MST**: The approximate inference approach for the maximum-weight subtree problem presented in §4.1.3. A maximum spanning tree is recovered from the potentially-fractional solutions of **LP-dep** and greedily pruned in order to generate a valid integral solution while observing the imposed compression rate.

When working with these approximate inference approaches, we avoid retraining models and instead reuse the model produced for the evaluation of **ILP-dep** in Chapter 3.⁹ Although reusing models makes comparison between systems more straightforward, it remains possible that new models trained with the same approximate inference strategy used during testing could compensate for search errors to some degree.

To begin, we are naturally interested in the tightness of the LP relaxation **LP-dep** which underpins the approximate inference techniques considered in this chapter. Table 4.1 enumerates the rate of integral solutions from these LPs over the BN and WN development corpora. When no compression rate is imposed, we find that the LPs are largely tight and on average only 1–2% of the non-zero components of \mathbf{x} and \mathbf{z} are fractional. However, when imposing the reference compression rate on the output compressions, the rate of fractional variables increases to 14–16% and overall LP tightness is lowered significantly. We also note an interesting discrepancy between the two corpora: the WN corpus features more tight LPs (92.1%) when the compression rate is absent and less tight LPs (17.5%) when it is imposed as opposed to the BN corpus (78.2% and 39.7% respectively).

As **LP-dep** cannot be used in isolation for compression evaluations, especially without imposing a compression rate, we rely on **LP-dep→MST** for recovering valid compressions from fractional LP solutions. Table 4.2 examines the compression solutions produced by **LP-dep→MST** and compares them to the optimal solutions produced by **ILP-dep** over the development corpora. These results indicate that the MST-based approximation strategy recovers optimal token solutions in over 60% of the instances considered despite starting with fewer integral solutions from the LP relaxation—39.7% of BN instances and 17.5% of

⁹However, since we follow Martins et al. (2009) in employing LP relaxations during training in §3.6, the actual inference approach used to generate the model is equivalent to **LP-dep**.

	Imposed ω	Tight LPs %	Integral tokens %	Integral deps %
BN	-	78.2	99.1	97.8
	reference	39.7	86.7	83.9
WN	-	92.1	99.8	98.8
	reference	17.5	85.1	84.6

Table 4.1: Empirical tightness of LP-**dep** and proportion of integral variables among all non-zero variables in each instance averaged over the BN and WN development corpora.

	Optimal \mathbf{x} %	Optimal \mathbf{z} %	Correct tokens %	Correct deps %
BN	64.1	62.8	94.0	89.6
WN	60.3	49.2	95.7	93.7

Table 4.2: Optimality of output compressions from LP-**dep**→MST and proportion of correct non-zero variables when compared to ILP-**dep** for each instance, averaged over the BN and WN development corpora.

WN instances. However, the lower empirical tightness of the WN dataset manifests in a lower rate of optimal compressed dependency trees—an absolute decrease of 13% from the BN dataset. This difference between the two corpora also appears to influence compression quality in the evaluations covered in 4.2.3.

Furthermore, although about 86% of the non-zero token variables are integral on average in Table 4.1 when a compression rate is imposed, the MST-pruning approach correctly recovers about 94–95% of the output tokens. The same is true for dependencies: about 84% of non-zero dependency variables are integral in Table 4.1 but LP-**dep**→MST correctly identifies 90–94% of the output dependencies. The high rate of component recovery but relatively low rate of optimal solutions paints a picture of a reasonable though imperfect approximation to dependency-based compressions, prompting the question of whether performance can be improved by pairing it with an exact bigram-based compression technique via Lagrangian relaxation as described in §4.1.1. We now turn to evaluating these joint approaches, starting with a consideration of the tradeoff between bigram and dependency solutions when producing token solutions.

4.2.2 Tradeoff between structural solutions

Lagrangian relaxation allows us to pair the approximation techniques for edge-factored dependency compression with the dynamic program for bigram-factored compressions described in §4.1.2, yielding the following multi-structured inference techniques:

- **DP-2gr+LP-dep→MST**: An approximate joint inference approach based on dual decomposition that uses **DP-2gr** for the maximum weight path problem and pairs it with **LP-dep→MST** for the maximum weight subtree problem.
- **DP-2gr+LP-dep**: Another dual decomposition approach that pairs **DP-2gr** directly with potentially-fractional solutions from **LP-dep**. When dependency solutions are fractional, the MST algorithm can simply be run over token solutions from the bigram solution in order to recover the optimal accompanying dependency tree for evaluation.

As with the approximate dependency-based compression techniques, both these approaches use the model trained for **ILP-2gr-dep** in §3.6. The learning rate schedule for the subgradient descent approach in Algorithm 2 is obtained by $\eta_i \triangleq \tau/(\tau + i)$ for each iteration i with the hyperparameter τ set to 100 for aggressive subgradient updates. The repetition limit j_{\max} is set to 20 after which we assume non-convergent oscillation between solutions. We set the iteration limit i_{\max} to 30 since most solutions are observed to converge or oscillate within 20 iterations. These hyperparameters were not tuned extensively and we acknowledge that better results may be achieved here with additional attention on the dual decomposition formulation.

The decomposition of the multi-structured objective function described in §4.1.1 separates the bigram and dependency-factored subproblems while distributing token scores Δ_{tok} over bigram variables and dependency variables. As one of the subproblems is not exact, we include a hyperparameter ψ to trade off the influence of each structural solution on the token configuration \mathbf{x} of the output compression. Here, we examine the effect of ψ on output compressions over the development partitions of the BN and WN corpora.

Figure 4.2 shows the variation in RASP F_1 over the BN and WN development datasets as ψ is varied from 0—meaning the token solution \mathbf{x} is produced entirely from the dependency solution \mathbf{z} —to 1—implying \mathbf{x} is drawn from the bigram solution \mathbf{y} . Although the

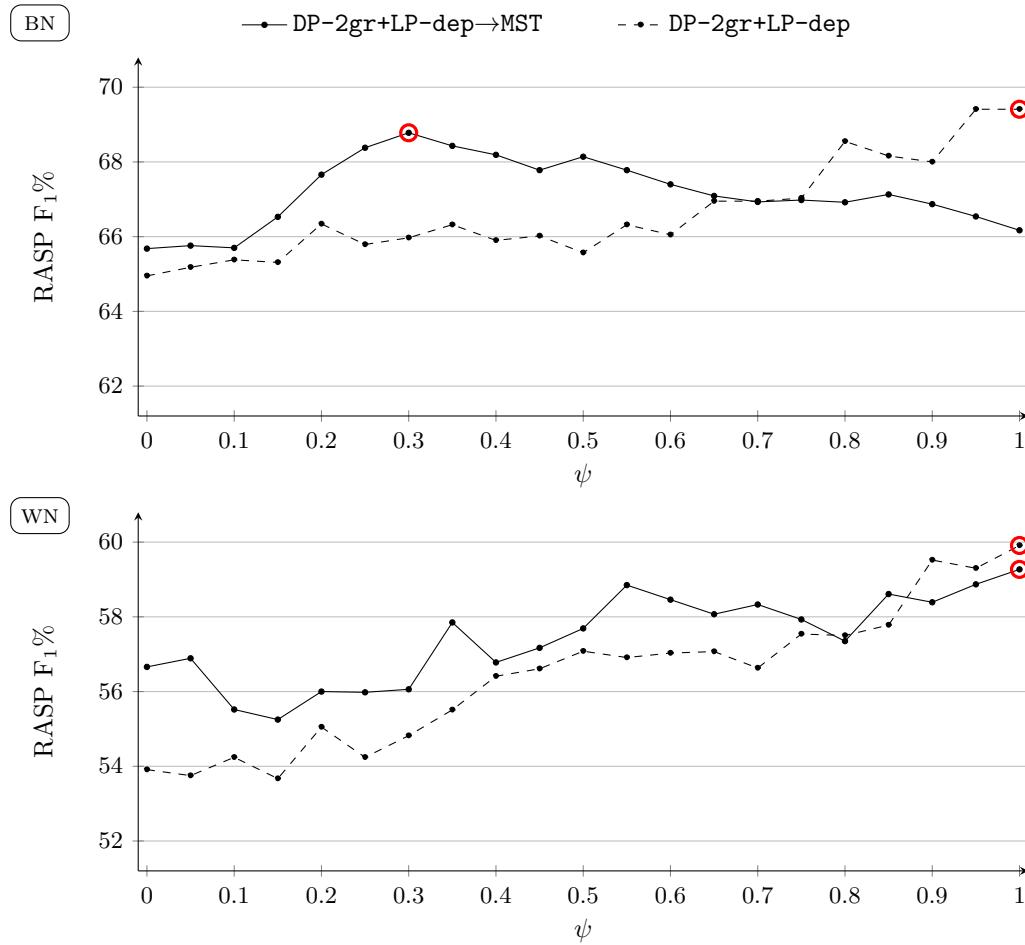


Figure 4.2: Variation in RASP F_1 with ψ on the BN and WN development datasets. As defined in (4.1), high values of ψ amplify the influence of the exact bigram solution \mathbf{y} on the token solution \mathbf{x} while low values of ψ attenuate it in favor of the approximate dependency solution \mathbf{z} . Highlighted datapoints indicate maxima and selected ψ values for later experiments.

RASP measure appears less sensitive to ψ over the BN corpus than the WN corpus,¹⁰ we note some common trends. Low values of ψ lead to diminished performance for all systems, likely reflecting the impact of approximation errors within the dependency-based subproblem. This is echoed in the observation that performance increases predictably with ψ for DP-2gr+LP-dep and peaks at $\psi = 1$ when the LP-relaxed tree solutions have no influence on the token solution.

¹⁰We attribute the higher volatility observed over the WN dataset to fact that BN corpus results are averaged over 3 reference compressions.

Performance for `DP-2gr+LP-dep`→`MST` shows a marked difference between the two datasets. RASP F_1 peaks around $\psi = 0.3$ on the BN dataset, possibly implying that the feasible tree approximations produced by the MST pruning strategy are sufficiently accurate so as to balance the exact solutions produced by `DP-2gr`. For the WN dataset, the best performance is again achieved at $\psi = 1$, perhaps because the approximation strategy produces fewer optimal tree solutions as seen in Table 4.2. With their best hyper values, the two systems produce similar RASP F_1 numbers on both datasets with `DP-2gr+LP-dep` outperforming `DP-2gr+LP-dep`→`MST` by one percentage point.

In all cases above, we set ψ to values that produce the best RASP F_1 for their respective system and corpus in the remaining experiments. Specifically, `DP-2gr+LP-dep` uses $\psi = 1$ for both corpora and `DP-2gr+LP-dep`→`MST` uses $\psi_{\text{BN}} = 0.3$ and $\psi_{\text{WN}} = 1$.

4.2.3 Compression quality

In this section, we compare the approximate inference techniques for dependency-based and multi-structured compression against the following systems from 3.6.1:

- `LM-3gr`: A reimplementaion of the mostly unsupervised ILP of Clarke and Lapata (2008) which infers order-preserving trigrams parameterized with log-likelihood under an LM and a token significance score.
- `DP-2gr`: The dynamic program of McDonald (2006) described in §4.1.2 for inference of bigram-factored compressions under a compression rate.
- `ILP-dep`: The ILP described in §3.3.1 for inference of edge-factored compressed dependency trees along with constraints from §3.3.3 for imposing a compression rate.
- `ILP-2gr-dep`: The joint inference approach which combines the constraints from §3.3.1 to produce an edge-factored dependency tree with §3.3.2 to yield a bigram-factored token ordering for the output sentence.

We do not include `LP-dep` in the roster of systems evaluated under measures of compression quality since feasible output sentences cannot be recovered directly from its fractional solutions. The remaining approximate inference approaches use the models generated by

BN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		path	LM-3gr (CL08)	74.96	60.60	46.83	38.71	-
DP-2gr (McD06)	82.94		72.84	61.08	52.65	-	70.96	66.34
tree	LP-dep→MST	81.81	67.74	53.95	45.05	73.22	68.02	62.91
	ILP-dep	82.70	70.05	56.81	47.94	75.76	70.88	65.25
path + tree	DP-2gr+LP-dep→MST	82.47	71.00	58.53	50.45	75.23	71.93	67.42
	DP-2gr+LP-dep	82.35	72.33	60.61	52.57	73.35	72.14	67.94
	ILP-2gr-dep	82.85	72.66	60.87	52.87	75.73	73.82	68.56

Table 4.3: Experimental results for the BN corpus averaged over 3 reference compressions per instance. All systems were restricted to compress to the size of the median reference compression yielding an average compression rate of 77.26%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

the equivalent exact inference approach from the list above.¹¹

Table 4.3 contains the results over the BN corpus. We see that approximating dependency-based compression with LP-dep→MST results in a significant 2–3% absolute decrease in n-gram F₁ (excluding unigrams) as well as dependency F₁ for both generated trees and output sentence parses with respect to the optimal ILP-dep solutions. This gap is narrowed slightly by the influence of exact bigram-based compression on the equivalent joint approach DP-2gr+LP-dep→MST when compared to ILP-2gr-dep. In particular, F₁ over the dependency edges from the output tree **z** is especially improved and approaches—though remains significantly lower than—that of the exact ILP-2gr-dep approach.

Dropping the greedy MST pruning following the LP relaxation in DP-2gr+LP-dep results in a system which more closely follows DP-2gr with no statistical difference in n-gram F₁ between these two systems as well as ILP-2gr-dep. In terms of F₁ measures over trees produced by parsing output sentences, both approximate joint inference methods improve significantly over DP-2gr although they remain significantly lower than ILP-2gr-dep when the Stanford parser is used to identify syntactic relations.

¹¹Albeit with LP-relaxed inference for training as in §3.6.

WN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		path	LM-3gr (CL08)	66.68	51.59	39.33	30.54	-
DP-2gr (McD06)	75.36		63.40	52.15	42.97	-	63.08	59.43
tree	LP-dep→MST	75.05	57.70	43.75	34.35	67.39	61.41	56.39
	ILP-dep	75.74	60.07	46.84	37.48	69.08	64.33	58.49
path + tree	DP-2gr+LP-dep→MST	75.03	61.33	49.32	40.32	68.09	64.35	59.19
	DP-2gr+LP-dep	74.95	63.44	52.66	44.09	66.69	64.89	60.40
	ILP-2gr-dep	76.15	64.47	53.63	45.15	69.52	67.35	61.82

Table 4.4: Experimental results for the WN corpus with all systems compressing to the size of the reference compression, yielding an average compression rate of 70.24%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

Table 4.4 contains the results over the WN corpus, which appear largely consistent with those over the BN corpus. LP-dep→MST yields compressions that score 2–3% lower than ILP-dep on all F₁ measures, performance improves when pairing this with the exact bigram compression approach through dual decomposition in DP-2gr+LP-dep→MST and, furthermore, avoiding the greedy MST pruning strategy for dependency trees in DP-2gr+LP-dep yields a system that produces results similar to DP-2gr under n-gram F₁ but with significant gains in dependency F₁ that nevertheless do not match the performance of the exact joint approach ILP-2gr-dep. We attribute the weaker performance of approximate inference over the WN dataset to two factors: the greater reliance on dependencies for this corpus that was noted in §3.6.5 and the relatively poor approximation tightness observed for this corpus in Table 4.2.1.

In summary, while compressions drawn from approximate inference appear clearly inferior to those from exact inference, the performance gap remains relatively small for DP-2gr+LP-dep. The advantage of these approximate inference strategies, however, is in improving practical runtime for compression tasks when ILP is not usable. We examine this further in the following section.

Structures	Inference	Average time (sec)		Median time (sec)	
		BN	WN	BN	WN
path	LM-3gr (CL08)	0.86	1.95	0.16	0.73
	DP-2gr (McD06)	0.01	0.01	0.01	0.01
tree	LP-dep→MST	0.07	0.13	0.03	0.06
	ILP-dep	0.18	0.30	0.04	0.11
path + tree	DP-2gr+LP-dep→MST	0.24	0.50	0.08	0.22
	DP-2gr+LP-dep	0.28	0.70	0.08	0.27
	ILP-2gr-dep	0.48	0.93	0.10	0.26

Table 4.5: Time in seconds for inference over the BN and WN test datasets, excluding the time required for initializing and scoring features.

4.2.4 Timing

In order to obtain comparable running times for each of the systems studied, we ran experiments for each corpus on a single machine. The specifications for these machines include 24GB of memory and two quad-core Intel® Xeon® X5550 CPUs clocked at 2.67GHz with hyper-threading enabled, exposing 16 processors in total.¹² Timing results were observed to be largely consistent on repeated runs.

The Gurobi ILP solver that we use in these experiments is explicitly designed to exploit multi-core CPUs for speeding up the recovery of ILP solutions and the verification of their optimality. However, our implementation of subgradient-based dual decomposition is single-processed—except for LP relaxation components which also rely on Gurobi—without significant effort toward runtime optimization. A fair timing evaluation would thus consist of restricting Gurobi to use only a single processor in execution rather than the available 16, as is standard in other timing experiments on compression (Qian and Liu, 2014). However, our chief interest in this evaluation is in characterizing *real-world* performance: multi-core systems are widespread in both industrial and personal computing settings and it stands to reason that practical implementations of compression ILPs would leverage all available

¹²Hyper-threading permits two threads to execute concurrently on a single CPU core.

Structures	Inference	Convergence %		Num. iterations	
		BN	WN	BN	WN
path + tree	DP-2gr+LP-dep→MST	27.97	21.23	2.11	5.98
	DP-2gr+LP-dep	24.26	12.94	1.65	4.59

Table 4.6: Fraction of instances which converge under dual decomposition and the average number of iterations to convergence for them in the BN and WN test datasets.

resources for faster solutions. We therefore opt to let Gurobi use all 16 processors in these timing experiments while acknowledging that this obscures the potential runtime benefits of our approximate alternatives in resource-constrained environments.

Table 4.5 summarizes the timing results on the BN and WN test corpora respectively for the systems evaluated in the previous section. Starting with the n-gram approaches, it is not surprising to observe that the high-order LM-3gr ILP is the slowest among the systems tested and the DP-2gr dynamic program is the speediest. In particular, DP-2gr appears to be a particularly appropriate choice for real-time compression tasks despite its mediocre performance on quality metrics. A further evaluation with dynamic programming variants of multi-structured inference is presented in Chapter 5.

Turning to the dependency-based techniques, although the asymptotic ILP size of the dependency-only ILP-dep is similar to the multi-structured ILP-2gr-dep, we observe that the former takes a third of the time required by the latter for inference across both corpora. The proposed approximation technique LP-dep→MST further halves the average runtime of this approach albeit with a significant drop in compression quality as shown in the previous section. Using this approximation with dual decomposition in DP-2gr+LP-dep→MST improves significantly over ILP-dep but the iterative dual decomposition procedure naturally increases inference time.

We record largely similar runtime performance for the two multi-structured approximation techniques DP-2gr+LP-dep→MST and DP-2gr+LP-dep with both approaches reducing average inference time by 30–50% when compared to the equivalent exact inference approach ILP-2gr-dep. These relatively modest runtime gains may be attributed to the power of a highly-optimized multi-core solver when compared to our relatively naïve Python imple-

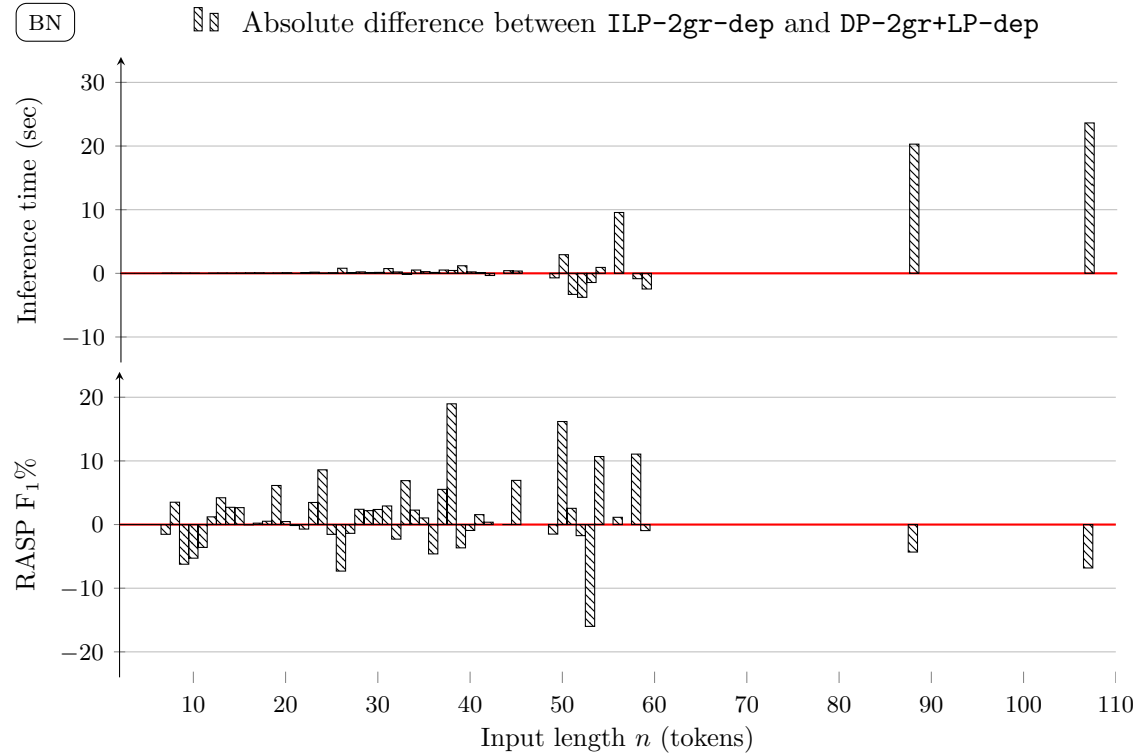


Figure 4.3: Average difference between ILP-2gr-dep and DP-2gr+LP-dep in terms of inference time (above) and RASP F_1 (below) plotted against input sentence length over the BN test dataset.

mentation of subgradient-based dual decomposition. We were also surprised to observe that the average runtime of DP-2gr+LP-dep→MST—which introduces an overhead by running the Chu-Liu Edmonds algorithm at every dual update—was noticeably lower than DP-2gr+LP-dep; however, this can be explained by examining the convergence rates of the two approaches listed in Table 4.6. DP-2gr+LP-dep→MST exhibits a higher empirical convergence rate than DP-2gr+LP-dep without significantly increasing the number of iterations to convergence; its average runtime is therefore lowered as it more readily avoids the minimum $\frac{1}{2}J_{\max}$ iterations of dual updates needed to identify oscillating solutions.

In all ILP and LP-based results, median inference time is significantly lower than average inference time, indicating that the latter measure is dominated by the largest instances with the slowest inference. We examine this in more detail by analyzing the difference between ILP-2gr-dep and DP-2gr+LP-dep, the more performant of the two multi-structured inference techniques, as the length of the input sentence n varies. Figure 4.3 compares the

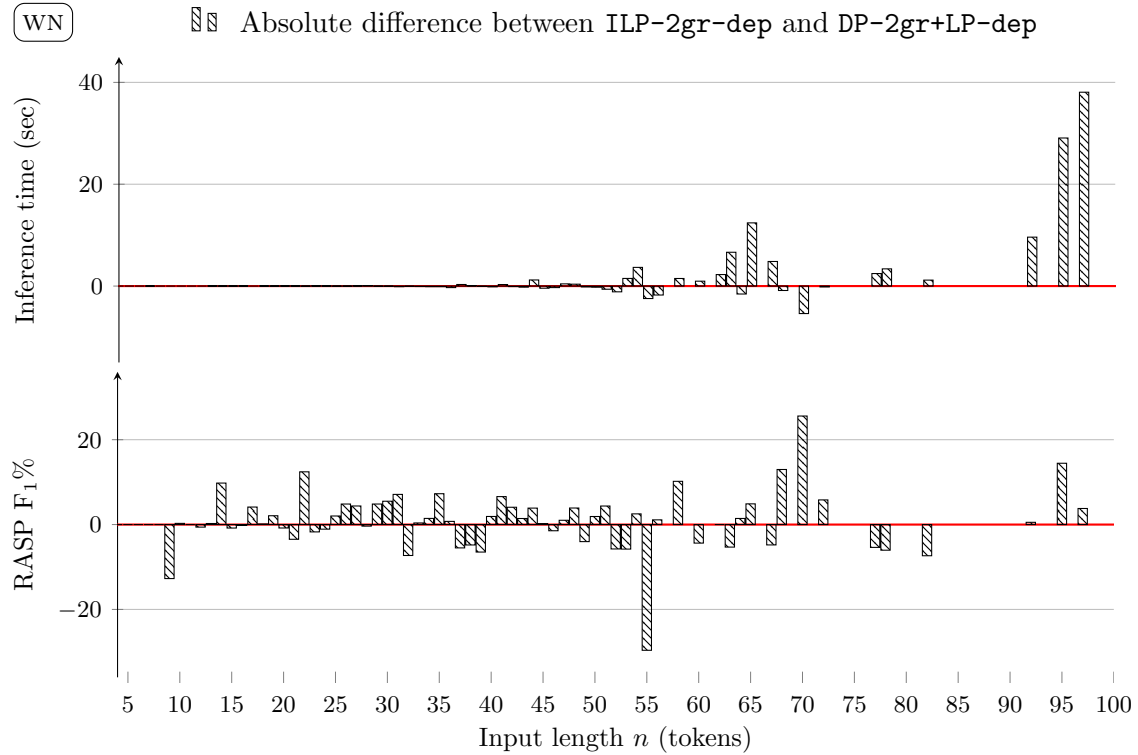


Figure 4.4: Average difference between ILP-2gr-dep and DP-2gr+LP-dep in terms of inference time (above) and RASP F_1 (below) plotted against input sentence length over the WN test dataset.

absolute difference in inference time and compression quality—measured as RASP F_1 —between these systems on the BN test dataset. For sentences with less than 50 tokens, DP-2gr+LP-dep appears occasionally faster while ILP-2gr-dep gains more frequently on RASP F_1 . Longer sentences,¹³ however, are often much faster under DP-2gr+LP-dep with inconsistent results on RASP F_1 . Similar results are seen over the WN corpus in Figure 4.4 with a more pronounced gain in RASP F_1 for ILP-2gr-dep balanced out by dramatic improvements in DP-2gr+LP-dep inference time for longer instances—in one case, saving 38 seconds over ILP-2gr-dep.

This suggests that a practical multi-structured compression system may be able to exploit the advantages of both approaches by using ILP-2gr-dep for short sentences and DP-2gr+LP-dep for long sentences with little deterioration in overall compression quality as

¹³In some cases, these longer inputs are not single sentences but a result of errors in sentence segmentation in the original corpora.

long as the latter is rare. We also note that the runtime and performance of the approximate techniques remains variable and is highly dependent on the hyperparameters and strategies used to limit dual updates, thereby making it amenable to situations in which compressions must be produced under a strict time budget.

4.3 Remarks

The central contribution of this chapter is a natural dual decomposition variant of our approach to multi-structured compression from Chapter 3. We propose fast LP relaxations alongside an efficient maximum spanning tree algorithm to quickly recover approximate solutions for dependency-based compression with a minimal loss in performance. When pairing this with efficient bigram-based compression, we find that we can largely emulate the performance of the ILP approach with a decrease in average runtime. Although the advantages of this particular compression system are surpassed in the following chapter, the proposed template of dual decomposition with relaxation-based approximations for hard subproblems remains viable for other tasks, including the sentence fusion task from Chapter 7 and the phrase-based alignment problem from Chapter 8.

Despite the relatively low rate of optimal solutions produced by an LP relaxation and approximation technique for recovering compressed dependency trees, their use in dual decomposition approaches to multi-structured compression yields significant improvements over bigram and dependency-based compression techniques and come close to matching the performance of the exact ILP approach described in Chapter 3. This improvement comes with reduced average inference time over both datasets when compared to the ILP, even though the latter relies on a state-of-the-art solver and exploits multiple CPU cores while our implementation of the approximate techniques is relatively unoptimized.

Our analysis reveals further opportunities for improving the runtime of practical compression techniques through a hybrid approach which only triggers approximate inference for long input sentences. Furthermore, this approach remains compatible with scenarios in which inference runtime is bounded, since iterative dual decomposition can simply be used to opportunistically improve over bigram-based compression within the available time

budget for each instance.

Although the dynamic program for bigram-based compression relies on the assumption that the ordering of output tokens is fixed beforehand, it remains feasible—and a potential area of future research—to replace this with alternative exact or approximate approaches to n-gram-based compression. Furthermore, when the assumption is appropriate—as in the extractive compression scenarios explored here—we can extend it to the dependency compression approach and develop efficient techniques for recovering *projective* compressed dependency trees under a multi-structured objective as described in the following chapter.

Chapter 5

Efficient Compression via Dynamic Programming

We have thus far considered techniques for multi-structured compression that, while practically useful, offer exponential worst-case time complexity. Chapter 3 considered a formulation of the problem with no restrictions on the token ordering, n-gram factorization and dependency tree of the output sentence but consequently required the use of an ILP solver for inferring optimal compressions. In Chapter 4, this formulation was decomposed into separate subproblems for recovering output n-grams and dependency trees. However, while the former could be made efficient by restricting output compressions to paths over order-preserving bigrams, an approximate approach was necessary for the latter and consequently for the joint problem—thereby sacrificing the guarantee of optimal solutions in order to achieve a practical runtime improvement.

In this chapter, we investigate a formulation for multi-structured compression that permits both efficient *and* exact inference.¹ We present polynomial-time algorithms for *compressive parsing* which generate optimal dependency trees over compressions of an input sentence, extending the well-known Eisner algorithm for projective parsing (Eisner, 1996) in order to drop a fixed or variable number of tokens in the output parse tree. Our formulation

¹This is joint work with Alexander Rush. A similar approach to this problem was independently developed by Qian and Liu (2014) albeit with higher runtime complexity than the techniques we present here. Both approaches can be seen as specializations of the lattice parsing techniques in Eisner and Smith (2010).

also scores a bigram factorization of the compressed sentence with no asymptotic overhead. Finally, this dynamic program can easily be extended to richer second-order compressive parsing (Eisner, 1996; McDonald and Pereira, 2006) which permits scores to be defined over consecutive parsing decisions without any further increase in runtime.

The contributions of this chapter include:

- An $O(n^3)$ time dynamic programming algorithm to jointly recover the optimal compressed dependency tree and bigram factorization over an input sentence of length n when no compression rate is specified.
- A variant of this technique which permits the parameterization of second-order dependencies with no overhead in asymptotic complexity.
- An $O(n^3m^2)$ time algorithm to recover the optimal compressed tree and bigram path covering exactly $m < n$ output tokens.
- A bisection-based approximate approach which relaxes the length constraint in order to trade off exact solutions for better runtime performance.

5.1 Compressive Parsing

In Chapter 4, the inference problem for dependency structure is cast as the NP-hard problem of finding the maximum-weight subtree in an arbitrary directed graph, with no relationship to the ordering of tokens in the solution. However, both the compression dataset in §3.1 and the dynamic program for bigram paths in §4.1.2 abide by the assumption that the order of tokens in the input determines the ordering of output tokens. Extending this assumption to the dependency subproblem permits the design of polynomial-time algorithms that recover optimal *projective* dependency trees over the output compression.

Projective trees are only defined over nodes that comprise a linearly ordered set² such as the words in a sentence. Formally, a tree is said to be projective if every subtree covers nodes which are contiguous in the linear ordering. In the context of dependency parsing, a projective dependency tree is one in which a token and all its syntactic descendents form a

²Specifically, assuming the presence of an ordering relation to map nodes into positions, the set of nodes must abide by a *strict* total order which rules out nodes that share the same position.

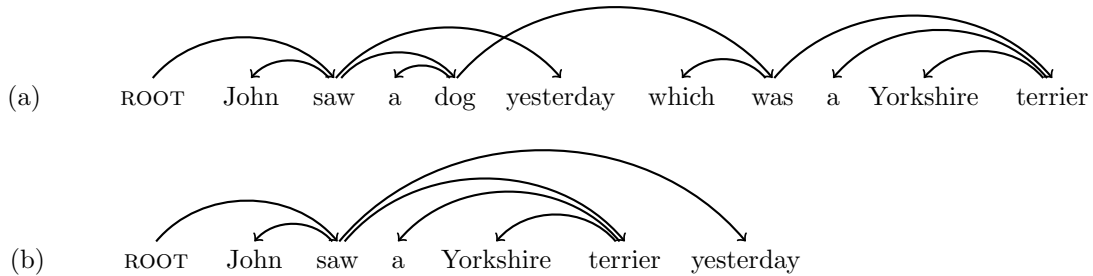


Figure 5.1: Examples of unlabeled dependency analyses with (a) a non-projective parse drawn from McDonald et al. (2005b), and (b) a projective parse with no crossing edges.

contiguous substring in the sentence, or, equivalently, a tree which can be drawn over the sentence tokens with no crossing edges, as shown in Figure 5.1.

Most sentences in languages with treebank resources—including English—can be analyzed with projective trees (Buchholz and Marsi, 2006), and even canonical non-projective languages such as Czech, Danish and Turkish have a low rate (1-2%) of non-projective edges in their respective treebanks (Nivre and Nilsson, 2005). Even though non-projective dependency parsing is tractable using MST algorithms (McDonald et al., 2005b), imposing near-projectivity is often beneficial in parsing evaluations (Martins et al., 2009). Restricting compressed dependency trees to be projective is therefore a compelling tradeoff for efficient multi-structured inference in extractive compression scenarios.

In this section, we modify a well-known dynamic programming approach to projective dependency parsing (Eisner, 1996) in order to recover a compressed tree structure over a subset of input tokens as well as a bigram path which is consistent with the tree. This yields an optimal solution to the multi-structured compression formulation from §3.2 under the assumptions that

1. There exists a linear ordering over all possible output tokens in \mathbf{x} .
2. The n -gram factorization in \mathbf{y} is restricted to bigrams.
3. The dependency tree in \mathbf{z} is projective.

In §5.1.3, we extend this to second-order dependency variables with no further increase in asymptotic runtime complexity. However, imposing a compression rate does introduce a runtime penalty; we explore options to minimize this in §5.1.4.

5.1.1 Edge-factored parsing

We begin by assuming that all possible output tokens in \mathcal{T} form a linear ordering t_1, \dots, t_n . As in §3.2, the START of the output bigram path \mathbf{y} and ROOT of the output tree \mathbf{z} are both denoted by the auxiliary token t_0 which precedes the tokens from the input sentence. Similarly, the END of the path defined by \mathbf{y} is denoted by t_{n+1} which follows the input tokens.

The standard algorithm for projective dependency parsing is known as Eisner’s algorithm (Eisner, 1996) and is specified through a set of deductive rules acting on items in a dynamic programming chart. Each item consists of a tuple (σ, i, j) where σ is a symbol in $\{\triangleleft, \triangleright, \sqsubseteq, \sqsupseteq\}$ and $\langle i, j \rangle$ represents a span over the ordered tokens where $0 \leq i \leq j \leq n$. Items in the chart indicate partial structures which can be combined with logical rules: for instance, an item (\triangleright, i, j) indicates a subtree rooted at t_i and covering its right dependents up to t_j and (\triangleleft, i, j) indicates a subtree rooted at t_j and covering its left dependents starting from t_i . The chart is initialized with left and right subtree items for each of the n tokens as well as the ROOT at t_0 .³

$$\boxed{\text{PREMISES}} \quad (\triangleright, i, i), (\triangleleft, i, i) \quad \forall i \in \{0 \dots n\} \quad (5.1)$$

In edge-factored projective parsing, dependency attachments are established through two sets of deductive rules in order to ensure that the resulting dependency structure forms a valid tree. The first set of rules creates new dependency attachments by combining the heads of adjacent left and right subtrees, i.e., items of the form (\triangleright, i, k) and $(\triangleleft, k + 1, j)$. A right attachment z_{ij} (which makes t_j a dependent of t_i) entails the combination of these items to produce a new partial structure (\sqsubseteq, i, j) and is written as

$$\frac{(\triangleright, i, k) \quad (\triangleleft, k + 1, j)}{(\sqsubseteq, i, j)} \Delta_{\text{dep}}(S, i, j) \quad \forall i \leq k < j \quad (5.2)$$

where the symbol to the right of the rule indicates the *consequence* of applying the rule—in this case, the addition of the corresponding dependency score $\Delta_{\text{dep}}(S, i, j)$ to the total cost

³Note that the left subtree $(\triangleleft, 0, 0)$ of the ROOT and the right subtree (\triangleright, n, n) of the final token are unnecessary. We omit this detail in the rules for brevity.

of the output structure. Similarly, a left attachment z_{ji} combines the same items to produce a different partial structure (\sqsupset, i, j) over the same span.

$$\frac{(\triangleleft, i, k) \quad (\triangleleft, k+1, j)}{(\sqsupset, i, j)} \Delta_{\text{dep}}(S, j, i) \quad \forall i \leq k < j \quad (5.3)$$

The result (\sqsupset, i, j) of a right attachment from (5.2) indicates a partial subtree that must be *completed* by consuming the remaining right subtree of the dependent t_j , thereby ensuring that t_j cannot also make a left attachment in the output structure. Similarly, the result (\sqsupset, i, j) of a left attachment from (5.3) is completed by the left subtree of the dependent t_i . These steps are accomplished through the following completion rules which do not affect the structure score.

$$\frac{(\sqsupset, i, k) \quad (\triangleleft, k, j)}{(\triangleleft, i, j)} \quad \forall i < k \leq j \quad (5.4)$$

$$\frac{(\triangleleft, i, k) \quad (\sqsupset, k, j)}{(\triangleleft, i, j)} \quad \forall i \leq k < j \quad (5.5)$$

Finally, a valid solution for the dynamic program must take the form of a right tree rooted at t_0 and spanning all tokens in the input sentence.

$$\boxed{\text{GOAL}} \quad (\triangleleft, 0, n) \quad (5.6)$$

This specifies a dynamic program for recovering an uncompressed projective parse tree of the input sentence—optimal under the objective $\mathbf{z}^\top \mathbf{\Delta}_{\text{dep}}$ —in $O(n^3)$ running time with $O(n^2)$ space.⁴

5.1.2 Bigram-factored compressions

In order to target the multi-structured objective from (3.3) in the edge-factored dependency parsing algorithm above, we introduce scores Δ''_{ngr} over bigrams where we define

$$\Delta''_{\text{ngr}}(S, i, j) \triangleq \Delta_{\text{ngr}}(S, i, j) + \Delta_{\text{tok}}(S, j) \quad (5.7)$$

to fold the scores for individual output tokens into those for output bigrams. These bigram scores can now be incorporated directly into the dependency attachment rules of the

⁴Chart items have two free variables ranging over n and rules involve at most three.

dynamic program, e.g., (5.2) could be updated as shown

$$\frac{(\triangleleft, i, k) \quad (\triangleleft, k+1, j)}{(\sqsupset, i, j)} (\Delta_{\text{dep}}(S, i, j) + \Delta''_{\text{ngr}}(S, k, k+1)) \quad \forall i \leq k < j \quad (5.8)$$

to introduce bigrams $y_{k(k+1)}$ alongside right attachments z_{ij} while (5.3) could be similarly modified for left attachments. However, these updated scores would have no effect without permitting compression in the output structure.

For compressive parsing, the dynamic program must be able to account for *gaps*, i.e., dropped tokens that are not descendants of either input subtree. To accomplish this, we could modify the attachment rules to skip words by combining non-adjacent subtrees.

$$\frac{(\triangleleft, i, k) \quad (\triangleleft, l, j)}{(\sqsupset, i, j)} (\Delta_{\text{dep}}(S, i, j) + \Delta''_{\text{ngr}}(S, k, l)) \quad \forall i \leq k < l < j \quad (5.9)$$

This rule produces a right attachment z_{ij} with a bigram y_{kl} and drops tokens t_{k+1}, \dots, t_{l-1} in the output sentence. With a similar modification to the left attachment rule (5.3), the dynamic program generates the desired output for multi-structured compression. However, the modification also introduces an additional free variable l in the rules and consequently yields an algorithm which requires $O(n^4)$ time.⁵

Observe, though, that it is not necessary to produce dependencies and bigrams with the same rule. Instead, we can specify rules to first *predict* if there will be a gap from t_{k+1}, \dots, t_{l-1} and then assume that these tokens are dependents of t_k and therefore included in the right subtree rooted at t_i when applying the standard attachment rules (5.2)–(5.3). This optimization is known as the “hook trick” (Eisner and Satta, 1999) and can be implemented by replacing initial right subtree items (\triangleleft, i, i) with special items (\triangleright, i, i) which are only allowed to skip words to their right before becoming standard \triangleleft items.⁶

⁵Qian and Liu (2014) have recently presented a similar approach for efficient multi-structured compression in $O(n^4)$ time and $O(n^3)$ space without a compression rate restriction. However, the dynamic program described here—independently developed by Alexander Rush and the author—improves over these bounds by a factor of n and introduces a variant for second-order compressive parsing.

⁶We are grateful to Ryan McDonald for the initial suggestion to consider the hook trick in this setting.

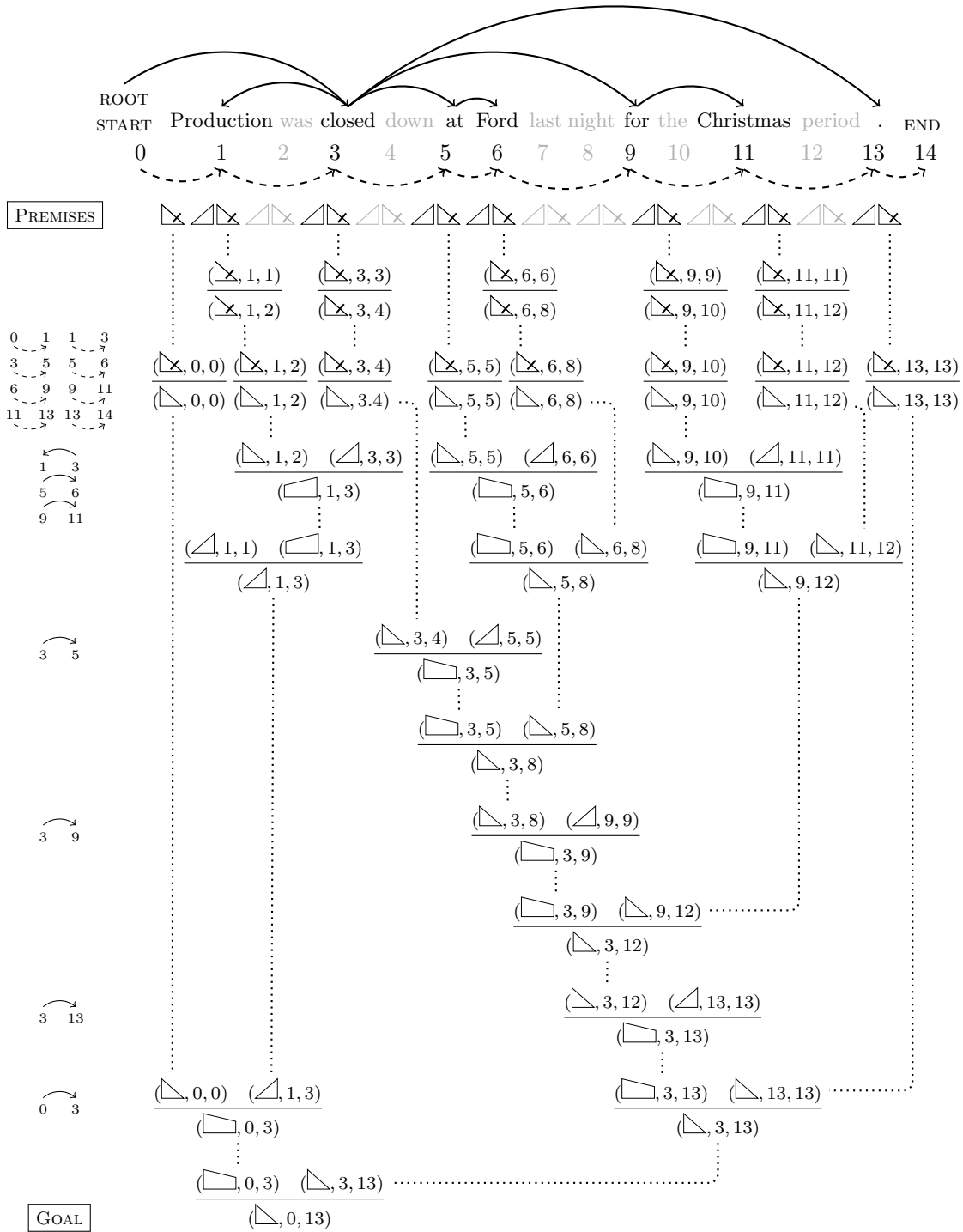


Figure 5.2: An example derivation for edge-factored compressive parsing. Dependencies are indicated by solid lines and established by \square and \square items while bigrams are indicated by dashed lines and defined whenever \boxtimes items are converted to \square items.

The full dynamic program for multi-structured compression therefore initializes the chart with the following items for each token in lieu of (5.1)

$$\boxed{\text{PREMISES}} \quad (\Downarrow, i, i), (\triangleleft, i, i) \quad \forall i \in \{0 \dots n\} \quad (5.10)$$

where a special item (\Downarrow, i, i) can consume any number of tokens to its right

$$\frac{(\Downarrow, i, i)}{(\Downarrow, i, j)} \quad \forall i < j \quad (5.11)$$

prior to becoming a regular right subtree item and in the process activating the bigram $y_{i(j+1)}$ which straddles the dropped tokens t_{i+1}, \dots, t_j .

$$\frac{(\Downarrow, i, j)}{(\Downarrow, i, j)} \Delta''_{\text{ngr}}(S, i, j + 1) \quad \forall i \leq j \quad (5.12)$$

The standard dependency parsing rules (5.2)–(5.5) can then be used to target the goal state (5.6) and thereby find an optimal compressed path and projective tree that maximizes the full multi-structured objective from (3.3). As no rule involves more than two free variables and there are at most n dependency attachments for a solution, this dynamic program retains the $O(n^3)$ runtime complexity of Eisner’s algorithm. A full compressive parse derivation for the example from §3.2 is provided in Figure 5.2.

5.1.3 Second-order parsing

The straightforward adaption of first-order parsing to the multi-structured compression task introduces the possibility of incorporating higher-order dependency structure from the parsing literature in our formulation. We describe here an approach to compression inspired by second-order extensions to Eisner’s algorithm (Eisner, 1996; McDonald and Pereira, 2006) and note that higher-order dependency factorizations (Carreras, 2007; Koo and Collins, 2010; Pitler, 2014) also appear tractable albeit with an inevitable increase in time complexity.

In second-order dependency parsing, we want to score not only single dependency edges like $t_i \rightarrow t_j$ but to also take into account the previous token modifying t_i in the same direction, known as a *sibling*. For instance, with a right attachment $t_i \rightarrow t_j$ where $i < j$, the

sibling of t_j is the token t_k with the largest index $k \in \{i+1, \dots, j-1\}$ such that $t_i \rightarrow t_k$ or ϵ if no such index exists. Similarly, for a left attachment $t_j \rightarrow t_i$ where $i < j$, the sibling of t_i is the token t_k with the smallest index $k \in \{i+1, \dots, j-1\}$ such that $t_j \rightarrow t_k$ or ϵ if no such index exists. We denote second-order dependencies with variables z_{ikj} where the middle index k refers to the sibling and score them with revised scoring functions $\Delta_{\text{dep}}(S, i, k, j)$, leaving the remainder of our compression formulation unchanged.

The dynamic program for second-order parsing requires a new type of item (\square, i, j) which is created by combining adjacent right and left subtrees headed by t_i and t_j respectively. This item indicates that t_i and t_j will eventually be attached to some common head which is currently unknown.

$$\frac{(\triangleleft, i, k) \quad (\triangleleft, k+1, j)}{(\square, i, j)} \quad \forall i \leq k < j \quad (5.13)$$

The first right or left dependency attachment to any token has no siblings and therefore follows the attachment rules from first-order parsing.

$$\frac{(\triangleleft, i, i) \quad (\triangleleft, i+1, j)}{(\square, i, j)} \Delta_{\text{dep}}(S, i, \epsilon, j) \quad \forall i < j \quad (5.14)$$

$$\frac{(\triangleleft, i, j-1) \quad (\triangleleft, j, j)}{(\square, i, j)} \Delta_{\text{dep}}(S, j, \epsilon, i) \quad \forall i < j \quad (5.15)$$

Subsequent right or left dependencies for these tokens consume adjacent \square items. As seen in (5.13), these rules simultaneously complete the previous right or left attachment—by consuming its remaining \triangleleft or \triangleleft subtree respectively—and establish a sibling relationship for the new attachment.

$$\frac{(\square, i, k) \quad (\square, k, j)}{(\square, i, j)} \Delta_{\text{dep}}(S, i, k, j) \quad \forall i \leq k < j \quad (5.16)$$

$$\frac{(\square, i, k) \quad (\square, k, j)}{(\square, i, j)} \Delta_{\text{dep}}(S, j, k, i) \quad \forall i \leq k < j \quad (5.17)$$

The final right or left dependency of any token will not be completed by \square items, so we also require the completion rules (5.4)–(5.5) from edge-factored parsing. This set of rules can now be combined with the original parsing premises (5.1) and goal state (5.6) to yield a dynamic program for second-order dependency parsing in $O(n^3)$ time.

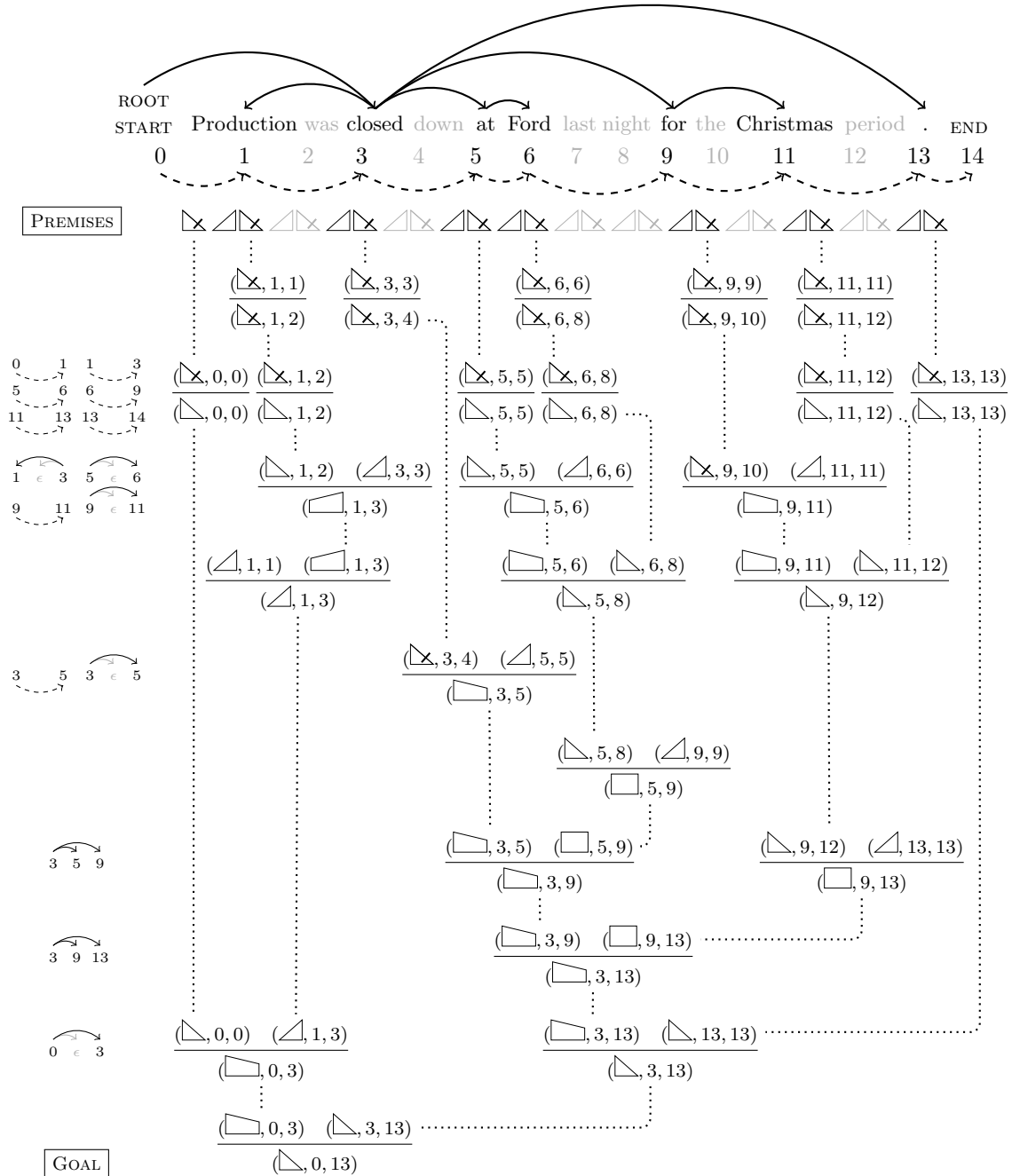


Figure 5.3: An example derivation for second-order compressive parsing. Dependencies are indicated by solid lines and established by \square and \square items while bigrams are indicated by dashed lines and defined whenever \boxtimes items are converted to \triangleleft or \square items.

Just as before, we can extend this approach to second-order *compressive* parsing with the modifications from §5.1.2, i.e., by introducing revised premises (5.10) and token-skipping bigram rules (5.11)–(5.12). However, a further modification is necessary for the second-order scenario. Recall that the hook trick in §5.1.2 relies on introducing special items (\boxtimes, i, j) that skip tokens t_{i+1}, \dots, t_j and are then converted to regular right subtrees (\triangleleft, i, j) that can participate in regular attachment rules. While this approach remains compatible with most second-order attachment rules, the rule for *initial* right attachments in (5.14) is necessarily restricted to the single-token item (\triangleleft, i, i) which by definition cannot accommodate a gap. Therefore, the dynamic program described thus far does not permit gaps between any token and its first right dependent.

To rectify this limitation, we include an additional rule that permits initial right attachments directly for the special \boxtimes items which may contain gaps. An application of this rule would preclude the conversion of \boxtimes items to \triangleleft items via (5.12) and therefore the consequence for this rule must incorporate the score of the corresponding bigram.

$$\frac{(\boxtimes, i, k) \quad (\triangleleft, k+1, j)}{(\triangleleft, i, j)} (\Delta_{\text{dep}}(S, i, \epsilon, j) + \Delta''_{\text{ngr}}(S, i, k+1)) \quad \forall i \leq k < j \quad (5.18)$$

The dynamic program specified by premises (5.10), the goal state (5.6) and the rules (5.4)–(5.5) and (5.11)–(5.18) can now recover optimal multi-structured compressions under the joint objective (3.3) using a second-order projective dependency factorization in $O(n^3)$ time. A second-order derivation for the example from §3.2 is shown in Figure 5.3.

5.1.4 Enforcing compression rates

The efficient inference techniques described thus far rely on the absence of a compression rate ω for the output. Imposing a restriction on the length of an output sentence $\mathbb{1}^\top \mathbf{x}$ to equal a particular value $m \triangleq \lfloor n \cdot \omega \rfloor$ entails a runtime overhead. We discuss two strategies for this problem here.

5.1.4.1 Tracking subtree size

The most straightforward approach to enforcing the length constraint is to explicitly keep track of the size of the substructure represented by every item. Observe from Figures 5.2

and 5.3 that every token retained in the final compression corresponds to exactly one \triangleleft in the initialization of the respective dynamic programs. Keeping track of the total number of \triangleleft items encountered in the derivation of each item allows the size of the output sentence to be controlled.

Specifically, we redefine the items to be larger tuples (σ, i, j, q) where, as before, $\sigma \in \{\triangleleft, \triangle, \sqsupset, \sqsubset, \boxtimes, \square\}$, $0 \leq i \leq j \leq n$ and $q \leq m$ is the size of the partial structure represented by the item. On initialization, every token can be associated with its \triangleleft item.

$$\boxed{\text{PREMISES}} \quad (\boxtimes, i, i, 0), (\triangleleft, i, i, 1) \quad \forall i \in \{0 \dots n\} \quad (5.19)$$

The deductive rules can simply be revised to combine the sizes of the structures in their premise. Bigram rules (5.11)–(5.12) are updated to propagate the size of the item in their premise without modification⁷

$$\frac{(\boxtimes, i, i, q)}{(\boxtimes, i, j, q)} \quad \forall i < j \quad (5.20)$$

$$\frac{(\boxtimes, i, j, q)}{(\triangleleft, i, j, q)} \Delta''_{\text{ngr}}(S, i, j + 1) \quad \forall i \leq j \quad (5.21)$$

while edge-factored parsing rules (5.2)–(5.5) are updated to add together the sizes of substructures in their premises.

$$\frac{(\triangleleft, i, k, q^L) \quad (\triangleleft, k + 1, j, q^R)}{(\sqsupset, i, j, q^L + q^R)} \Delta_{\text{dep}}(S, i, j) \quad \forall i \leq k < j, \quad q^L + q^R \leq m \quad (5.22)$$

$$\frac{(\triangleleft, i, k, q^L) \quad (\triangleleft, k + 1, j, q^R)}{(\sqsubset, i, j, q^L + q^R)} \Delta_{\text{dep}}(S, j, i) \quad \forall i \leq k < j, \quad q^L + q^R \leq m \quad (5.23)$$

$$\frac{(\sqsupset, i, k, q^L) \quad (\triangleleft, k, j, q^R)}{(\triangleleft, i, j, q^L + q^R)} \quad \forall i < k \leq j, \quad q^L + q^R \leq m \quad (5.24)$$

$$\frac{(\triangleleft, i, k, q^L) \quad (\sqsubset, k, j, q^R)}{(\triangleleft, i, j, q^L + q^R)} \quad \forall i \leq k < j, \quad q^L + q^R \leq m \quad (5.25)$$

⁷Moreover, no item appearing in the bigram rules can have encountered \triangleleft items yet so q must be 0 in the revised bigram rules (5.20)–(5.21).

Algorithm 3 Subgradient descent for approximate length constraint

Input: scores Δ , iteration limit i_{\max} , learning rate schedule $\eta \in \mathbb{R}^{i_{\max}}$

Output: token configuration \mathbf{x}

- 1: $\lambda^{(0)} \leftarrow 0$
- 2: **for** iteration i **in** $0 \dots i_{\max}$ **do**
- 3: $\mathbf{x}^{(i)} \leftarrow \arg \max_{\mathbf{x}} h(\mathbf{x}, \mathbf{y}, \mathbf{z}, \Delta, \lambda^{(i)})$
- 4: **if** $\mathbb{1}^{\top} \mathbf{x}^{(i)} = m$ **then return** $\mathbf{x}^{(i)}$ // optimal solution
- 5: **if** oscillating **then break**
- 6: $\lambda^{(i+1)} \leftarrow \lambda^{(i)} + \eta_i (m - \mathbb{1}^{\top} \mathbf{x}^{(i)})$

return $\arg \min_{\mathbf{x}^{(j)}: j \leq i} |m - \mathbb{1}^{\top} \mathbf{x}^{(j)}|$ // approximation

A similar modification can be introduced to the deductive rules for second-order parsing (5.13)–(5.18). Finally, a valid solution for either dynamic program must cover a tree with m tokens.⁸

$$\boxed{\text{GOAL}} \quad (\triangleleft, 0, n, m) \quad (5.26)$$

Because every chart item has three input-dependent variables and the updated rules (5.22)–(5.23) involve at most five free variables, these dynamic programs for length-constrained compressive parsing require $O(n^2m)$ space and $O(n^3m^2)$ running time.

5.1.4.2 Lagrangian relaxation

We would naturally prefer to avoid the $O(m^2)$ runtime penalty for strict adherence to the compression rate ω . One practical alternative is Lagrangian relaxation of the length constraint $\mathbb{1}^{\top} \mathbf{x} = m$ in order to find approximate solutions with compression rates which approach ω but do not always equal it. A Lagrange multiplier λ can incorporate the length

⁸The goal state must be correspondingly revised in order for the dynamic program to satisfy a minimum and/or maximum compression rate. Note that the attachment rules (5.22)–(5.23) must range over all $q^L, q^R \leq n$ when only a minimum rate is specified.

Algorithm 4 Bisection for approximate length constraint

Input: scores Δ , initial bounds λ_{\min} and λ_{\max} , tolerance $\pm\varepsilon$
Output: token configuration \mathbf{x}

```

1:  $i_{\max} \leftarrow \lceil \log_2 \frac{1}{\varepsilon} (\lambda_{\max} - \lambda_{\min}) \rceil$ 
2: for iteration  $i$  in  $0 \dots i_{\max}$  do
3:    $\lambda^{(i)} \leftarrow \frac{1}{2} (\lambda_{\min} + \lambda_{\max})$ 
4:    $\mathbf{x}^{(i)} \leftarrow \arg \max_{\mathbf{x}} h(\mathbf{x}, \mathbf{y}, \mathbf{z}, \Delta, \lambda^{(i)})$ 
5:   if  $\mathbb{1}^\top \mathbf{x}^{(i)} = m$  then return  $\mathbf{x}^{(i)}$  // optimal solution
6:   else if  $\mathbb{1}^\top \mathbf{x}^{(i)} < m$  then  $\lambda_{\min} \leftarrow \lambda^{(i)}$ 
7:   else if  $\mathbb{1}^\top \mathbf{x}^{(i)} > m$  then  $\lambda_{\max} \leftarrow \lambda^{(i)}$ 

return  $\arg \min_{\mathbf{x}^{(j)}: j \leq i} |m - \mathbb{1}^\top \mathbf{x}^{(j)}|$  // approximation
    
```

restriction in the objective from (3.3), resulting in the following Lagrangian:

$$\begin{aligned}
 L(\lambda, \mathbf{x}, \mathbf{y}, \mathbf{z}) &= \mathbf{x}^\top \Delta_{\text{tok}} + \mathbf{y}^\top \Delta_{\text{ngr}} + \mathbf{z}^\top \Delta_{\text{dep}} \\
 &\quad + \lambda \left(\mathbb{1}^\top \mathbf{x} - m \right)
 \end{aligned} \tag{5.27}$$

The corresponding dual problem is a relaxation of the problem of finding a solution to (3.3) with exactly m output tokens.

$$\begin{aligned}
 &\min_{\lambda} \max_{\mathbf{x}, \mathbf{y}, \mathbf{z}} L(\lambda, \mathbf{x}, \mathbf{y}, \mathbf{z}) \\
 &= \min_{\lambda} \max_{\mathbf{x}, \mathbf{y}, \mathbf{z}} \mathbf{x}^\top (\Delta_{\text{tok}} + \lambda \cdot \mathbb{1}) + \mathbf{y}^\top \Delta_{\text{ngr}} + \mathbf{z}^\top \Delta_{\text{dep}} - \lambda \cdot m \\
 &= \min_{\lambda} \max_{\mathbf{x}, \mathbf{y}, \mathbf{z}} h(\mathbf{x}, \mathbf{y}, \mathbf{z}, \Delta, \lambda) - \lambda \cdot m
 \end{aligned} \tag{5.28}$$

where $h(\mathbf{x}, \mathbf{y}, \mathbf{z}, \Delta, \lambda)$ can be solved in $O(n^3)$ time by running one of the dynamic programs for unconstrained compressive parsing and adding λ to the bigram scores $\Delta''_{\text{ngr}}(S, i, j)$ defined in (5.7). The outer minimization in (5.28) can be solved using subgradient descent by iteratively finding $\arg \max_{\mathbf{x}} h(\mathbf{x}, \mathbf{y}, \mathbf{z}, \Delta, \lambda^{(i)})$ and then updating $\lambda^{(i+1)}$ as shown in Algorithm 3. Alternatively, since we only have a single Lagrange multiplier to contend with, the solution can also be found by bisection as shown in Algorithm 4. Starting with some λ_{\min} and λ_{\max} and some tolerance for convergence ε , bisection will converge within $\log_2 \frac{1}{\varepsilon} (\lambda_{\max} - \lambda_{\min})$ iterations. In either approach, the solution is optimal if $\mathbb{1}^\top \mathbf{x}^{(i)} = m$ at

any iteration i . In these instances, the tightness of the length constraint (5.28) allows us to sidestep the $O(m^2)$ computational overhead of the exact approach from §5.1.4.1.

Absent convergence, an approximate solution with $k \neq m$ tokens can be retrieved. If a solution with exactly m tokens is needed, we can select an approximate solution with $k > m$ and use it to initialize a second-pass of inference with the exact dynamic program while optionally pruning the problem to minimize computational overhead. For instance, if we can assume that compression solutions vary somewhat smoothly with compression rate, the intermediate solutions from the bisection procedure can be used to eliminate unlikely output tokens from the input to a length-constrained dynamic program. Furthermore, if we assume the size of the reduced input sentence $k > m \gg k - m$, we can update the dynamic program from §5.1.4.1 to count tokens *dropped* rather than tokens retained with the following revisions to the premises (5.19), bigram rules (5.20) and goal state (5.26).⁹

$$\boxed{\text{PREMISES}} \quad (\triangleright, i, i, 0), (\triangleleft, i, i, 0) \quad \forall i \in \{0 \dots n\} \quad (5.29)$$

$$\frac{(\triangleright, i, i, 0)}{(\triangleright, i, j, j - i)} \quad \forall i < j \quad (5.30)$$

$$\boxed{\text{GOAL}} \quad (\triangleright, 0, n, n - m) \quad (5.31)$$

Thus we can use Lagrangian relaxation and dynamic programming in a coarse-to-fine strategy with the former used to solve a problem if its relaxation is tight and otherwise prune it—identifying $k > m$ likely tokens from the input—and then use the dynamic program from §5.1.4.1 with the revisions from (5.29)–(5.31) to retrieve a length m output sentence in $O(k^3(k - m)^2)$ time with $O(k^2(k - m))$ space.

5.2 Features

We reuse the features and trained models from Chapter 3 in order to evaluate the dynamic program for edge-factored multi-structured inference described in §5.1.1. However, we introduce additional features for the second-order dependency variables described in §5.1.3, enumerated below.

⁹This modification is also useful for exact length-constrained compressive parsing with compression rates under 50%, thereby reducing its runtime complexity to $O(n^3 \min(m, n - m)^2)$.

5.2.1 Second-order dependency features

The following lexico-syntactic indicators comprise feature templates for a second-order dependency edge $\langle t_i, t_k, t_j \rangle$ where t_k represents the sibling of the dependency edge $t_i \rightarrow t_j$ with t_k set to t_i iff $k = \epsilon$, i.e., when no sibling exists.

- A conjunction of the POS tags of t_i , t_j and t_k as well as the orientation of the edge, i.e., $\text{sign}(t_j - t_i)$.
- The POS tags of t_i and t_j and the edge orientation conjoined with binary indicators of whether $|t_k - t_i| \in \{0, 1\}$ and $|t_k - t_j| = 1$.
- The POS tags of t_k and t_j and the edge orientation conjoined with binary indicators of whether $|t_k - t_i| \in \{0, 1\}$ and $|t_k - t_j| = 1$.

We also experimented with various additional features over the POS tags, distances and intervening tokens covered by second-order dependencies but were not able to identify features that offered a consistent gain over these indicators in development tests.

5.3 Experiments

We now endeavor to characterize the runtime performance and compression quality of dynamic programs for multi-structured compression through additional experiments over the BN and WN compression datasets (Clarke and Lapata, 2006b; Clarke and Lapata, 2007) described in §3.1. Our experimental environment is configured as described in §3.6.

5.3.1 Compression quality

In this section, we consider the performance of the following dynamic programs¹⁰ for compression inference:

- **DP-dep**: A dynamic program based on the edge-factored compressive parsing approach that recovers projective dependency trees over a subset of tokens in the input.

¹⁰These dynamic programs were implemented using the `pydecode` toolkit maintained by Alexander Rush, a collaborator for this work. This C++/Python library is freely available at <http://www.github.com/srush/pydecode> with documentation at <http://www.pydecode.org>.

- **DP-2gr-dep**: The full dynamic program of §5.1.2 which produces an edge-factored projective dependency tree alongside a bigram-factored token ordering for the output sentence.

These approaches are compared on measures of output compression quality against the following systems previously described in §3.6.1:

- **DP-2gr**: The dynamic program of McDonald (2006) described in §4.1.2 for inference of bigram-factored compressions under a compression rate.
- **ILP-dep**: The ILP described in §3.3.1 for inferring edge-factored compressed dependency trees which are not necessarily projective.
- **ILP-2gr-dep**: The full ILP for multi-structured inference which combines the constraints from §3.3.1 to produce an edge-factored dependency tree with §3.3.2 to yield a bigram-factored token ordering for the output sentence.

For evaluation purposes, the new dynamic programming approaches use the models generated by the equivalent ILP inference approach from the list above.¹¹ The effect of non-projective solutions is expected to be negligible as the dependency trees for reference sentences are generated by the Stanford parser and do not feature non-projective edges.

Table 5.1 contains the results of an evaluation for these systems under the usual measures of compression quality described in §3.6. It is clear from these results that the restriction to projective trees for **DP-dep** and **DP-2gr-dep** has little practical consequence. The performance gap between these dynamic programming approaches and the equivalent ILPs is statistically insignificant under all measures considered.

We note similar trends in Table 5.2 which contains results for the same systems over the WN corpus. An even smaller difference is observed between the dynamic programming approaches and their ILP equivalents than the BN corpus. This is explained by the higher rate of projective trees recovered by ILPs over the WN corpus as seen in Table 5.3. We assume this disparity between the owes to the more formal language present in written news documents as opposed to the more conversational speech in broadcast news transcripts.

¹¹Albeit with LP-relaxed inference for training as in §3.6.

BN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		path	DP-2gr (McD06)	82.94	72.84	61.08	52.65	-
tree	DP-dep	82.69	70.04	56.80	47.92	75.74	70.83	65.25
	ILP-dep	82.70	70.05	56.81	47.94	75.76	70.88	65.25
path + tree	DP-2gr-dep	82.82	72.58	60.74	52.69	75.80	73.79	68.53
	ILP-2gr-dep	82.85	72.66	60.87	52.87	75.73	73.82	68.56

Table 5.1: Experimental results for the BN corpus averaged over 3 reference compressions per instance. All systems were restricted to compress to the size of the median reference compression yielding an average compression rate of 77.26%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

5.3.2 Timing

Since output quality remains largely identical to the ILP-based approaches from Chapter 3, the chief advantage of the dynamic programming techniques is in runtime performance. We evaluate these aspects in this section using the same experimental environment from §4.2.4. In addition to the systems described previously, we also consider the runtime performance of the following techniques which attempt to avoid the $O(m^2)$ penalty for counting tokens by constraining the length of the output sentence to m tokens through Lagrangian relaxation and bisection as described in §5.1.4.2.

- **DP-dep+m:** A variation of the dynamic program for compressed dependency tree inference which uses Lagrangian relaxation to impose the output sentence length as a first pass and falls back to **DP-dep** if it does not converge.
- **DP-2gr-dep+m:** A similar variation of the full dynamic program for multi-structured compression inference which first uses Lagrangian relaxation to impose the output length and then resorts to **DP-2gr-dep** if no convergence is achieved.

For the bisection procedure, we recover the value of the sole Lagrange multiplier following Algorithm 4 by binary search within $[-1000, 0]$ with the maximum number of iterations i_{\max} set to 16, leading to an effective error tolerance of $\varepsilon = 0.015$. If bisection does not

		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		path	DP-2gr (McD06)	75.36	63.40	52.15	42.97	-
tree	DP-dep	75.74	60.07	46.84	37.48	69.08	64.33	58.49
	ILP-dep	75.74	60.07	46.84	37.48	69.08	64.33	58.49
path + tree	DP-2gr-dep	76.14	64.47	53.62	45.15	69.51	67.34	61.78
	ILP-2gr-dep	76.15	64.47	53.63	45.15	69.52	67.35	61.82

Table 5.2: Experimental results for the WN corpus with all systems compressing to the size of the reference compression, yielding an average compression rate of 70.24%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

Structures	Inference	Projective z %	
		BN	WN
tree	ILP-dep	98.02	99.83
path + tree	ILP-2gr-dep	97.03	99.83

Table 5.3: Fraction of system-generated dependency tree solutions **z** which are projective over the BN and WN test datasets.

converge on the optimal solution, we then run the full $O(n^3m^2)$ dynamic program. We did not experiment with further optimizations such as seeding the second pass dynamic program with a solution from the bisection approach as suggested in §5.1.4.2; however, this remains an area of interest for future implementations of this compression approach.

Table 5.4 lists the average and median inference time for the techniques studied over the BN and WN test datasets. Unsurprisingly, we observe that the use of dynamic programming for inference as opposed to ILP results in a significant decrease in runtime over both corpora. Average runtime is decreased by 20–40% for dependency-based compression and by 60–80% for multi-structured compression over the equivalent ILP techniques. Median runtime remains significantly lower than average runtime for these systems, indicating that slow inference over large instances dominates the timing measures. Finally, while the $O(n^2m)$

Structures	Inference	Average time (sec)		Median time (sec)	
		BN	WN	BN	WN
path	DP-2gr (McD06)	0.01	0.01	0.01	0.01
tree	DP-dep	0.12	0.18	0.01	0.04
	DP-dep+m	0.10	0.14	0.01	0.04
	ILP-dep	0.18	0.30	0.04	0.11
path + tree	DP-2gr-dep	0.18	0.19	0.01	0.04
	DP-2gr-dep+m	0.11	0.19	0.01	0.04
	ILP-2gr-dep	0.48	0.93	0.10	0.26

Table 5.4: Time in seconds for inference over the BN and WN test datasets, excluding the time required for initializing and scoring features.

algorithm for DP-2gr prevails in terms of average time over the $O(n^3m^2)$ approach used for DP-dep and DP-2gr-dep, the median runtime of all dynamic programming approaches remains broadly similar.

Using Lagrangian relaxation and the iterative bisection procedure as a first pass for recovering compressions of the required length results in further runtime improvements with no change in system-generated compressions. Average convergence rates and the number of iterations to convergence for this procedure are reported in Table 5.5. We observe that convergence is more frequent over the smaller sentences of the BN corpus and note a drastic decrease in coverage rates when bigram scores are introduced to the algorithm. With our hyperparameter configuration, the use of bisection does not result in consistent runtime improvements due to the overhead of the two-pass approach which is incurred by non-converging instances; this may be ameliorated if resources are available for the two procedures to be run in parallel. Furthermore, since runtime advantages are dominated by the convergence of large instances—a phenomenon observed in these experiments as well—a practical implementation of multi-structured compression could obtain equivalent or further runtime improvements by confining bisection to these instances.

Structures	Inference	Convergence %		Num. iterations	
		BN	WN	BN	WN
tree	DP- dep +m	75.99	67.33	5.75	6.96
path + tree	DP- 2gr-dep +m	46.53	35.82	4.32	5.93

Table 5.5: Fraction of instances which converge under bisection and the average number of iterations to convergence for them in the BN and WN test datasets.

5.3.3 Second-order dependencies

In addition to edge-factored dependencies, the proposed dynamic programs for multi-structured compression admit second-order dependencies with no asymptotic increase in runtime complexity.¹² We evaluate the effect of the second-order parameterization on measures of compression quality through the following systems.

- DP-**dep**²: A dynamic program based on the second-order compressive parsing approach that recovers projective dependency trees over a subset of tokens in the input.
- DP-**2gr-**dep****²: The full dynamic program of §5.1.3 which produces a second-order projective dependency tree alongside a bigram-factored token ordering for the output sentence.

New models are trained for these techniques using the minibatched structured perceptron as described in §3.5.

Table 5.6 contains the results over the BN corpus. We observe that second-order dependency factorizations offer dramatic improvements to the performance of dependency-only compression models but only moderate improvements over multi-structured compression models. The improvement by DP-**dep**² over DP-**dep** is reflected largely in n-gram F_1 measures—a gain of nearly 5% for trigrams and 4-grams—but not in dependency F_1 , although the n-gram improvements translate to significant gains for the re-parsed Stanford and RASP F_1 measures as well. In contrast, DP-**2gr-**dep****² appears largely similar to DP-**2gr-**dep**** and the statistically significant differences—improved trigram F_1 and, some-

¹²However, feature generation for higher-order dependencies invariably introduces an additional overhead.

BN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		tree	DP-dep	82.69	70.04	56.80	47.92	75.74
DP-dep ²	82.86		73.23	61.53	53.02	75.58	74.56	69.13
path + tree	DP-2gr-dep	82.82	72.58	60.74	52.69	75.80	73.79	68.53
	DP-2gr-dep ²	82.30	73.05	61.86	53.28	74.50	74.13	68.57

Table 5.6: Experimental results for the BN corpus averaged over 3 reference compressions per instance. All systems were restricted to compress to the size of the median reference compression yielding an average compression rate of 77.26%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

what surprisingly, lower dependency F₁—suggest that the benefit of second order dependencies is primarily in capturing local ordering information.

Similar patterns are observed in Table 5.7 over the WN corpus, although the typically stronger performance of the joint models on written news data is also visible here. The improvement for DP-dep² over DP-dep is more noticeable here with an increase of more than 7% in 4-gram F₁, while the joint variant DP-2gr-dep² does not display a statistically significant gain over DP-2gr-dep in any measure other than F₁ over Stanford dependencies after parsing system outputs. This appears to align with our conjecture that second-order dependencies are useful surrogates for n-gram structure.

Why are second-order dependency models not more helpful in identifying output dependencies? We hypothesize that the small size of the Edinburgh compression corpus limits the effectiveness of our more powerful representations including second-order dependencies and the supervised trigram models from §3.6.5, which offer similar boosts in compression quality. When tuning features on the respective development corpora, we observed that these higher-order models would overfit more readily and that a broadly effective set of features was challenging to identify. It is likely that the use of larger compression corpora such as the dataset proposed by Filippova and Altun (2013) will result in clearer advantages of these techniques, similar to the gains offered by second-order dependency models in dependency parsing.

WN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		tree	DP-dep	75.74	60.07	46.84	37.48	69.08
DP-dep ²	76.09		63.91	52.58	44.00	69.42	67.17	61.77
path + tree	DP-2gr-dep	76.14	64.47	53.62	45.15	69.51	67.34	61.78
	DP-2gr-dep ²	76.07	64.53	53.72	45.42	69.18	68.03	62.45

Table 5.7: Experimental results for the WN corpus with all systems compressing to the size of the reference compression, yielding an average compression rate of 70.24%. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

5.4 Remarks

The contributions of this work include efficient dynamic programs for multi-structured sentence compression when assuming a fixed token ordering and projective parse trees over the compressed sentence. This results in a large improvement in runtime for inference with no meaningful variation in output quality when compared to the ILP from Chapter 3. We also show how these models can be extended to second-order dependencies to further improve output quality and paired with bisection to further minimize the runtime cost of imposing a compression rate. These algorithms therefore appear to be especially well-suited for practical implementations of multi-structured sentence compression.

We observe that the dependency trees produced under multi-structured ILP inference are largely projective and that a restriction to projectivity has no practical effect on the generated compressions—likely attributable to the projective Stanford parses used for reference compressions. For extractive compression in largely projective languages like English, these dynamic programs therefore offer no disadvantages while yielding an order-of-magnitude improvement in median runtime over equivalent ILPs and improved robustness on compression performance over McDonald (2006).

The proposed extension to second-order dependencies while preserving runtime complexity also appears promising, although we believe that evaluations in this setting are hampered

by the small size of the compression evaluation corpus. Although the improvements seen in our evaluations are similar to those produced by the ILP for multi-structured inference with trigrams (cf. §3.6.5), the runtime difference makes the second-order dependency approach far more usable in practice.

Although all the multi-structured approaches discussed thus far produce parse trees for compressed text, the extension of a well-known parsing algorithm for this task raises the question of whether treebanks can be employed in estimating parameters for dependency edge features directly rather than using potentially noisy parse structures over input sentences and reference compressions. We are actively exploring potential synergies between parsing and compression tasks, both of which may benefit from large-scale extractive compression datasets that can be acquired with relatively little cost following Filippova and Altun (2013).

Chapter 6

Compression over Predicate-Argument Structures

We have previously examined distinct algorithms for multi-structured sentence compression and found that practical speedups and even asymptotic efficiency are available through relaxations and restrictions over the output structures. However, a particularly appealing attribute of the original ILP approach to compression from Chapter 3 is its *flexibility*: additional linear variables and constraints can easily be included to accommodate the needs of different tasks and domains. For instance, when working in a domain with a shortage of training data, we can impose linguistically-motivated restrictions such as those proposed by Clarke and Lapata (2008) or even restrict outputs to follow specific templates, a common strategy in practical text generation systems. Alternatively, we might adapt this approach to similar text-to-text tasks such as text simplification and sentence fusion, the latter of which is considered in Chapter 7. And finally, we can expand beyond n-gram and dependency structure to new forms of structured abstraction, which is the topic of this chapter.

Although most statistical sentence compression techniques are based on either syntactic parses or n-gram factorizations, numerous other forms of structured text representation are studied in natural language analysis research. We turn our attention now to structures which aim to capture semantic relationships that lie beyond the surface syntax of a sentence. For example, in the sentences “*John closed the door.*” and “*The door was closed.*”, *the door*

is the object and subject of the predicate *closed* respectively but their relationship clearly remains constant and should be recognized as such in a sufficiently high-level representation of text meaning.

We are particularly interested in structured representations of semantics that can contribute to high-level problems such as text generation. Among the wide variety of diverse semantic resources that have received attention over the years, some of the most well-known are the data-driven Propbank (Palmer et al., 2005) built around verb senses and their arguments,¹ as well as the FrameNet lexicon (Fillmore et al., 2003), built around the theory of *frame semantics* (Fillmore, 1982). A recent entrant to this landscape is the Abstract Meaning Representation (Banarescu et al., 2013) which combines Propbank-style argument annotations with entity coreference to build a full-sentence semantic representation.

In the most general case, these structured semantic representations can be viewed as *directed acyclic graphs* defined over tokens as well as non-token nodes representing categories such as predicate labels. We therefore first address the general problem of recovering the highest-scoring directed acyclic graph over predicate labels from an input sentence by further extending the flow formulation from §3.2. We then consider a specialization of this approach for sentence compression, in which we aim to recover the semantic structure of the output sentence in the form of FrameNet frame-semantic relationships as generated by the SEMAFOR tool (Das et al., 2013).² We target Framenet because of its relative maturity as a semantic formalism (as opposed to AMR) as well as its relative orthogonality to dependency syntax (as opposed to Propbank); however, our ILP framework can support both of these as well as combinations thereof.

The contributions of this chapter are:

- An extension of the ILP formulation in §3.2 to the joint inference of directed acyclic graphs in addition to paths and trees.
- A practical application of this approach to recovering FrameNet predicate-argument relations for output compressions.

¹The NomBank project extends Propbank-style arguments to nouns (Meyers et al., 2004).

²A demo of SEMAFOR 2.1 is available at <http://demo.ark.cs.cmu.edu/parse>.

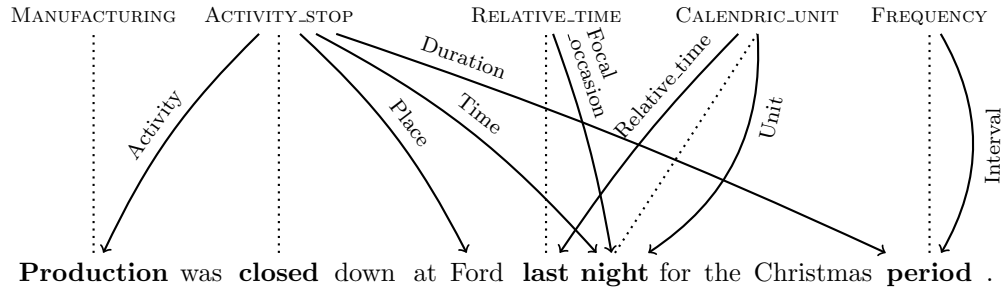


Figure 6.1: A frame-semantic parse of an example sentence using frames from the FrameNet lexicon (Fillmore et al., 2003). Boldfaced words evoke frames through dotted lines. Frame elements (FEs) are denoted with labeled arrows to head words of the corresponding text spans.

6.1 Structured Semantic Graphs

Semantic formalisms have generated much interest and polarization within the field of natural language processing. Indeed, it is a challenging task to even specify a lexicon of predicate categories that offers broad coverage across domains as well as sufficient generalization for ease of annotation and statistical learning. Different projects approach the notion of semantics in many different ways ranging from formalisms based on propositional or first-order logic to lexical embeddings in Euclidean space. Here, we are particularly interested in structured representations of semantics that can contribute to high-level problems such as text generation.

The Propbank project (Palmer et al., 2005) eschews hand-crafted predicate lexicons by treating verb occurrences as lexicalized predicates. The arguments of a particular verb are identified according to their canonical positions in text featuring that sense of the verb and verb-argument relationships correspond closely to their syntactic modifiers. Broad coverage was achieved by annotating arguments for all verbs in a large corpus—specifically the Wall Street Journal section of the Penn Treebank—which has driven the development of statistical approaches to the problem of *semantic role labeling* (Carreras and Màrquez, 2004; Carreras and Màrquez, 2005).

In contrast, the FrameNet project (Baker et al., 1998; Fillmore et al., 2001; Fillmore et al., 2003) takes a hands-on approach in defining a frame taxonomy as well as argument labels semantics for frames and patterns of instantiation. Coverage is limited since the annotation

process was not driven by a corpus—indeed, early approaches to frame-semantic labeling relied only on the exemplar sentences in the FrameNet project, each of which was only annotated for a single frame and its arguments as opposed to all the frames present in the sentence. However, the richness of these representations and the eventual release of corpora with full frame-semantic annotations (Baker et al., 2007) resulted in steady research progress on the task of frame-semantic parsing. One such approach is SEMAFOR (Das et al., 2010; Das and Smith, 2011; Das et al., 2012; Das et al., 2013) which we employ to approximate gold-standard semantic parses in §6.1.4. An example of a SEMAFOR parse for our running example is provided in Figure 6.3.

The Abstract Meaning Representation (AMR) project (Banarescu et al., 2013) is a recent effort toward a full-sentence semantic specification and represents a sentence as a rooted DAG of concepts where the relations between concepts generalize Propbank-style arguments and incorporate coreference. AMR development is ongoing and we hope to apply the inference techniques described here on text-to-text generation experiments with AMR representations in the future. However, despite initial progress on parsing into AMR (Flanigan et al., 2014), we opt to use the relatively more mature Framenet resources for experiments in this chapter.

In addition to AMR, predicate-argument structures representing Propbank or Framenet parses can also be cast as DAGs over non-token labels—verb senses, frames, concepts, etc—and a subset of tokens representing the syntactic heads of arguments in an input sentence. DAG structures defined over all tokens are also of interest from a syntactic perspective: for instance, Kromann (2001) has proposed an expressive dependency formalism for Danish with explicit multi-headed constructions such as verb coordination, inspiring techniques for the task of maximum spanning DAG parsing (McDonald and Pereira, 2006; Sagae and Tsujii, 2008) which is known to be NP-hard (Schluter, 2014). The induction of richer syntactic representations and predicate-argument structure appears valuable for statistical approaches to text generation and serves to motivate an extension to our compression framework to support DAG structures in addition to paths and trees.

In this section, we first describe an ILP approach to recover the maximum-weight edge-factored DAG within a graph, thereby yielding a formulation which can be used for a variety

of semantic formalisms. We then empirically explore the question of whether semantic structures can be useful in sentence compression by extending our ILP formulation from §3.2 to recover a frame-semantic parse for the output sentence using the frames identified in the SEMAFOR parse of an input sentence.

6.1.1 Multi-structured objective

We begin by augmenting the ILP formulation from §3.2 to permit non-token nodes that can represent verb senses from Propbank or frames in the Framenet lexicon or concepts in AMR, etc. Without loss of generality, we define an expanded set of token-like nodes $\mathcal{T}^+ \triangleq \mathcal{T} \cup \mathcal{F}$ to include the n tokens from \mathcal{T} as well as ℓ abstract concept labels from $\mathcal{F} \triangleq \{t_i : t_i \text{ is an abstract concept labeled in } S, n+1 \leq i \leq n+\ell\}$. We also define an expanded set of binary variables x_i^+ and an incidence vector \mathbf{x}^+ which corresponds to the tokens and concepts in \mathcal{T}^+ that appear in an output compression C .

The set of possible directed semantic relationships in S is denoted by $\mathcal{U} \triangleq \{\langle i, j \rangle : t_i, t_j \in \mathcal{T}^+, t_j \text{ is an argument of } t_i \text{ in } S\}$. Note that this is a very general formulation of the problem in which semantic relations are permitted between tokens (e.g., coreference links), between concepts (as in AMR), from tokens to concepts (e.g., frame-evoking relations) as well as from concepts to tokens (for arguments). Although different formalisms and tasks will entail different subsets of these relations, we simply address the general problem of recovering the maximum-weight DAG connecting a subset of tokens from \mathcal{T}^+ .

We use $u_{ij} \in \{0, 1\}$ to represent a binary indicator variable indicating whether t_j is a direct argument of t_i in the semantic structure of the output sentence, and the corresponding incidence vector $\mathbf{u} \triangleq \langle u_{ij} \rangle_{\langle t_i, t_j \rangle \in \mathcal{U}}$ represents a subset of the semantic relations from \mathcal{U} . We can now update the objective from (3.3) to account for semantic features in the output compression \hat{C} .

$$\hat{C} = \arg \max_{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}} \mathbf{x}^\top \Delta_{\text{tok}} + \mathbf{y}^\top \Delta_{\text{ngr}} + \mathbf{z}^\top \Delta_{\text{dep}} + \mathbf{u}^\top \Delta_{\text{sem}} \quad (6.1)$$

where Δ_{sem} is a vector where each component corresponding to the indicator u_{ij} represents the feature-based score $\Delta_{\text{sem}}(s, i, j)$ for a semantic relation between t_i and t_j . Concept-specific scores can also be included in (6.1) by replacing \mathbf{x} with \mathbf{x}^+ and redefining the

corresponding scoring vector Δ_{tok} to account for concept nodes. We omit this aspect here for conciseness of notation although concept-specific features described in §6.2 are employed in our experiments.

6.1.2 Enforcing DAG structure

Assuming the n-gram configuration \mathbf{y} and dependency configuration \mathbf{z} are either omitted or appropriately addressed by the constraints from §3.3.2 and §3.3.1 respectively, we now attend to the problem of defining constraints to ensure that the directed graph structure specified by the semantic configuration \mathbf{u} represents a valid DAG and remains consistent with the token/concept configuration \mathbf{x}^+ . The following general conditions describe a DAG:

1. Every active node can have zero or more incoming and outgoing edges.
2. The structure has no directed cycles.

Even though the first condition implies that multiple roots can exist with no incoming edges, we can still opt to define an auxiliary $\text{ROOT} \in \mathcal{T}^+$ with outgoing edges to all nodes as this allows us to control the appearance of disconnected substructures like isolated concepts. However, there is no requirement for a constraint equivalent to (3.4) unless the semantic structure is required to have a single concept or token as the root of the DAG.

Although the semantic relations are relatively unconstrained, we must still ensure that they are consistent with the active tokens and concepts in \mathbf{x}^+ . This is accomplished by ensuring that a variable x_j^+ is active if any of its incoming or outgoing semantic relations are active.

$$\sum_i u_{ij} - \Upsilon_{\max} \cdot x_j^+ \leq 0, \quad \forall 1 \leq j \leq n + \ell \quad (6.2)$$

$$\sum_k u_{jk} - \Upsilon_{\max} \cdot x_j^+ \leq 0, \quad \forall 1 \leq j \leq n + \ell \quad (6.3)$$

where Υ_{\max} serves as an upper bound on the number of active incoming or outgoing semantic relationships³ for any token or concept in \mathcal{T}^+ . If no limit on branching is required, we can simply set $\Upsilon_{\max} = |\mathcal{T}^+| = n + \ell$.

³We use a single constant Υ_{\max} here for notational convenience but separate upper bounds can be imposed for incoming semantic relations in (6.2) and outgoing semantic relations in (6.3) if needed.

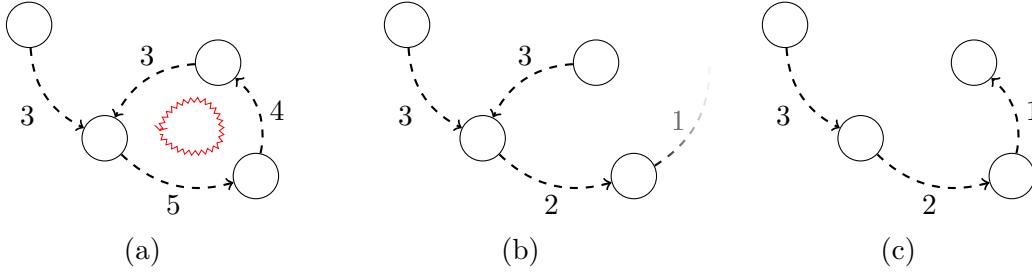


Figure 6.2: An illustrative flow network with edge weights indicating non-zero flow (a) permitting an undesired cycle when imposing the §3.2 constraint $\sum_i \gamma''_{ij} - \sum_k \gamma''_{jk} = x_j^+$ for all nodes, (b) & (c) constrained to acyclic structures when using revised constraints (6.6)–(6.7).

The above constraints do not address the prevention of directed cycles in \mathbf{u} . For this, we revisit the constrained flow network from §3.3.1 in conjunction with the directed acyclicity property from Lemma 3.3.1. New auxiliary variables γ''_{ij} are defined to carry some real-valued *relational* commodity between all pairs of tokens $\langle t_i, t_j \rangle$ where $t_i, t_j \in \mathcal{T}^+$ and $i \neq j$. The flow network is made to correspond to the active semantic configuration \mathbf{u} by constraining the u_{ij} variables to be zero whenever t_j is not an argument of t_i in the output sentence.

$$\gamma''_{ij} \geq 0, \quad \forall 1 \leq i \leq n + \ell, \quad (6.4)$$

$$1 \leq j \leq n + \ell, \quad i \neq j$$

$$\gamma''_{ij} - \Gamma''_{\max} u_{ij} \leq 0, \quad \forall 1 \leq i \leq n + \ell, \quad (6.5)$$

$$1 \leq j \leq n + \ell, \quad i \neq j$$

where Γ''_{\max} is the maximum amount of flow that the γ''_{ij} variables may carry and, as before, can be set to an arbitrary high value.⁴

In prior flow networks described in §3.3.1 and §3.3.2, active nodes consume a *single unit* of flow from their incoming flow variables and transmit the remainder to their outgoing flow variables, thereby establishing structural connectivity. As illustrated in Figure 6.2(a), this is not sufficient to prevent cycles when nodes can have multiple incoming inputs with corresponding positive flow. However, Lemma 3.3.1 shows that the output structure is

⁴Note that this value should generally be much higher than $|\mathcal{T}^+|$ unlike the limits on flow variables previously suggested in (3.22) because active tokens can consume a *variable* amount of flow due to the constraint in (6.6).

x_j^+	u_{ij}	Imposed constraint (6.6)	Effect
1	1	$\gamma_{ij}'' - \sum_k \gamma_{jk}'' \geq 1$	Flow must be consumed
1	0	$\Upsilon_{\max} \cdot \Gamma_{\max}'' - \sum_k \gamma_{jk}'' \geq 1$	Always satisfied
0	1	—	Forbidden by (6.2)
0	0	$\Upsilon_{\max} \cdot \Gamma_{\max}'' \geq 0$	Always satisfied

Table 6.1: Effective variant of constraint (6.6) imposed on the ILP for different binary states of a token/concept variable x_j^+ and its incoming semantic relation u_{ij} . The constraint on output flow is only imposed by active incoming semantic relations.

guaranteed to be acyclic if *every* input flow variable is larger than every output flow variable, i.e., flow always decreases when crossing an active node. This can be accomplished by replacing the usual per-token equality constraints for flow consumption with $n + \ell$ inequality constraints for each token or concept variable.

$$\gamma_{ij}'' + \Upsilon_{\max} \cdot \Gamma_{\max}'' (1 - u_{ij}) - \sum_k \gamma_{jk}'' \geq x_j^+, \quad \begin{array}{l} \forall 1 \leq i \leq n + \ell, \\ 1 \leq j \leq n + \ell, \\ i \neq j \end{array} \quad (6.6)$$

The effect of these constraints on the ILP is explored in Table 6.1. When x_j^+ is active, they ensure that *every* incoming flow variable γ_{ij}'' —and hence the minimum—corresponding to an active incoming relation u_{ij} remains larger than the *total* outgoing flow—and hence the maximum—from that node. If an inactive incoming relation is present (i.e., $u_{ij} = 0$), it has no corresponding flow γ_{ij}'' due to (6.5) but is nevertheless prevented from affecting output flow by the $\Upsilon_{\max} \cdot \Gamma_{\max}'' (1 - u_{ij})$ term, thereby ensuring that the constraint is not violated.

This latter scenario introduces a side effect: active nodes with no incoming active relations—and consequently zero incoming flow—have no restriction on output flow. In other words, any active node can become a root in the output DAG. This can be undesirable in some scenarios, e.g., an AMR parse is generally interpreted as a *rooted* DAG of concepts. In order to restrict potential roots in the output structure, we can reintroduce the notion that flow should only be created by special nodes and must simply be *transmitted*

by the remaining nodes.

$$\sum_i \gamma''_{ij} - \sum_k \gamma''_{jk} \geq 0, \quad \forall 1 \leq j \leq n + \ell \quad (6.7)$$

where the constraint is not applied to designated root nodes, represented here without loss of generality by a solitary ROOT denoted by t_0 . Note that specifying a single root also guarantees that \mathbf{u} will be connected.

The ILP can also be made compatible with an interpretation of a semantic parse as a maximum *spanning* DAG over \mathbf{x}^+ . This is accomplished by replacing the right hand side of constraint (6.7) with x_j^+ which ensures that active tokens in the output have positive incoming flow and consequently active incoming semantic relations. Furthermore, when $\ell = 0$ and a single root node is specified, the ILP yields dependency DAGs—of interest for both syntactic and semantic representations (McDonald and Pereira, 2006; Sagae and Tsujii, 2008; Schluter, 2014).⁵

By Lemma 3.3.1, an ILP featuring the constraints (6.2)–(6.7) or the extensions described above will produce an optimal DAG over the relations in \mathbf{u} and the tokens/concepts in \mathbf{x}^+ . These programs require at most $O((n + \ell)^2)$ variables and constraints for an input sentence with n tokens and ℓ potential concepts. While this may appear relatively compact, note that a large concept lexicon ($\ell \gg n$) would significantly increase the size of the program and consequently the running time for feature generation and inference with an ILP solver.

6.1.3 Constraining concept lexicons

The primary complication with maintaining a full-fledged semantic parser within an ILP for sentence compression is the relatively large lexicon of concept labels that must be accounted for. The Framenet lexicon, for instance, contains over 1,000 semantic frames and a naïve approach that includes each of these in compression problem would introduce a substantial

⁵Schluter (2014) suggests that edge-factored spanning DAGs over tokens—generalized by the ILP discussed here—may counter linguistic intuitions of semantics owing to two reasons: (i) edges do not constrain each other (ii) edge-factored maximization tends to encourage unwanted edges in a solution. However, this can be easily addressed in the ILP through constraints on the arguments of non-token concepts or higher-order hyperedges which are consistent with the edge-factored DAG. Although intractable in general, ILPs appear appropriate in this setting because the edge-factored problem is APX-hard (Schluter, 2014).

overhead for feature generation and inference. However, a number of simple pruning strategies can be used to limit the number of potential concepts that can be evoked in an output sentence.

As the sentence compression task explicitly mandates that the output sentence accurately reflect a portion—if not the totality—of the semantic information in the input sentence, one such strategy is the straightforward approach of merely restricting the potential concepts in the output to a subset of those in the input. The primary disadvantage of such an approach is an inherent sensitivity to errors in the semantic parser used to parse the input sentence. However, we might expect that concept identification errors are more local and less likely to propagate than errors in a syntactic parse.⁶ Moreover, in the absence of gold-standard semantic parses for compressions, the same parser would likely be used to generate semantic parses for reference compressions. Consequently, *consistency* in semantic parses across the input and compressed sentences is a crucial assumption for this task and one might expect that consistency in concept identification is relatively easy to achieve in an extractive compression context when output tokens are a subset of input tokens.

6.1.4 Preserving frame semantics in compression

Owing to the relative recency of AMR development and the observation that Propbank relations largely overlap with syntactic dependencies, we adopt the Framenet representation for initial experimentation with these ideas. We use the SEMAFOR parser (Das et al., 2013) to generate frame-semantic parses over both the input sentences and reference compressions, assuming that the latter approximates gold-standard compressed semantic structures. Following frame-semantic terminology, we hereby refer to concepts as *frames* and their arguments as *frame elements* or FEs which are realized in text via text spans termed *lexical units*. To examine the consistency of SEMAFOR on our extractive compression data, we examined how many frames and FEs that occur in the input sentence

⁶Although inter-predicate interactions can be considered by semantic parsers, concepts are closely related to their evoking words and phrases. Furthermore, since semantic parses are not restricted to cover all tokens in the sentence like syntactic parses, it appears less likely that an error in one concept label will propagate to the remaining output concepts.

	Frame overlap %	Frame reachability %	FE overlap %	FE reachability %
BN	97.54	77.73	84.86	35.91
WN	96.65	81.74	77.84	31.27

Table 6.2: Percentage of SEMAFOR frames and FEs from reference compressions which are present in input sentences alongside the percentage of *reachable* cases—references with frames or FEs entirely drawn from the input—over the BN and WN training corpora.

were also seen in reference compressions. Table 6.2 lists the results of this analysis on the training partitions of the BN and WN compression corpora described previously in §3.1.

Across both corpora, we observe that about 97% of the frames identified in reference compressions overlap with those identified in corresponding input sentences, thereby supporting our hypothesis that frame-semantic parsers like SEMAFOR are largely consistent in extractive compression scenarios. New frames were present in references of 196/880 BN instances, leading to relatively high *reachability* rates for frame configurations of 77.7% and 81.7% respectively. This analysis appears to support an approach in which the only frames considered for output compressions are those identified by SEMAFOR in the input sentence, with unseen frames ignored during training.

However, this consistency does not appear to extend to frame elements, even without considering their role labels. We see lower FE overlap rates of 77–85% on the two datasets with only 31–36% of reference compressions featuring FEs drawn entirely from input sentences. These numbers are partially affected by the equivalent rates for frames—all FEs of an unseen frame are naturally also considered unseen—but the presence of new FEs even in compressions with minor deletions is somewhat surprising. For this reason, we do not constrain output FEs by those in the input and instead consider all possible relations between frames and tokens in trying to recover an output frame-semantic parse.

As a motivating example for this approach, consider the SEMAFOR (Das et al., 2013) parses in Figure 6.3. Although the frame for the head verb in the input sentence is incorrect when contrasted with Figure 6.1, all relevant frames remain consistent in the SEMAFOR parse of a plausible compression. Note however that the FE vary from those in the input parse when the token configuration is altered, e.g., the frame element *Goal* is not present

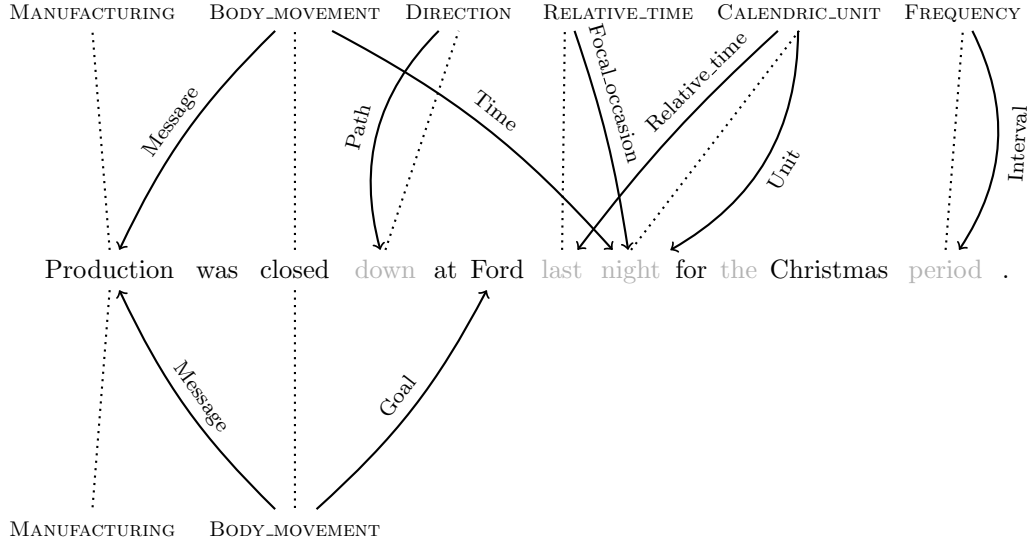


Figure 6.3: Frame-semantic relations produced by the SEMAFOR semantic parser (Das et al., 2013) over an example sentence and a possible compression. Dotted edges indicate the lexical units which evoke each frame while frame elements (FEs) are denoted by labeled edges to head words of the corresponding lexical units.

in the input sentence.

With an absence of coreference links and inter-frame relations, Framenet parses form a substantially simpler structure than AMR parses—they can be viewed as a bipartite graph of frames and tokens with edges representing potential argument relationships oriented in the same direction. Because this interpretation of a frame-semantic parse does not include directed cycles, compression over such structures merely requires the constraints (6.2)–(6.3) without a flow network to prevent cycles.

More specifically, x_i^+ now represents a token t_i in the sentence when $1 \leq i \leq n$ and a frame when $n < i \leq n + \ell$, which we denote henceforth by f_i .⁷ The semantic relation u_{ij} , $1 \leq i \leq n < j \leq n + \ell$ represents a frame element $\langle f_i, t_j \rangle$, i.e., a core, peripheral or extra-thematic argument of the frame f_i realized as a lexical unit in the sentence for which the rightmost head word is token t_j . The constraints defined in (6.2)–(6.3) thus ensure that an active FE indicator u_{ij} will activate indicators x_i^+ representing a frame f_i and x_j^+

⁷We abuse notation in keeping the subscripted index of a frame symbol f_i consistent with its indicator variable x_i^+ although this index only ranges over $n < i \leq n + \ell$ for frames.

representing t_j , the head of its lexical unit.

In addition, we introduce one further constraint to curb the occurrence of ungrounded frames in the output compression. All frames present in a sentence are evoked by a *Target* lexical unit, indicated by dotted lines in Figure 6.1 and Figure 6.3. To prevent the presence of frames with FEs but with no grounding in the output sentence, we enforce a restriction that an active frame must activate all tokens in its *Target* lexical unit.

$$x_i^+ \leq x_j^+, \quad \forall 1 \leq j \leq n < i \leq n + \ell \quad (6.8)$$

s.t. t_j is in the *Target* lexical unit of frame f_i

6.2 Features

We now describe the features that are used to expand the definition of $\Delta_{\text{tok}}(S, i)$ for $n < i \leq n + \ell$ to indicate the relevance of frames f_i from the input sentence to the output compression, as well as an additional linear scoring function $\Delta_{\text{sem}}(Si, j)$ to characterize potential frame elements $\langle f_i, t_j \rangle$ which relate these output frames to tokens in the compressed sentence.

6.2.1 Frame features

The feature templates used in the feature map $\phi_{\text{tok}}(S, i)$ for a frame f_i consist solely of the following indicators:

- The unique name of f_i along those of its parent and grandparent in the Framenet taxonomy.

We considered further ancestors in development experiments but found little benefit over this configuration although the performance variation observed was relatively subtle.

6.2.2 FE features

The feature templates used in the feature map $\phi_{\text{sem}}(S, i, j)$ for a frame element $\langle f_i, t_j \rangle$ consist of the following indicators:

- The role label of the FE conjoined with its *type* from the set $\{\text{CORE}, \text{PERIPHERAL}, \text{EXTRA-THEMATIC}\}$ as well as indicators for whether (a) t_j participates in a lexical

unit for any FE in the input sentence, (b) t_j is in the *Target* lexical unit responsible for evoking the frame f_i in the input sentence, and (c) a dependency relation exists between t_j and any token in the *Target* lexical unit for f_i in the input sentence.

In addition to these indicators, we experimented with richer features which conjoined frame names with FE labels and types as well combinations with syntactic features such as the POS tags of t_i and the label of any dependency edge between t_j and the target of f_i . However, these richer features did not lead to a significant and stable improvement over existing features in our tuning experiments over development datasets.

6.3 Experiments

We now consider an evaluation of multi-structured compression over frame-semantic structures over the BN and WN compression datasets (Clarke and Lapata, 2006b; Clarke and Lapata, 2007) described in §3.1. Our experimental environment is configured as described in §3.6. Chief among our goals is examining whether the addition of frame-semantic structure introduces any gain in compression quality. In addition, we are interested in analyzing the errors in compressed sentences in terms of their effect on output frame-semantic structures.

6.3.1 Compression quality

In this section, we consider the performance of the following inference approaches based on dynamic programming:

- **ILP-2gr-sem**: The ILP described in §3.3.2 for recovering a bigram-factored compressed sentence combined with the proposed extension in §6.1.4 for inferring corresponding frame-semantic structures.
- **ILP-dep-sem**: The ILP described in §3.3.1 for inferring edge-factored compressed dependency trees combined with the proposed extension in §6.1.4 for inferring corresponding frame-semantic structures.
- **ILP-2gr-dep-sem**: The full ILP which combines the constraints from §3.3.2 for producing a bigram-factored compression, §3.3.1 for producing an edge-factored depen-

BN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		path	DP-2gr (McD06)	82.94	72.84	61.08	52.65	-
path + DAG	ILP-2gr-sem	82.86	72.70	60.62	52.07	-	70.53	66.58
tree	ILP-dep	82.70	70.05	56.81	47.94	75.76	70.88	65.25
tree + DAG	ILP-dep-sem	82.36	69.33	55.97	47.11	75.30	71.13	63.95
path + tree	ILP-2gr-dep	82.85	72.66	60.87	52.87	75.73	73.82	68.56
path + tree + DAG	ILP-2gr-dep-sem	82.77	72.74	61.14	53.17	75.73	73.93	68.67

Table 6.3: Experimental results for the BN corpus with all systems restricted to compress to the size of the median reference compression. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

dependency tree and §6.1.4 for inducing a corresponding frame-semantic structure over the compressed sentence.

New models are trained for these techniques using the minibatched structured perceptron as described in §3.5. These approaches are compared on measures of output compression quality against the following systems previously described in §3.6.1:

- **DP-2gr:** The dynamic program of McDonald (2006) described in §4.1.2 for inference of bigram-factored compressions under a compression rate.
- **ILP-dep:** The ILP described in §3.3.1 for inferring edge-factored compressed dependency trees.
- **ILP-2gr-dep:** The full ILP for multi-structured inference which combines the constraints from §3.3.1 to produce an edge-factored dependency tree with §3.3.2 to yield a bigram-factored token ordering for the output sentence.

Table 6.3 lists measures of compression quality for these systems over the test partition of the BN corpus. Most noticeable is the observation that including frame-semantic structures in the compression objective does not appear to improve results significantly on any of the measures considered. Although minor gains are observed for the most expressive system **ILP-2gr-dep-sem**, these results remain in lockstep with **ILP-2gr-dep** when statistical

WN		F ₁ % for n-grams of order				Syntactic relations F ₁ %		
Structures	Inference	1	2	3	4	z	Stanford	RASP
		path	DP-2gr (McD06)	75.36	63.40	52.15	42.97	-
path + DAG	ILP-2gr-sem	75.21	63.22	52.02	42.78	-	62.55	59.03
tree	ILP-dep	75.74	60.07	46.84	37.48	69.08	64.33	58.49
tree + DAG	ILP-dep-sem	75.31	59.20	45.58	36.05	68.52	62.90	57.71
path + tree	ILP-2gr-dep	76.15	64.47	53.63	45.15	69.52	67.35	61.82
path + tree + DAG	ILP-2gr-dep-sem	75.88	63.71	52.71	43.87	69.19	66.65	61.92

Table 6.4: Experimental results for the WN corpus with all systems restricted to compress to the size of the reference compression. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

significance is considered. Moreover, `ILP-dep-sem` is noticeably weaker than `ILP-dep` in n-gram and RASP F₁ measures, giving rise to a conjecture that the scores derived from frame-semantic structures may conflict with those drawn from dependency structures.

Table 6.4 contains the corresponding numbers over the WN test dataset. As with the BN dataset, the frame-semantic approaches do not yield significant improvements over the equivalent bigram and dependency compression approaches. In contrast, adding frame-semantic structures appears to diminish performance significantly on some measures such as trigram and 4-gram F₁ for `ILP-2gr-dep-sem` when compared to `ILP-2gr-dep`, with the performance drop appearing most consistent when comparing the dependency-based approach `ILP-dep-sem` with `ILP-dep`. Note however that this drop in performance may be partially explained by our decision to favor the BN corpus in tuning experiments for feature selection owing to its lower average inference time (cf. Table 5.4) and multiple references.

Why do these more expressive models fail to produce significant improvements over the compression data? Although the results were not unexpected given the challenge of identifying a compelling set of features for frame and FE variables in tuning experiments, we might reasonably expect that these richer models would offer moderate gains over approaches that do not consider frame semantics. We suspect that the reasons for this result are five-fold:

1. These models are likely hampered by a small training corpora that cannot adequately exploit higher-order models as shown by our experiments with supervised trigrams in §3.6.5 and with second-order dependencies in §5.3.3. Sparsity issues may therefore be responsible for poor parameterizations of frame and FE features.
2. In addition, the utility of these system-generated representations is naturally limited by their robustness on unseen text and frame-semantic parsing is particularly challenged in this respect by a setting of greater ambiguity than dependency parsing,⁸ visible in the inconsistency of frame elements between input sentences and their reference compressions shown in Table 6.2.
3. Furthermore, unlike n-grams and dependency edges which are closely tied to the surface realization of a sentence, frame-semantic parses are more abstract and, consequently, the identification of frames and FEs is not necessarily aligned with the problem of finding good surface realizations for sentence compression. For example, only 21% of the SEMAFOR-identified frames in the BN and WN training datasets have FEs; the rest offer little benefit⁹ for these problems as they do not establish codependencies between tokens or enforce a global structure but are nevertheless considered in the inference objective.
4. This also points to the limitation of our structured interpretation of frame-semantic parses, i.e., that they do not involve inter-frame relations. Although these representations adhere closely to SEMAFOR output, alternative interpretations which extend FEs to link inter-related frames may lead to different results.
5. Finally, the extractive compression task fails to take advantage of a key aspect of this representation, namely the use of frames as abstractions over predicates with varying surface realizations (e.g., predicates *buy* and *sell* can both be realized under COMMERCE_SELL) and these representations might indeed be more valuable in tasks which involve paraphrasing.

⁸Indeed, since SEMAFOR relies on a dependency parse of a sentence as its input, its performance is strictly circumscribed by parsing errors.

⁹Regardless of the presence of FEs, including these variables in the inference objective is equivalent to introducing new features which indicate the frames that are evoked by each token.

BN							
Structures	Inference	Frames			FEs		
		P%	R%	F ₁ %	P%	R%	F ₁ %
path + DAG	ILP-2gr-sem	81.25	79.31	79.53	36.99	18.36	21.00
tree + DAG	ILP-dep-sem	80.32	81.75	80.16	43.56	21.42	24.99
path + tree + DAG	ILP-2gr-dep-sem	81.12	81.06	80.18	41.37	19.92	23.43

Table 6.5: Precision and recall of frames and frame elements (FEs) with respect to the reference compressions for the BN corpus. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

WN							
Structures	Inference	Frames			FEs		
		P%	R%	F ₁ %	P%	R%	F ₁ %
path + DAG	ILP-2gr-sem	73.75	74.82	73.10	28.45	8.33	11.14
tree + DAG	ILP-dep-sem	74.31	77.81	74.67	29.47	8.19	11.42
path + tree + DAG	ILP-2gr-dep-sem	74.90	77.16	74.75	29.24	8.53	11.43

Table 6.6: Precision and recall of frames and frame elements (FEs) with respect to the reference compression for the WN corpus. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

6.3.2 Frame-semantic integrity

Although the addition of frame-semantic structures does not appear to help the multi-structured compression framework, we are interested in whether the degree to which it yields accurate frame-semantic representations on output text. Table 6.5 lists the precision, recall and F₁ for frames and FEs from the prior experiments over the BN test dataset while Table 6.6 contains the corresponding results over the WN dataset.

With respect to frames, the most readily apparent trend across both datasets is the relatively small variation in performance. Most differences are not statistically significant other than a weaker recall rate for frames when the dependency-based perspective is excluded from the model. We attribute this to a decrease in the number of inferred frames in the output when relying on a models that discourages the presence of isolated frame-evoking lexical units without strong adjacency relationships.

Turning to the FEs, we observe that all absolute measures are drastically lowered. Whereas the precision and recall of frames were balanced and relatively strong, FE precision is more than twice FE recall for the BN corpus and more than 3.5 times FE recall for the WN corpus but FE identification remains half as precise as frame identification across both datasets. Most interestingly, peak FE F_1 is achieved with `ILP-dep-sem` on both datasets although this was observed to be the weakest of the frame-semantic models in the prior evaluation. We interpret this phenomenon as evidence that FE identification is stronger when output tokens are less constrained—most true of the dependency-only compression model—but that these relations do not capture adequate information on whether their potential lexical units tokens should be removed or preserved under compression—a property that is generally stronger in the bigram and joint models.

6.4 Remarks

The chief contribution of this work is an extension of the flow-based ILP formulation from Chapter 3 to recover various forms of maximum-weight DAGs in graphs, including *rooted* DAGs such as AMR graphs and *spanning* DAGs for augmented syntactic representations. In general, these formulations can be incorporated in multi-structured inference models to represent predicate-argument structures which capture higher-level semantic relations than are normally made explicit through syntactic parse trees.

We also observe that a straightforward application of our multi-structured inference formulation to frame-semantic structures using SEMAFOR yields no significant advantage in our experiments. Although we expect that the use of larger compression corpora such as Filippova and Altun (2013) will yield better features and more robust parameters for these models, it remains an open question as to whether the limitations of the relatively abstract frame-semantic formalism and the ambiguity inherent in parsing these structures can be sufficiently controlled in order to yield performance gains on the compression task.

Regardless of this, we are keen to experiment with the full DAG-recovering ILP for the recent AMR formalism, which appears to address many of the deficiencies that we encountered with Framenet. AMR parses are relatively close to surface lexicalizations and explic-

itly capture many linguistic phenomena that are not addressed by syntactic representations such as the use of re-entrant edges for coreferent entities. Moreover, the development of AMR is explicitly data-driven and motivated by machine translation applications, thereby prioritizing generality and broad coverage over depth within a particular domain. These aspects suggest a greater degree of orthogonality for AMR structures with respect to syntactic representations, making the formalism a particularly interesting candidate for our multi-structured techniques.

Other potential applications are suggested by our generalization of flow-based inference to DAGs. For instance, spanning DAGs may be used to represent rich syntactic or semantic representations (McDonald and Pereira, 2006; Sagae and Tsujii, 2008; Schluter, 2014) as well as a forest of k -best edge-factored dependency parses. A further (straightforward) generalization to arbitrary connected graphs can be used to directly target the RASP structures used for evaluation in these experiments. These techniques therefore hold promise for task-based evaluations of syntactic and semantic representations and toolkits in the context of text-to-text generation problems.

Chapter 7

Multi-Structured Sentence Fusion

A primary application for sentence compression is in the high-level task of *abstractive summarization*, long seen as an important problem in natural language processing. In an analysis of manual approaches to summarization, Jing and McKeown (2000) showed that humans frequently reduce the size of their summaries by compressing sentences as well as *merging* them. In the literature, the latter task is generally termed sentence *fusion* and broadly defined as the problem of combining two or more sentences which overlap in information content, avoiding extraneous details and preserving common information. In addition to its role in human summarization, sentence fusion has been shown to be a valuable component of automated summarization systems (Barzilay and McKeown, 2005). However, although recent years have seen steady progress in single-sentence tasks such as sentence compression and paraphrase generation, research in sentence fusion has long been hampered by the absence of datasets for the task.

This chapter presents a new fusion dataset generated from existing human annotations and explores extensions of the single sentence compression approach of Chapter 3 to n-way sentence fusion.¹ Our fusion dataset is constructed from evaluation data for summarization shared tasks in the Document Understanding Conference (DUC)² and the Text Anal-

¹An early version of this work was presented in Thadani and McKeown (2013b). We have also explored the topic of evaluation for intersection-based fusion in Thadani and McKeown (2011b).

²Document Understanding Conference (DUC) resources: <http://duc.nist.gov>

ysis Conference (TAC).³ We use human-generated annotations produced for the pyramid method (Nenkova et al., 2007) for summarization evaluation to produce a dataset of natural human fusions with quantifiable agreement. This offers advantages over previous datasets used for standalone English sentence fusion which contain annotator-induced noise (McKeown et al., 2010) or cannot be freely distributed to researchers (Elsner and Santhanam, 2011). In addition, each of these datasets contains around 300 instances of fusion while the new dataset presented here contains 1860 instances and can be further expanded by relaxing the filters used in its construction.

Crucially, this larger corpus encourages supervised approaches to sentence fusion, thereby enabling experimentation with multi-structured inference strategies for the task. Previous approaches to fusion have generally relied on variations of dependency graph combination (Barzilay and McKeown, 2005; Filippova and Strube, 2008b; Elsner and Santhanam, 2011) for content selection with a separate step for linearization that is usually based on an LM. However, as the flow-based ILP for multi-structured compression from §3.2 is capable of simultaneously identifying a dependency tree and its linearization, we adapt it to study its utility in discriminative sentence fusion. Furthermore, we can also incorporate relevance cues across input sentences through additional features and constraints that encourage redundant input information to appear in the output fusion.

The primary contributions of this chapter are:

- A novel dataset of natural sentence fusions drawn from pyramid evaluation datasets for summarization which are available to the NLP community.
- A supervised approach to sentence fusion that jointly addresses non-redundant content selection and linearization.
- An extension of the dependency-based subproblem from §3.3.1 which incorporates the orientation of output dependency edges in the solution.

³Text Analysis Conference (TAC) resources: <http://www.nist.gov/tac>

1	In 1991, the independents claimed nearly a third of adult book purchases but six years later their market share was nearly cut in half, down to 17% .
2	By 1999, independent booksellers held only a 17 percent market share.
SCU	Six years later independent booksellers' market share was down to 17%

1	The heavy-metal group Metallica filed a federal lawsuit in 2000 against Napster for copyright infringement , charging that Napster encouraged users to trade copyrighted material without the band's permission.
2	The heavy metal rock band Metallica , rap artist Dr. Dre and the RIAA have sued Napster , developer of Internet sharing software, alleging the software enables the acquisition of copyrighted music without permission.
3	The heavy-metal band Metallica sued Napster and three universities for copyright infringement and racketeering, seeking \$10 million in damages.
SCU	Metallica sued Napster for copyright infringement

1	The government was to pardon 23 FARC members as the two sides negotiate prisoner exchanges.
2	The Columbian government plans to pardon more than 30 members of FARC as they negotiate a prisoner swap.
3	The government and FARC continued to argue over details of a prisoner swap.
SCU	The government and FARC negotiate prisoner exchanges

Table 7.1: SCU annotations drawn from DUC 2005–2007 and TAC 2008–2011. Human-annotated contributors to the SCU are indicated as boldfaced spans within the respective source sentences.

7.1 Pyramid Fusion Corpus

The pyramid method is a technique for summarization evaluation that aims to quantify the semantic content of summaries and compare automated summaries to human summaries on the basis of this semantic content (Nenkova et al., 2007). First, for each summarization topic to be evaluated, multiple human-authored summaries are produced—4 for most DUC and TAC evaluations although DUC 2005 features 7 summaries per topic.⁴ Annotators then

⁴We also noted one document with 5 human summaries in TAC 2010.

identify *summarization content units* or SCUs—intended to correspond to atomic units of information—by reading and comparing these summaries.⁵ Each SCU contains a *label* which is a concise English sentence stating its semantic meaning and a list of *contributors* which are discontinuous character spans from the summary sentences—hereafter referred to as *source* sentences—in which that SCU is realized. Table 7.1 contains examples of SCUs drawn from DUC 2005–2007 and TAC 2008–2011 data.

Our fusion corpus is constructed by taking the source sentences of an SCU as input and the SCU labels as the gold-standard output fusion of these sentences. The fusion task captured by this corpus is similar to sentence *intersection* as defined by Marsi and Krahmer (2005) although it does not fit the criteria for *strict* intersection as addressed in Thadani and McKeown (2011b) because source sentences may not expressly mention all the information that is contained in an SCU label due to unresolved anaphora and entailments made using external knowledge.

The following procedure was used to extract a corpus of fusion instances from the 17756 SCUs annotated in the DUC and TAC datasets.

1. We first choose to ignore all contributor sentences which are sourced from more than one summary sentence, thereby ignoring higher-level concepts which straddle multiple sentences. Dropping these contributors also reduces the dataset by 323 SCUs which only feature multi-sentence contributors.
2. Naturally, SCUs that have only a single contributor are not useful for the fusion task. These comprise a majority (55%) of SCUs in the corpus and dropping them leaves 7845 SCUs for further consideration.
3. In addition, we chose to restrict the number of input sentences to at most 4 since SCUs with 5–7 contributors are present only in DUC 2005 and are thus fairly infrequent in the dataset. This is accomplished by iteratively removing the contributors of these SCUs which share the fewest words with the SCU label until only 4 remain.
4. Although SCU descriptions are required to be full sentences, we found that this was not upheld in practice. We therefore removed SCUs whose labels contain fewer than

⁵An SCU annotation guide from DUC 2005 is available at <http://www1.cs.columbia.edu/~ani/DUC2005/AnnotationGuide.htm>.

5 words and did not have an identifiable verb beyond the first token. As a practical consideration, SCUs with source sentences which have more than 100 tokens were also dropped. Applying these filters leaves 6453 fusion candidates in the corpus.

5. Crucially, annotated concepts in this dataset often only cover a small fraction of source sentences and may not represent the full overlap between them. To account for this, we ignored SCUs without contributors that are at least half the length of their source sentences as well as SCUs whose labels are less than half the length of the smallest contributor. Imposing these constraints preserves 4628 SCUs in the dataset.
6. Finally, we chose to retain only SCUs whose labels contain terms present in at least one source sentence, thus ensuring that the SCUs which are presumed to be reference fusions are entirely reachable without paraphrasing.

This procedure yields 1860 fusion instances of which 873 have two inputs, 569 have three and 418 have four.⁶ Figure 7.1 shows the number of instances of each type with respect to the DUC or TAC dataset they are drawn from. The fraction of SCUs retained from each dataset is fairly consistent and ranges from 8.5% (DUC 2005) to 11.8% (TAC 2009). Naturally, the size and quality of the dataset can be varied by adjusting the hyperparameters used in the filtering procedure.

7.2 Multi-Structured Fusion

Prior approaches to fusion have often involved multiple stages. A content selection phase typically combines the dependency graphs of input sentences to produce an intermediate syntactic representation of the information in the sentence (Barzilay and McKeown, 2005; Filippova and Strube, 2008b; Elsner and Santhanam, 2011). Linearization of output fusions is usually performed by ranking hypotheses with an LM—sometimes with language-specific heuristics to filter out ill-formed sentences. This pipelined strategy is also known as *overgenerate-and-rank* and is often found to be a source of errors in fusion problems (Barzilay and McKeown, 2005).

⁶We hope to eventually distribute the extracted corpus directly but interested researchers can currently retrieve the raw data from NIST and reconstruct it from our guidelines.

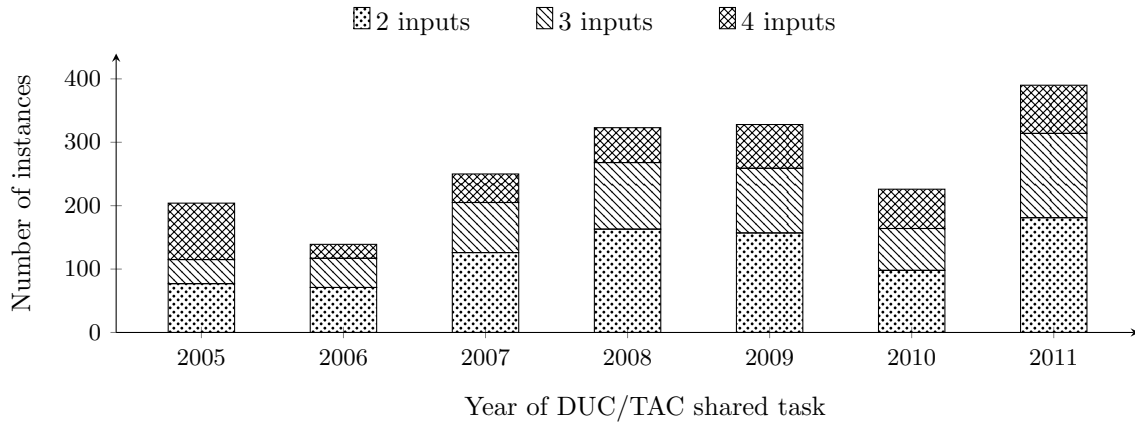


Figure 7.1: Distribution of instances in the pyramid fusion corpus constructed according to §7.1 with respect to the DUC or TAC dataset that they were drawn from.

The multi-structured sentence compression approach discussed previously already addresses both these problems by simultaneously producing a dependency tree and an n-gram factored linearization rather than relying on pipelined stages to first select output content and then linearize an intermediate dependency representation. Furthermore, the presence of a relatively large dataset enables supervised approaches in which likelihood under an LM is one of many features of output quality. Generic fusion—without a specific query or restriction for output content—is similar to the sentence compression task albeit with multiple input sentences; indeed, recent literature has eschewed the term *fusion* in favor of *multi-sentence compression* (Filippova, 2010; Boudin and Morin, 2013; Tzouridis et al., 2014). The remainder of this section discusses an adaptation of the multi-structured compression approach from §3.2 to multiple input sentences. In addition, we introduce new features and constraints that directly address the unique aspects of the fusion task.

7.2.1 ILP formulation

Consider a single fusion instance involving p input sentences $\mathcal{S} \triangleq \{S_1, \dots, S_p\}$ and let F denote their fusion. Assuming a linear model for inference, we formulate the MAP inference

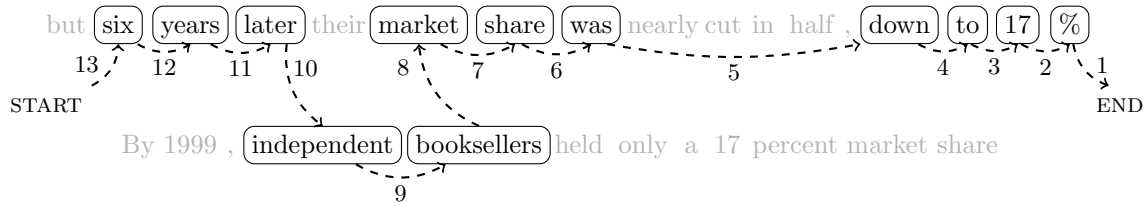


Figure 7.2: An illustration of adjacency flow values for a fusion of two input sentences using the ILP from §3.3.2. Dashed lines denote all non-zero flow variables.

objective for the fusion task to follow (3.2) for sentence compression.

$$\begin{aligned} \hat{F} &\triangleq \arg \max_F \Delta(\mathcal{S}, F) \\ &= \arg \max_F \boldsymbol{\theta}^\top \phi(\mathcal{S}, F) \end{aligned} \quad (7.1)$$

where the scoring function Δ , feature map ϕ and learned parameters $\boldsymbol{\theta}$ apply to the fusion task. If $\mathcal{T}_r \triangleq \{t_i^r : i\text{th token in } S_r \in \mathcal{S}\}$ and $\mathcal{T} \triangleq \bigcup_r \mathcal{T}_r$, the optimal output fusion \hat{F} can be recovered by an ILP solver which targets the joint objective from (3.3) under structural constraints.

$$\hat{F} = \arg \max_{\mathbf{x}, \mathbf{y}, \mathbf{z}} \mathbf{x}^\top \boldsymbol{\Delta}_{\text{tok}} + \mathbf{y}^\top \boldsymbol{\Delta}_{\text{ngr}} + \mathbf{z}^\top \boldsymbol{\Delta}_{\text{dep}} \quad (7.2)$$

where a token configuration \mathbf{x} , a path of bigrams \mathbf{y} and an edge-factored dependency tree \mathbf{z} define the highest-scoring output sentence \hat{F} . Valid structural configurations for \mathbf{y} and \mathbf{z} are ensured by using the flow network formulations from §3.2, which are defined over generic token graphs and require no modification to work with multiple input sentences.⁷ For example, Figure 7.2 illustrates the path-structured adjacency flow that would accompany an n-gram solution extracted from two input sentences using the constraints from §3.3.2.

7.2.2 Redundancy

Although the ILP for compression yields non-degenerate structures for output sentences, novel issues arise as a consequence of having multiple input sentences. One such issue is that of *redundancy* in the input text. Table 7.2 illustrates the utility of redundancy as a

⁷DAGs that specify semantic parses can also be retrieved by adopting the expanded objective (6.1) and corresponding structural constraints from §6.1.

1	The <u>heavy-metal</u> group <u>Metallica</u> filed a federal lawsuit in 2000 against <u>Napster</u> for <u>copyright infringement</u> , charging that <u>Napster</u> encouraged users to trade <u>copyrighted</u> material without the <u>band's</u> permission.
2	The <u>heavy metal</u> rock <u>band</u> <u>Metallica</u> , rap artist Dr. Dre and the RIAA have <u>sued</u> <u>Napster</u> , developer of Internet sharing software, alleging the software enables the acquisition of <u>copyrighted</u> music without permission.
3	The <u>heavy-metal</u> <u>band</u> <u>Metallica</u> <u>sued</u> <u>Napster</u> and three universities for <u>copyright infringement</u> and racketeering, seeking \$10 million in damages.
Fusion	<u>Metallica</u> <u>sued</u> <u>Napster</u> for <u>copyright infringement</u>

1	The <u>government</u> was to <u>pardon</u> <u>23</u> <u>FARC</u> <u>members</u> as the two sides <u>negotiate</u> <u>prisoner</u> exchanges.
2	The <u>Columbian government</u> plans to <u>pardon</u> more than <u>30</u> <u>members</u> of <u>FARC</u> as they <u>negotiate</u> a <u>prisoner</u> <u>swap</u> .
3	The <u>government</u> and <u>FARC</u> continued to argue over details of a <u>prisoner</u> <u>swap</u> .
Fusion	The <u>government</u> and <u>FARC</u> <u>negotiate</u> <u>prisoner</u> exchanges

Table 7.2: An illustration of lexical redundancy in fusion instances from our corpus. Dashed underlining indicates content words—nouns, verbs, adjectives and adverbs—whose stems occur in two input sentences while solid underlining indicates those which occur in all three.

cue for salience in fusion examples drawn from the pyramid dataset. In the first example, every content word in the output fusion appears in at least two input sentences and most appear in three, even without considering paraphrases such as *filed a ... lawsuit* \Leftrightarrow *sued*. In addition, most redundant words in the input are present in the output except the phrase *heavy-metal band*, whose inclusion would not be out of place in a good fusion. The second example similarly demonstrates the importance of redundancy, particularly if a paraphrase dictionary can identify the synonymy of *exchange* \Leftrightarrow *swap*.

Although redundancy in the input sentences yields a potential indication of information salience, redundancy in an output sentence is undesirable. We address both these aspects: the former by identifying groups of similar content words across sentences for additional salience features and the latter through constraints over these groups.

7.2.2.1 Identifying input redundancy

We consider every pair of input sentences in \mathcal{S} and identify groups $\{\mathcal{G}_1, \dots, \mathcal{G}_a\}$ of similar or identical open-class words—nouns, verbs, adjectives and adverbs. Word similarity is established via stem/lemma overlap, Wordnet synonymy⁸ and abbreviation fitting. The word groups are assumed to be closed under transitivity⁹ and are identified using a single-link agglomerative clustering procedure which takes $O(n^2)$ time where $n = \max_r |\mathcal{T}_r|$. Each group \mathcal{G}_c is further partitioned into subgroups of similar tokens from each sentence, i.e., $\mathcal{G}_c \triangleq \{\mathcal{H}_c^r : \mathcal{H}_c^r \subseteq \mathcal{T}_r, 1 \leq r \leq |\mathcal{S}|\}$ where a subgroup \mathcal{H}_c^r consists of the matched tokens from input sentence S_r . In the first example from Table 7.2, this procedure would identify a group $\mathcal{G} = \{\mathcal{H}^1, \mathcal{H}^2, \mathcal{H}^3\}$ for the lemma *copyright* where the subgroups $\mathcal{H}^1 = \{\textit{copyright}, \textit{copyrighted}\}$, $\mathcal{H}^2 = \{\textit{copyrighted}\}$ and $\mathcal{H}^3 = \{\textit{copyright}\}$ contain instantiations of the lemma in each input sentence.

As previously discussed, we might expect that content words from large groups are more likely to appear in an output fusion owing to their support from multiple input sentences. We therefore define the *support* of a token $\Omega : t_i^r \rightarrow \mathbb{N}$ as the number of sentences which contain words that match t_i^r .

$$\Omega(t_i^r) \begin{cases} = |\mathcal{G}_c|, & \exists \mathcal{H}_c^r \in \mathcal{G}_c \text{ s.t. } t_i^r \in \mathcal{H}_c^r \\ = 0, & \text{otherwise} \end{cases} \quad (7.3)$$

In §7.3.2, we describe features that allow the structured learner to exploit this measure of support as an indication of salience for n-grams and dependency edges.

7.2.2.2 Minimizing output redundancy

While we expect largely positive weights on features for supporting tokens, this may also have the effect of encouraging more than one token from the *same* group to occur in the output. In order to avoid this problem, we can simply add a constraint for each group that prevents all tokens within a group from appearing more than once in an ILP solution. In

⁸Specifically, whether either word shares at least a third of its Wordnet synsets with the other word.

⁹In other words, if a word is found to match with two groups, the groups are merged. Any input token will therefore participate in at most one group.

practice, however, we impose less stringent constraints which restrict each token in a group to appear in the solution no more than the number of times *any* token in that group appears in an input sentence. If we use x_i^r to denote the indicator variable representing the presence of token t_i^r from sentence S_r in the output fusion, these constraints can be expressed as

$$\sum_{\substack{i,r,c: \\ t_i^r \in \mathcal{H}_c^r \in \mathcal{G}_c}} x_i^r \leq \max_{\mathcal{H}_c^r \in \mathcal{G}_c} |\mathcal{H}_c^r|, \quad \forall c \quad (7.4)$$

7.2.3 Dependency orientation

One aspect of dependency tree inference which we have not discussed thus far is the *orientation* of edges in the output sentence. The ILP from §3.2 defines dependency edges z_{ij} with no consideration to the relative positions of the tokens t_i and t_j in the solution. Consequently, z_{ij} is scored identically regardless of whether it manifests as a right attachment or a left attachment in the output compression. Note that the participating tokens t_i and t_j might be drawn from different input sentences for the fusion task but, for clarity of exposition, we omit sentence indices in this section and build on the notation from §3.2.

Although an invariance to dependency orientation is true for compression as well as fusion, the issue is less significant in the case of the former. Since the dataset for compression experiments from §3.1 does not include token reordering, the orientation of a dependency edge between any two tokens remains fixed, i.e., if $i < j$, $t_i \rightarrow t_j$ is always a right attachment and $t_j \rightarrow t_i$ is always a left attachment. Information on the direction of dependencies is therefore easily incorporated into features which can consider the relative positions of the two words in the input sentence.

However, ordering decisions are unavoidable for inference in fusion problems and positional features cannot always account for edges between tokens from different sentences. This scenario is somewhat problematic since it does not distinguish between dependency relationships which are closely tied to the direction of attachment such as *subject* and *object*, e.g., the dependency parses of *dog bites man* and *man bites dog* would be scored identically. Without the ability to specify direction-aware features for dependency edges, the trees produced by the dependency variables may be ineffective in constraining other output structures.

Conveniently, the flow formulation in §3.3.2 provides a solution. Although actual positional information is not captured in any structural indicator variables, observe that the adjacency flow variables γ'_{ij} implicitly encode the position of every active token in the output sentence.¹⁰ If the total value of outgoing adjacency flow from token indicator x_i is greater than the outgoing adjacency flow from x_j , t_i must precede t_j in the output and vice versa. We can exploit this observation for additional constraints that forbid inconsistent scoring of left and right attachments.¹¹

Consider now *two* distinct types of dependency indicator variables: z_{ij}^R indicates a right attachment in which the governor t_i precedes the dependent t_j in \widehat{F} and z_{ij}^L indicates a left attachment in which t_i follows t_j in \widehat{F} . These orientation-aware variables can be incorporated in the objective function from (7.2) by redefining $\mathbf{z} \triangleq \langle \mathbf{z}^L, \mathbf{z}^R \rangle$. The score for any dependency relationship $\Delta_{\text{dep}}(\mathcal{S}, i, j)$ with dependent t_j governed by t_i can now be replaced by an orientation-aware scoring function $\Delta'_{\text{dep}}(\mathcal{S}, i, j, o)$ where $o \in \{\text{L}, \text{R}\}$ depends on whether the dependency is expressed through z_{ij}^L or z_{ij}^R .

We now require that these indicators remain consistent with the relative positions of t_i and t_j in an output sentence hypothesis. The constraints below forbid a left attachment z_{ij}^L from becoming active when the difference between the outgoing flow from t_i is greater than that from t_j , i.e., when t_i precedes t_j in the output sentence. Similarly, a right attachment z_{ij}^R is prevented from becoming active when t_j precedes t_i .

$$\sum_k \gamma'_{ik} - \sum_l \gamma'_{jl} \leq \Gamma'_{\max}(1 - z_{ij}^L), \quad \forall 0 \leq i \leq n, \quad (7.5)$$

$$1 \leq j \leq n, \quad i \neq j$$

$$\sum_l \gamma'_{jl} - \sum_k \gamma'_{ik} \leq \Gamma'_{\max}(1 - z_{ij}^R), \quad \forall 0 \leq i \leq n, \quad (7.6)$$

$$1 \leq j \leq n, \quad i \neq j$$

¹⁰Specifically, the total outgoing flow of adjacency commodity $\sum_k \gamma'_{ik}$ from any active token variable t_i is always equal to $m - p_i + 1$ where $p_i \in \{1, \dots, m\}$ is the position of t_i in an output sentence.

¹¹Beyond attachment direction, we can also exploit adjacency flow variables to define $O(n^2)$ variables that correspond to the *range* of some active dependency, i.e., the number of tokens between its governor and dependent. These non-binary variables can then be explicitly constrained—for instance, to prevent long-range dependencies in the output—or can be scored by a feature-based scoring function and directly incorporated into the output objective. We experimented briefly with the latter approach in the context of the compression task from Chapter 3 but initial results have not shown promise.

Because any difference between positive flow variables γ' is always less than Γ'_{\max} , the constraints (7.5)–(7.6) have no effect on the program when the respective dependency variable is not active.

7.3 Features

We largely base our features for the fusion task on the features for compression problems previously described in §3.4. The changes introduced for fusion problems include a reduced emphasis on lexical features and the inclusion of feature templates based on token *support* as a signal of salience for n-grams and dependency edges in the output sentence. As before, we record both absolute and normalized versions for scale-dependent features—normalizing them by the average length of input sentences in an instance—in order to encourage the models to be robust to variations in the size of a fusion problem.

7.3.1 Token features

We use a reduced set of token features when adapting the compression approach of Chapter 3 to fusion problems, excluding lexical features and POS context features with large spans which we found led more readily to overfitting. The full set of feature templates used in the feature map $\phi_{\text{tok}}(\mathcal{S}, r, i)$ for a token t_i^r consist of the following lexico-syntactic indicators:

- The POS of t_i^r conjoined with the label of the dependency edge incident on t_i^r in the Stanford dependency parse of the input sentence.
- The POS tag of t_j^r for $j \in i - 2, \dots, i + 2$.
- The POS tag sequence of the segment $\langle t_j^r, t_{j+1}^r \rangle$ for $j \in i - 2, \dots, i + 1$.

as well as the following fidelity and morphological features:

- Whether t_i^r appears in the input—always 1 for our dataset of reachable fusions.
- Whether t_i^r is capitalized and the relative position of t_i^r in a sequence of capitalized words if it is capitalized.
- Whether t_i^r lies within parentheses.

We do not directly incorporate the support of content words as defined in §7.2.2.1 in the per-token feature map $\phi_{\text{tok}}(\mathcal{S}, r, i)$; instead, we use this signal in richer feature templates over both bigrams and dependency edges.

7.3.2 Bigram and dependency features

All bigram features described in §3.4.3 and dependency features from §3.4.4 are reused for scoring \mathbf{y} and \mathbf{z} in (7.2). In addition, we draw on the measure of token support Ω defined in §7.2.2.1 as the basis for the following additional feature templates included in both feature maps $\phi_{\text{ngr}}(\mathcal{S}, r, r', i, j)$ for bigrams $\langle t_i^r, t_j^{r'} \rangle$ as well as $\phi_{\text{dep}}(\mathcal{S}, r, r', i, j)$ for dependencies $\langle t_i^r, t_j^{r'} \rangle$.

- $\Omega(t_i^r)$ and whether $\Omega(t_i^r) \in \{2, 3, 4\}$ conjoined with the coarse POS of t_i^r , i.e., an indicator of whether t_i^r is a noun, verb, adjective or adverb.
- If t_k^r governs t_i^r in a dependency parse of S_r , $\Omega(t_k^r)$ and whether $\Omega(t_k^r) \in \{2, 3, 4\}$ conjoined with the coarse POS of t_k^r .
- The fraction of tokens from the subtree rooted at t_i^r in the dependency parse of S_r which have their $\Omega \in \{2, 3, 4\}$ conjoined with the coarse POS of t_i^r .
- The above templates applied to the second/dependent token $t_j^{r'}$ from $S_{r'}$ instead of the first/governor token t_i^r from S_r in the bigram/dependency under consideration.
- Indicators of whether $\Omega(t_i^r) \in \{2, 3, 4\}$ and $\Omega(t_j^{r'}) \in \{2, 3, 4\}$ conjoined with the POS tags of t_i^r and $t_j^{r'}$.

The impact of these support-based features is evaluated in §7.4.3.

7.3.3 Deriving features for reference fusions

In §3.5.2, we discuss the problem of generating features for reference compressions to be used in model training. Our solution to this involves progressively mapping tokens in a reference compression to those in the input sentence, starting with the least ambiguous mappings and then using these as contextual cues for further disambiguation. However, as illustrated in Table 7.2, fusion instances often feature lexical redundancy across multiple input sentences and output tokens are not order-constrained as in the compression task.

Consequently, it is impractical to disambiguate reference tokens based on their position or lexical context in a reference fusion.

We must therefore adopt a less aggressive mapping approach when deriving reference features for the fusion task. Because the reference sentence cannot be assumed to be a subsequence of the input sentence, the assumption that mappings must be monotonic is no longer applicable and, similarly, contextual overlap is no longer a reliable signal for disambiguation. We therefore rely on only lexical and syntactic overlap for disambiguating reference tokens in this task. For remaining ambiguous tokens, all remaining mappings are considered when generating token, bigram and dependency features and the resulting feature vectors are averaged.

7.4 Experiments

In order to evaluate the performance of multi-structured fusion, we ran experiments over the corpus described in §7.1. To aid reproducibility, we did not split the corpus randomly; instead, the 593 instances from the DUC evaluations covering the years 2005–2007 were chosen as a test dataset, while the 1267 instances from the TAC evaluations over 2008–2011 were used as a training dataset.¹² This yields an approximate 70/30 train-test split with near-identical proportions of 2-way, 3-way and 4-way fusions across the training and test partitions.¹³ We used a further 10% of the training dataset—composed entirely of instances from TAC 2011—as a development partition in order to tune feature configurations.

Sentence fusion is notoriously hard to evaluate even with human raters (Daumé III and Marcu, 2004). In prior work, we proposed an approach to aid manual evaluations of automated sentence *intersection* by reducing the evaluation task to judgments of textual entailment (Thadani and McKeown, 2011b); however, this reduction is not applicable in the context of generic sentence fusion. Here, we follow the standard approach of machine translation and choose to rely on transparent automated measures of output quality in order

¹²The actual training dataset used in our experiments consisted of 1265 instances. An error in sentence splitting led to the inadvertent removal of 2 valid fusion instances during the corpus construction process.

¹³In addition, this approach ensures that training and testing datasets never contain instances drawn from the same summarization corpus, which may share input sentences.

to engender more repeatable evaluations of fusion systems. Furthermore, as our test dataset is larger than most previously-studied fusion corpora in their entirety, statistical measures of output sentence quality are both viable and preferable for their coverage.

For the experiments presented here, we consider all n -gram and dependency F_1 measures that have been used to evaluate compression systems in Chapters 3–6. In order to ensure that comparisons between different systems remain fair, we also follow the compression evaluations in constraining output fusions to be a certain length—the length of the reference fusion. It should be noted that this constraint makes the fusion task *easier* since the output space is relatively smaller than that of the unconstrained fusion task. However, we expect that practical implementations of fusion systems will be able to meet—if not surpass—the performance of length-constrained fusion systems by incorporating predictive models or heuristic approaches to estimate an appropriate length for the output sentence given the input sentences.

The primary question under consideration in the following experiments is: what empirical effect does multi-structured inference have in the fusion setting? In addition to average measures over the full test dataset, we examine examples from the fusion output and look at performance variation with a different number of inputs. In addition, we investigate the effect of content selection under our approach to lexical redundancy from §7.2.2 as well as whether orientation-aware dependencies from §7.2.3 have an impact on performance measures.

7.4.1 Fusion quality

In this section, we consider the performance of the following inference approaches to sentence fusion, redefining system names from sentence compression where appropriate.

- **ILP-2gr**: An ILP over only the token and bigram variables from the objective in (7.2) and using the constraints in §3.3.2 to produce bigram-factored fusions.
- **ILP-dep**: An ILP over only the token and dependency variables from the objective in (7.2) and using the constraints in §3.3.1 to infer edge-factored dependency trees representing output fusions. Because we do not assume a total ordering of output tokens for this task, the output of this model is treated as a collection of unordered

Structures	Inference	F ₁ % for n-grams of order				Syntactic relations F ₁ %		
		1	2	3	4	z	Stanford	RASP
path	ILP-2gr	51.67	37.64	28.53	22.47	-	34.00	32.89
tree	ILP-dep	47.89	-	-	-	34.22	-	-
path + tree	ILP-2gr-dep	54.95	41.64	33.28	28.06	40.89	39.71	38.79

Table 7.3: Experimental results over the pyramid fusion corpus with all systems restricted to produce outputs of the same length as reference fusions. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

words.

- **ILP-2gr-dep:** The full ILP for multi-structured sentence fusion which targets the objective in (7.2) and combines the constraints from §3.3.1 to produce an edge-factored dependency tree with those from §3.3.2 to yield a bigram-factored token ordering for the output sentence.

Models are trained for these techniques using the minibatched structured perceptron as described in §3.5.

Table 7.3 enumerates the performance of these systems under measures of output sentence quality. We observe lower absolute numbers on all measures compared to the compression task, illustrating the relative difficulty of fusion due to the larger output space and variable token ordering. Regardless, multi-structured inference appears to yield a large performance advantage over both single-structured techniques in all measures. Unlike the compression systems, a clear gain is seen for **ILP-2gr-dep** over **ILP-2gr** under n-gram F₁ and over **ILP-dep** under dependency F₁, suggesting that the joint approach appears to offer more than the sum of its parts. Out of 593 test instances, **ILP-2gr** reproduces 70 (11.8%) references perfectly while **ILP-2gr-dep** reproduces 103 (17.4%).

Also notable in this comparison is the poor performance of **ILP-dep**, which is dramatically worse than the other systems in terms of unigram F₁. However, this is not entirely surprising when considering the vastly increased output space of the fusion task with respect to the extractive compression task examined in previous chapters. These results highlight the fact that edge-factored tree structures are relatively unconstrained compared

to bigram-factored paths; for instance, a verb may take multiple subjects in a valid solution. These errors can be avoided with language-dependent linguistic constraints as in Clarke and Lapata (2008) and Filippova and Strube (2008b) or with a pruning stage which precedes fusion inference. However, since the dependency-based model cannot produce an output fusion without a linearization component, we do not pursue remedies for its shortcomings in isolation and instead focus on its utility in joint inference with *ILP-2gr-dep*.

Because the dataset contains fusion instances with 2, 3, and 4 input sentences, we can examine the impact of additional inputs—which offer an additional source of salience but also a larger output space—on performance with respect to the reference sentence. Table 7.4 reports the evaluation measures computed separately over the 275 test instances with 2 inputs, the 162 with 3 inputs and the remaining 156 with 4 inputs.¹⁴ In each case, *ILP-2gr-dep* maintains a statistically significant margin over the single-structure systems, with absolute gains ranging from 4–7% in 4-gram F_1 and 5–7% in RASP F_1 over *ILP-2gr* as well as 4–8% in dependency F_1 over *ILP-dep*.

Furthermore, system performance appears to steadily decrease as the number of input sentences increases. This phenomenon can be generally attributed to a contingent increase in the number of possibilities for constructing a valid output sentence, which appears to outpace the marginal signals of support provided by additional input sentences. We also considered whether this variation might stem from an increasing proportion of valid fusions which do not match the sole designated reference compression. To verify this, we assembled pseudo-references using only the *contributor* spans designated by SCU annotators (cf. 7.1) which are not necessarily full sentences or even contiguous fragments of text. As the number of contributors for any instance is equal to the number of input sentences, averaging the F_1 measures against these pseudo-references should be less sensitive to input size than the sole reference used in our primary evaluation. However, the variation in F_1 measures across systems and the number of input sentences was observed to be similar to the results in Table 7.4.

¹⁴The models used to produce these numbers remain unchanged and were trained on the full training dataset composed of instances with 2, 3 and 4 input sentences.

Num. inputs	Inference	F ₁ % for n-grams of order				Syntactic relations F ₁ %		
		1	2	3	4	z	Stanford	RASP
2	ILP-2gr	54.71	42.89	35.16	29.51	-	38.91	37.60
	ILP-dep	52.24	-	-	-	40.10	-	-
	ILP-2gr-dep	57.06	46.04	38.63	33.42	44.78	43.83	42.82
3	ILP-2gr	51.61	36.25	25.91	19.02	-	32.13	30.85
	ILP-dep	46.40	-	-	-	32.02	-	-
	ILP-2gr-dep	55.62	40.69	31.83	26.16	40.65	38.92	38.08
4	ILP-2gr	46.37	29.83	19.57	13.64	-	27.28	26.72
	ILP-dep	41.77	-	-	-	26.13	-	-
	ILP-2gr-dep	50.55	34.87	25.37	20.58	34.27	33.24	32.40

Table 7.4: Results over the pyramid fusion corpus broken down by number of input sentences. Boldfaced entries are statistically undistinguished from the best result within each column *and* row group under Wilcoxon’s signed rank test ($p < 0.05$).

We expect that this phenomenon of decreasing performance with additional input sentences can be controlled in various ways when considering practical fusion systems. For instance, most previous work on fusion uses a multi-stage approach in which the first stage consists of explicit alignment of the tokens in the input sentence and fusion inference operates over graphs of bigrams (Filippova, 2010; Thadani and McKeown, 2011b) or dependencies (Barzilay and McKeown, 2005; Filippova and Strube, 2008b). Although error-propagation is a risk with such strategies, they can be used to disambiguate redundant substructures and prune implausible constructions while remaining compatible with multi-structured inference in a following phase.

7.4.2 Example output

Table 7.5 contains examples of reference and system-generated fusions for instances from the test partition. We selected examples for readability and diversity, favoring instances with few errors with respect to the references while avoiding perfect results, which are uninteresting from the point of view of analysis, as well as overly noisy results, from which

insights cannot be easily gleaned. This is therefore not intended to be a representative sample of system performance over the test dataset.

Unsurprisingly, we observe that **ILP-2gr** makes errors common to n-gram models with short context windows. For instance in example (a),¹⁵ the semantic error introduced by **ILP-2gr** in substituting *capsules* for *current* is likely due to the inability of the model to perceive that the previous word is *by* rather than *using*. This error is avoided by **ILP-2gr-dep** which accounts for the syntactic dependency between *by* and *currents*. In other examples like (e) and (f), the limitations of relying on a local bigram factorization with **ILP-2gr** manifest as separate clauses which are joined by a single overlapping noun phrases—a rarer occurrence in **ILP-2gr-dep** solutions. However, both systems are vulnerable to outputs with leading and trailing truncations, as illustrated by examples (c) and (i).

We present the output of **ILP-dep** as a set of tokens since linearization is not provided by the model.¹⁶ In the absence of additional information to guide the production of output sentences, these **ILP-dep** token solutions largely do not appear amenable to resolution as meaningful text—as is also evidenced by the lower unigram F_1 scores for **ILP-dep**. For instance, the **ILP-dep** solution in example (d) contains all the verbs from input sentences because the redundancy constraints from §7.2.2.2 do not apply to non-synonyms. In contrast, the **ILP-2gr** and **ILP-2gr-dep** solutions in that instance avoid redundant verbs.

The bigram-based **ILP-2gr** often produces fluent sentences and occasionally recovers solutions that rival **ILP-2gr-dep**, e.g., examples (d) and (g) in Table 7.5. However, both **ILP-2gr** and **ILP-2gr-dep** are often penalized in comparisons with reference sentences even when they are perfectly valid fusions. In addition to comparing system outputs to contributor sentences as described previously, we also conducted an intrinsic evaluation by comparing the log-likelihood of system responses for test instances under a trigram LM trained on Gigaword 3. Similar performance is recorded for both systems: an average log-probability of -284.70 for **ILP-2gr** and -284.06 for **ILP-2gr-dep**—a statistically significant

¹⁵A similar issue is seen in example (h).

¹⁶Linearization of dependency trees is usually accomplished with LMs (Bangalore and Rambow, 2000) and is a common feature of most previous work on fusion (Barzilay and McKeown, 2005; Filippova and Strube, 2008b; Elsner and Santhanam, 2011).

(a)	Input (i)	Russian efforts to reach the submarine using underwater capsules failed and Russia talked to NATO officials .
	Input (ii)	Russian navy rescue efforts were complicated by strong underwater currents and near zero visibility at the 450 feet depth where the vessel was trapped and by heavy surface storms .
	Reference	Russian efforts to reach the submarine failed
	ILP-2gr	Russian efforts were complicated by underwater capsules
	ILP-dep	{Russian, reach, failed, efforts, near, and, by}
	ILP-2gr-dep	Russian efforts were complicated by underwater currents
(b)	Input (i)	At McKinney 's trial in October , McKinney 's attorney pleaded that McKinney committed manslaughter in five minutes of " gay panic " to a sexual advance from Shepard , because drug and alcohol addicted McKinney had been abused homosexually at age 7 and had consensual homosexual sex at 15 .
	Input (ii)	In attempting to get his client a life sentence rather than the death penalty , he argued that McKinney 's past , which included sexual abuse as a child , caused him to lash out in rage at Shepard 's sexual advances .
	Reference	His attorney pleaded that McKinney committed manslaughter in five minutes of " gay panic "
	ILP-2gr	At McKinney 's attorney pleaded that McKinney 's sexual abuse as a life sentence rather
	ILP-dep	{McKinney, McKinney, pleaded, McKinney, committed, in, ", to, rather, McKinney, past, lash, out, in, 's}
	ILP-2gr-dep	McKinney 's attorney pleaded that McKinney committed manslaughter in rage at Shepard 's sexual advances
(c)	Input (i)	Two aboriginal protestors won the 1999 Goldman Environmental Prize for having delayed the mine project .
	Input (ii)	In 1999 Jacqui Katona and Yvonne Mararula , two aboriginal protestors of the mine , were awarded the Goldman Environmental Prize for protecting the tropical rainforest through protests to delay mining in the Jubiluka mine and an appeal to a UN panel to halt mine operation .
	Reference	Katona and Mararula won the Goldman Environmental Prize in 1999
	ILP-2gr	In 1999 Goldman Environmental Prize for having delayed the mine
	ILP-dep	{won, 1999, Goldman, Environmental, for, having, delayed, Prize, for, protecting}
	ILP-2gr-dep	aboriginal protestors won the Goldman Environmental Prize for having delayed

Table 7.5: Examples of reference and system-generated fusions for instances with 2 input sentences from the test partition of the pyramid fusion corpus.

(d)	Input (i)	In 2000 the US agreed to resettle 600 Burmese refugees .
	Input (ii)	The US agreed to accept 1,500 Burmese student refugees from Thailand .
	Input (iii)	The US accepted hundreds of Burmese student refugees .
	Reference	In 2000 the US agreed to resettle 600 Burmese refugees
	ILP-2gr	The US agreed to resettle 600 Burmese refugees from Thailand
	ILP-dep	{In, to, resettle, refugees, agreed, to, accept, US, Burmese, student}
	ILP-2gr-dep	In 2000 the US agreed to accept 1,500 Burmese refugees
(e)	Input (i)	More than 450 bills were introduced nation-wide on gay and lesbian issues .
	Input (ii)	Shepard 's murder , in part , prompted over 450 bills to be introduced in state legislatures on issues important to gays and lesbians .
	Input (iii)	Over 450 bills on issues important to gays and lesbians were introduced in legislatures across the country in 1999 , prompted in part by the murder of Matthew Shepard .
	Reference	More than 450 bills were introduced nation-wide on gay and lesbian issues
	ILP-2gr	Shepard 's murder prompted over 450 bills were introduced nation-wide on issues
	ILP-dep	{bills, nation-wide, on, Shepard, prompted, to, be, in, issues, important, introduced, murder}
	ILP-2gr-dep	Over 450 bills were introduced in legislatures on issues important to gays
(f)	Input (i)	In a 1994 Justice Department settlement Microsoft changed some licensing practices with computer makers .
	Input (ii)	In 1994 , the Justice Department reached a settlement with Microsoft requiring the company to change its business practices .
	Input (iii)	In 1994 Justice settled with Microsoft , requiring them to change aspects of licensing agreements with PC makers .
	Reference	In a 1994 Justice Department settlement Microsoft changed some licensing practices with computer makers .
	ILP-2gr	Justice Department reached a settlement with Microsoft changed some licensing practices with PC makers
	ILP-dep	{settlement, Microsoft, changed, licensing, In, 1994, Department, with, practices, In, Justice, with, with, makers}
	ILP-2gr-dep	In 1994 Justice Department settlement with Microsoft changed some licensing practices with computer makers

Table 7.5: Examples of reference and system-generated fusions for instances with 3 input sentences from the test partition of the pyramid fusion corpus.

(g)	Input (i)	In France , art dealers are obliged by law to register all purchases except those bought at public auction .
	Input (ii)	Also , French dealers are required to register all purchases .
	Input (iii)	French law requires dealers to register purchases .
	Input (iv)	In France dealers must register all purchased art , except what is bought at auction .
	Reference	In France , art dealers are obliged by law to register all purchases
	ILP-2gr	French law requires dealers to register all purchases except those bought at auction
	ILP-dep	{law, to, except, bought, register, French, requires, dealers, to, must, art, except, at}
	ILP-2gr-dep	French dealers are required to register all purchases except those bought at auction
(h)	Input (i)	China ’s “ one child ” policy has kept population growth down to the same 1.9 percent as in western Europe .
	Input (ii)	China , the most populous country , has instituted programs to limit births but will still see large gains in population .
	Input (iii)	The policy has had a dramatic effect , with the fertility rate falling to around 1.9 from 2.25 by 1994 .
	Input (iv)	China has a strict birth control policy but it is not effective in the countryside .
	Reference	China ’s “ one child ” policy has kept population growth down
	ILP-2gr	China ’s “ policy has kept population growth down to the countryside
	ILP-dep	{China, one, kept, down, to, has, to, but, policy, to, 1.9, has}
	ILP-2gr-dep	China ’s policy has kept population growth down to around 1.9 percent
(i)	Input (i)	On January 1 , 1993 , the agreed upon separation was official .
	Input (ii)	On New Year ’s day , 1993 , Slovaks sloughed off a thousand year subservience to Hungary and seven decades as the junior partner in Czechoslovakia and celebrated the birth of a sovereign , independent republic .
	Input (iii)	Two new states were born on January 1 , 1993 .
	Input (iv)	The amicable separation became official January 1 , 1993 after which the two began moving rapidly apart .
	Reference	On January 1 , 1993 , the agreed upon separation was official
	ILP-2gr	and seven decades as the agreed upon separation was official January 1
	ILP-dep	{1993, separation, was, official, On, and, as, 1993, became, January, after, the}
	ILP-2gr-dep	as the agreed upon separation became official January 1 , 1993 .

Table 7.5: Examples of reference and system-generated fusions for instances with 4 input sentences from the test partition of the pyramid fusion corpus.

Structures	Inference	Content selection	F ₁ % for n-grams of order				Syntactic relations F ₁ %		
			1	2	3	4	z	Stanford	RASP
path	ILP-2gr	−support	45.58	32.72	25.14	20.86	-	29.26	28.63
		+support	51.67	37.64	28.53	22.47	-	34.00	32.89
tree	ILP-dep	−support	42.91	-	-	-	30.17	-	-
		+support	47.89	-	-	-	34.22	-	-
path + tree	ILP-2gr-dep	−support	53.31	40.28	32.46	27.51	40.16	38.85	37.40
		+support	54.95	41.64	33.28	28.06	40.89	39.71	38.79

Table 7.6: Experimental results over the pyramid fusion corpus with all systems restricted to produce outputs of the same length as reference fusions. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

improvement at $p < 0.05$ —suggesting that fluency might play a smaller role in the difference between these systems than the ability to identify salient information for fusions. A deeper examination of our approach to content selection follows.

7.4.3 Content selection

Table 7.6 compares the performance of the previously-seen systems, which use support features from §7.3.2 alongside redundancy constraints from §7.2.2.2, against equivalent systems trained without these features or constraints. The former scenario is denoted by the label *+support* and the latter, which is similar to compression inference from Chapter 3, is summarized by the label *−support*. Interestingly, although the *+support* additions appear to play a significant role in the performance of ILP-2gr—with a 5% increase in bigram F₁—as well as for ILP-dep—with a 4% increase in dependency F₁—they do not appear to make a large impact on ILP-2gr-dep, with all measures gaining by less than 2%. We interpret this as an indication that the joint model is relatively capable of identifying salient content for fusion whereas a single-perspective model might be challenged to do so without additional indicators of salient content.

To contextualize the quality of content selection by these models, we evaluate a further set of systems which do not rely on support features but instead operate on different input

Structures	Inference	Content selection	F ₁ % for n-grams of order				Syntactic relations F ₁ %		
			1	2	3	4	z	Stanford	RASP
path	ILP-2gr	<i>+support</i>	51.67	37.64	28.53	22.47	-	34.00	32.89
		(contribs)	61.94	47.54	37.67	31.74	-	44.84	43.74
tree	ILP-dep	<i>+support</i>	47.89	-	-	-	34.22	-	-
		(contribs)	55.97	-	-	-	41.68	-	-
path + tree	ILP-2gr-dep	<i>+support</i>	54.95	41.64	33.28	28.06	40.89	39.71	38.79
		(contribs)	64.86	51.62	42.99	37.92	51.34	51.05	49.47

Table 7.7: Experimental results over the pyramid fusion corpus with all systems restricted to produce outputs of the same length as reference fusions. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

data—the SCU *contributors* for each instance instead of the full source sentences. As described in §7.1, these contributors are human-selected spans within each input sentence that realize the content of an SCU, thereby serving as indirect human annotations for content selection in the fusion task. Moreover, one-third of the instances in the pyramid fusion corpus (660 instances) feature an output fusion that is an exact string match of one of their contributors—more than double the number of input sentences which exactly match a reference fusion (300 instances).¹⁷

Table 7.7 contains a comparison between the systems which operate on contributors as input to those which operate on the full input sentences and rely on support features for content selection. We observe a difference of 9–11% between the two groups, indicating that further refinements will be necessary to approach human performance on content selection for fusion problems.¹⁸ Although annotations that resemble these contributors are unlikely

¹⁷We chose to retain these contributors and input sentences in the dataset in order to more accurately model the decisions of human annotators who were generating SCUs over the sentences.

¹⁸Interestingly, initial results for ILP-2gr over this dataset (Thadani and McKeown, 2013b) yielded *+support* system performance which was competitive with the model over human-labeled contributors. However, all results reported here exhibit significant performance gains over those systems owing to feature refinements as well as a more rigorous derivation process for reference features. These improvements therefore appear to have a greater impact on the contributor-guided baseline system than systems which operate

to be available for practical applications of sentence fusion, this result also suggests that admitting *any* human-supplied cues for content selection—as in the query-based sentence fusion approaches of Krahmer et al. (2008)—may aid fusion models which otherwise struggle with identifying the salient information in complex sentences.

7.4.4 Dependency orientation

Without a fixed token ordering for fusion problems as in the compression task studied previously, it is perhaps unsurprising that output dependencies in \mathbf{z} may have orientations which are incompatible with the token ordering in \mathbf{y} . We therefore also considered the following extension of `ILP-2gr-dep` proposed in §7.2.3 to introduce orientation-aware dependencies which are scored differently depending on whether they form right or left attachments in the fusion solution.

- `ILP-2gr-depo`: A variation of `ILP-dep` with separate dependency variables for left and right attachments and constraints (7.5)–(7.6) linking them to the orientation of the dependencies in the output sentence.

A new model is trained for this technique in the same experimental environment as that of `ILP-2gr-dep`. The features used for each dependency edge are the features from the original model—namely those drawn from §3.4.4 and the support features from §7.3.2—although the use of separate variables allows them to acquire different parameters during training. During development experiments, we did not see an advantage in introducing features that explicitly acknowledged the orientation of output dependencies.

Table 7.8 contains a comparison of `ILP-2gr-depo` and `ILP-2gr-dep`. We observe a slim but noticeable improvement across all measures for `ILP-2gr-depo` relative to `ILP-2gr-dep` including a statistically significant absolute increase of 1% in {3,4}-gram F_1 and dependency F_1 . A modest gain for this model is in line with expectations due to the sparsity of direction-specific dependencies and the observation that output dependencies between tokens drawn from the same input sentence largely preserve their orientations. The improvement is distributed evenly across instances featuring 2, 3 and 4 input sentences.

over full input sentences.

Structures	Inference	F ₁ % for n-grams of order				Syntactic relations F ₁ %		
		1	2	3	4	z	Stanford	RASP
path + tree	ILP-2gr-dep	54.95	41.64	33.28	28.06	40.89	39.71	38.79
	ILP-2gr-dep ^o	55.16	42.19	34.31	29.11	41.79	40.46	39.39

Table 7.8: Experimental results for joint models over the pyramid fusion corpus with all systems restricted to produce outputs of the same length as reference fusions. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

We note, however, that inference with ILP-2gr-dep^o introduced a significant overhead in computational resources. In general, ILP inference over fusion problems is slower than for compression because of increase in program size for multiple input sentences. Doubling the number of dependency edges and all their contingent constraints appeared to yield significantly memory usage and far slower inference for the Gurobi solver¹⁹ used here. We therefore expect that orientation-aware edges will not be usable in practical fusion systems without a pruning stage to identify the dependency edges which are most likely to flip their orientations in attachments across input sentences.

7.5 Remarks

The key contribution of this chapter is an extension of the ILP-based compression technique of Chapter 3 to accommodate multiple input sentences with overlapping information, thereby yielding a multi-structured approach to the challenging problem of sentence fusion. We introduce additional features and constraints to recognize redundancy in the input sentences and control dependency orientation in the output sentences. Moreover, we identify a novel dataset of natural sentence fusions that can be reconstructed from summarization evaluation datasets which are freely available to the natural language processing community.

In our experiments, we observe a significant benefit for the multi-structured approach in all evaluations for fusion problems with a varying number of input sentences. Most

¹⁹We use Gurobi 6.0: <http://www.gurobi.com>.

interestingly, multi-structured fusion without features indicating token support across input sentences significantly outperforms bigram-based fusion with support features, indicating that the joint approach also has an inherent advantage in content selection.

The variations on compression inference described previously indicate potential research directions for the fusion task. For instance, the approximation technique in Chapter 4 can be extended to fusion with an efficient technique for bigram-based fusion inference. We are currently considering the use of shortest-path algorithms—previously applied to fusion by Filippova (2010)—with projected gradient descent to permit discriminative learning and efficient inference for these problems.

Our investigation thus far has indicated that multi-structured inference offers consistent benefits for a variety of text-to-text generation tasks, prompting the question: can these formulations be useful other natural language processing problems? We examine this further through a multi-structured formulation for *monolingual alignment* problems in the following chapter. Among other applications, monolingual alignment is useful for identifying support in fusion problems (Marsi and Krahmer, 2005; Elsner and Santhanam, 2011; Thadani and McKeown, 2011b) and generating lexical resources for text-to-text tasks such as paraphrasing and simplification.

Chapter 8

Multi-Structured Monolingual Alignment

Textual alignment involves the identification of links between words or phrases which are semantically equivalent in their respective sentences. Although alignment is often considered in the context of phrase-table construction for MT systems, *monolingual* alignment is often needed in natural language problems which involve pairs or groups of related sentences. For instance, most approaches for recognizing textual entailment rely on alignment techniques to establish the overlap between an input premise and a hypothesis before determining if the former entails the latter. Automated question answering techniques often rely on alignment to identify answers to input questions in large corpora. Monolingual alignment is also relevant to text-to-text generation—sentence fusion approaches often contain an explicit alignment phase (Barzilay and McKeown, 2005; Filippova and Strube, 2008b; Thadani and McKeown, 2011b) or incorporate alignment into inference (Elsner and Santhanam, 2011).

This chapter extends the notion of multi-structured inference as previously discussed for text-to-text generation techniques to the problem of supervised monolingual alignment. In addition to modeling alignments as a set of aligned phrase pairs (MacCartney et al., 2008; Thadani and McKeown, 2011a; Yao et al., 2013b), we also account for syntax by including an *edge-based* alignment representation composed of a set of aligned pairs of dependency

edges consistent with the phrase-based representation.¹ Under this formulation, the score of any alignment is defined to factor over all aligned phrase pairs and edge pairs in the alignment.

Recovering a full sentence alignment that optimizes this joint scoring function is non-trivial due to both the interdependence among individual phrase alignments—resulting in an NP-hard problem (DeNero and Klein, 2008)—as well as the interaction between phrase-based and edge-based alignments to ensure consistency between the two representations. As before, we describe an ILP to recover joint phrasal and edge-based alignments parameterized by feature-based scoring functions. We evaluate this joint aligner on human-annotated corpus for textual entailment (Brockett, 2007) and paraphrasing (Cohn et al., 2008) and observe significant gains over phrase-based alignments generated by the Meteor metric for machine translation (Denkowski and Lavie, 2011) as well as a state-of-the-art discriminatively-trained phrase-based aligner (Thadani and McKeown, 2011a).

The contributions of this chapter are as follows:

- An improved version of the paraphrase alignment corpus of Cohn et al. (2008) with improved tokenization and collapsed named entities.²
- A multi-structured inference approach to monolingual text alignment that jointly induces phrasal and dependency-based alignment relations.

8.1 Aligned Paraphrase Corpus

As our dataset, we use a modified version of the human-aligned corpus of paraphrases described by Cohn et al. (2008), which we call the *Edinburgh corpus*. We derive this dataset from the original corpus first by standardizing the treatment of quotes (both single and double) and by truecasing the text (Lita et al., 2003). Following MacCartney et al. (2006),

¹This is joint work with Scott Martin and Michael White and was originally presented in Thadani et al. (2012). An initial approach to syntactically-informed phrase-based alignment was addressed in Thadani and McKeown (2011a).

²The modified corpus is available at <http://www.ling.ohio-state.edu/~mwhite/data/coling12/>.

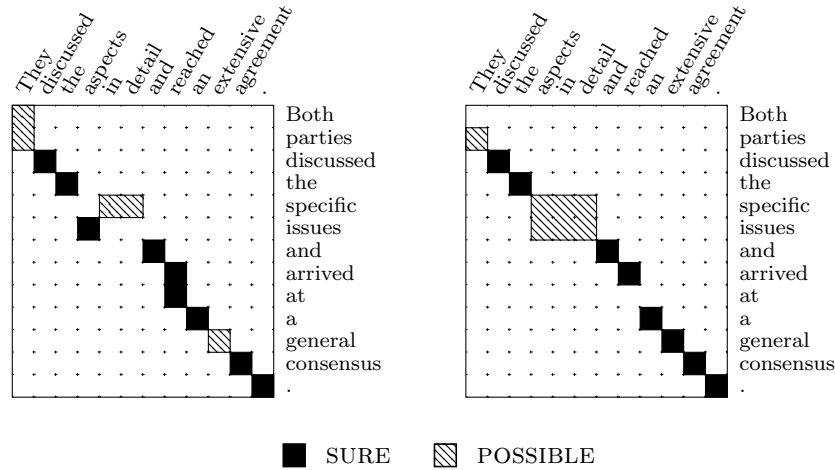


Figure 8.1: Two examples of human-authored phrase alignments between the sentences “*They discussed the aspects in detail and reached an extensive agreement.*” and “*Both parties discussed the specific issues and arrived at a general consensus.*” drawn from Cohn et al. (2008).

we collapse named entities using the Stanford named entity recognizer³ trained on the pre-built models distributed with it (Finkel et al., 2005). For example, the corpus contains a sentence with the named entity *Bank of Holland*, which we collapse to the single token *Bank_of_Holland*.

A fraction of the alignment instances in the original corpus of Cohn et al. (2008) contains annotations from multiple annotators. Figure 8.1 contains one such example in which the human annotators have disagreements. In order to extract training and testing splits, we use all of the non-overlapping portions of the corpus—those only aligned by a single human annotator—as training data. We then randomly sampled training instances from the overlapping portions of the corpus: 45 instances from the ‘trial’ portion drawn from the *mtc* subcorpus, 19 from the *news* portion, and 10 from the *novels* portion. The testing data includes all of the instances in the overlapping portions of the corpus that are not selected as training data, plus the five remaining ‘trial’ instances. The resulting splits yield 70% for training and 30% for testing, with identical ratios from the three subcorpora—*mtc*, *news* and *novels*) in both training and testing. The training set has 715 paraphrase pairs

³Stanford named-entity recognizer: <http://nlp.stanford.edu/software/CRF-NER.shtml>

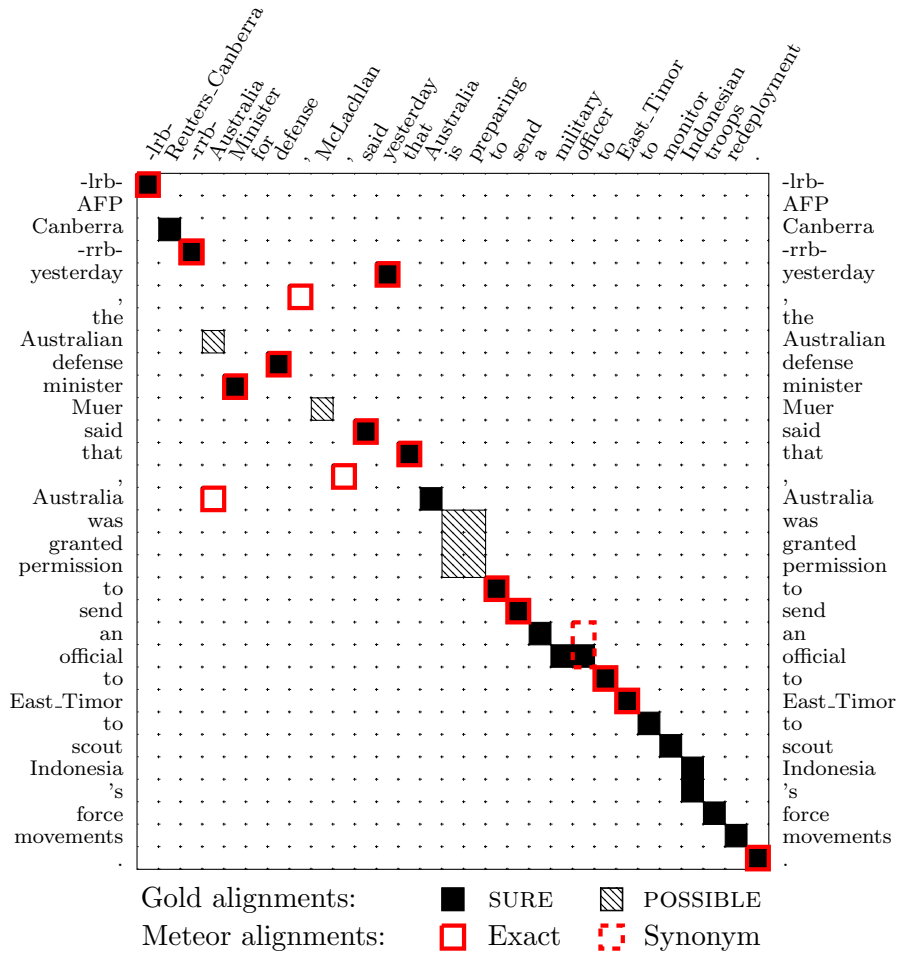


Figure 8.2: Alignment grid for a training example from the Edinburgh corpus with annotated alignments—both SURE and POSSIBLE—in black as well as Meteor alignments in red.

with a total of 29,827 tokens and an average of 20.9 tokens per sentence, while the test set has 305 paraphrase pairs with 14,391 tokens and 23.6 tokens/sentence on average. Finally, rather than using the merged alignments from the Edinburgh corpus for the overlapping portions, we randomly select one of the two annotators to use as the reference alignment in an unbiased way, with each annotator chosen exactly half of the time.

8.1.1 Corpus analysis

Figure 8.2 shows an example paraphrase pair from the training portion of the corpus. The recall errors in the Meteor alignments that are supported by Stanford parser dependencies

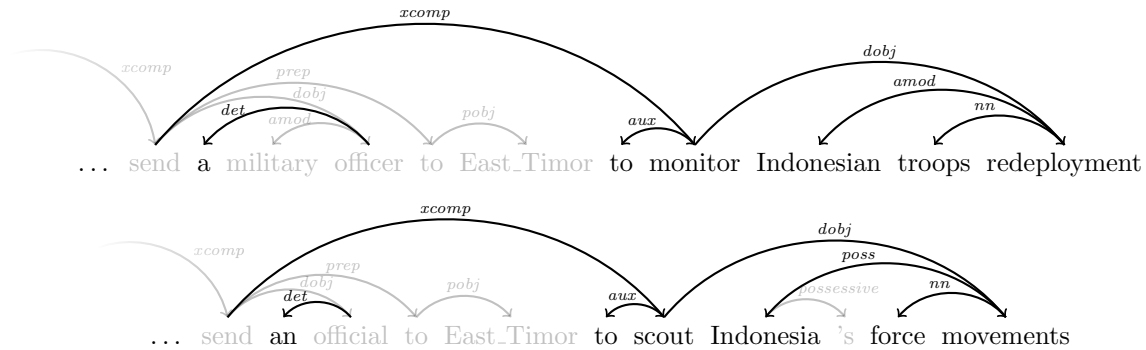


Figure 8.3: Highlighted tokens indicate recall errors for Meteor which are supported by easily-aligned Stanford dependencies in the alignment example from Figure 8.2.

are highlighted in Figure 8.3. These recall errors are *supported* in the sense that the missed aligned tokens participate in dependencies with other aligned tokens. For example, Meteor fails to align *scout* with *monitor*. This token-level alignment is supported by two aligned dependencies, namely the alignment of *send* \xrightarrow{xcomp} *scout* with *send* \xrightarrow{xcomp} *monitor* and *scout* \xrightarrow{aux} *to* with *monitor* \xrightarrow{aux} *to*. Here, the other tokens in the dependencies are identical, and thus the dependencies provide strong evidence for the token-level alignment. Interestingly, the final three recall errors involve interrelated dependencies, suggesting the need for joint inference.

Using this notion of dependency edge alignments supporting token-level alignments, we counted how frequently the token alignments were supported by dependency alignments, and found that 64% of the SURE alignments and 65% of the SURE+POSSIBLE alignments in the training dataset were supported in this way. We also tabulated how often the dependencies were aligned, and found that 54% of the dependency edges were aligned based on the SURE token alignments, and 62% were aligned based on the SURE+POSSIBLE alignments, thus indicating the greater potential of dependencies to aid alignment when including the POSSIBLES. The alignment percentages varied considerably by type: of the non-rare dependency types, 74% of the *aux* dependencies were aligned (including the POSSIBLES), while only 38% of the *rcmod* dependencies were aligned, with most core dependency types such as *xcomp* and *dobj* in the 64-70% range.⁴

⁴Note that dependencies can fail to be aligned for a variety of reasons including parse errors, head-

8.2 Multi-Structured Alignment

We first introduce some notation for alignment problems. Consider a pair of text segments $\langle \mathcal{S}_1, \mathcal{S}_2 \rangle$ where each segment consists of one or more sentences. Let \mathcal{T}_r denote the set of tokens in text segment \mathcal{S}_r , i.e., each t_i^r represents a token in the i th position of segment \mathcal{S}_r where $r \in \{1, 2\}$. Finally, we use $n_r \triangleq |\mathcal{T}_r|$ to denote the number of tokens in each input text segment.

8.2.1 Alignment as linear optimization

Let A indicate an alignment between \mathcal{S}_1 and \mathcal{S}_2 . The MAP inference problem can be cast as the problem of finding the alignment \hat{A} under some linear scoring function.

$$\begin{aligned} \hat{A} &\triangleq \arg \max_A \Delta(\mathcal{S}_1, \mathcal{S}_2, A) \\ &= \arg \max_A \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathcal{S}_1, \mathcal{S}_2, A) \end{aligned} \quad (8.1)$$

where we recycle notation for the scoring function Δ , the alignment feature map $\boldsymbol{\phi}$ and parameters $\boldsymbol{\theta}$ from Chapter 3. The tractability of this maximization hinges on the definition of *alignment* used and specifically the factorizations it permits.

A fundamental feature of text alignment is the restriction that every input token must participate in only one alignment relation. In a *word alignment* setting, alignment relations are defined over pairs of tokens and the maximization in (8.1) can be solved in polynomial time using the Hungarian algorithm for assignment (Kuhn, 1955; Munkres, 1957). However, multi-word paraphrases such as the ones in the alignment examples from Figure 8.1 cannot be easily decomposed into token pairs and thereby introduce ambiguity for both human annotators and automated alignment systems.

Instead, we address the more general problem of *phrase-based alignment* in which a phrase is composed of one or more contiguous tokens from an input text segment, often with a limit δ on the length of a phrase. Using $\mathcal{V}_r \triangleq \{\langle t_i^r, \dots, t_j^r \rangle : 1 \leq i \leq j \leq i + \delta \leq n_r\}$ to denote the set of all alignable phrases from text segment \mathcal{S}_r , a valid phrase-based alignment is a subset of $\mathcal{V}_1 \times \mathcal{V}_2$ with every input token represented no more than once in an

dependent inversions (not taken into account here) and more large-scale structural divergences.

alignment relation. Phrase-based alignment provides a more natural representation for textual equivalence in natural language and has also found widespread use as a component of phrase-based machine translation (Och and Ney, 2003). Unfortunately, this problem is NP-hard in the general case (DeNero and Klein, 2008) and has previously been addressed through simulated annealing search (MacCartney et al., 2008) and ILP formulations (DeNero and Klein, 2008; Thadani and McKeown, 2011a).

8.2.2 Multi-structured objective

As illustrated in §8.1.1, correspondence in syntactic structure over input sentences offers a valuable signal for alignment even in the absence of exact word overlap. Moreover, we demonstrated in Thadani and McKeown (2011a) that syntactic constraints for certain modifiers such as determiners and prepositions help avoid spurious alignments of common function words. However, rather than manually defining hard rules over potentially-noisy input syntax, we now adopt a multi-structured perspective to this problem and view input text segments as both an ordered sequence of tokens as well as a forest of dependency parse trees for every sentence in the text segment. We can thus capture this syntactic context in an alternative alignment structure defined over *dependencies* from the parses of the input text segments.

More formally, we associate each \mathcal{T}_r with a dependency forest represented by a set of labeled edges $\mathcal{Z}_r \triangleq \{(t_i^r, t_j^r) : t_j^r \in \mathcal{T}_r \text{ is a dependent of } t_i^r \in \mathcal{T}_r \cup \{\text{ROOT}\}, i \neq j\}$ such that every token from \mathcal{T}_r is a dependent in exactly one edge and the ROOT is represented by a special token $t_0^r \notin \mathcal{T}_r$ which is not a dependent in any edge.⁵ A valid dependency alignment is now a subset of $\mathcal{Z}_1 \times \mathcal{Z}_2$ with every input dependency represented no more than once.⁶ The alignment of dependency trees in this manner is a key component of some sentence fusion systems (Barzilay and McKeown, 2005; Filippova and Strube, 2008b) and we might expect that jointly aligning dependencies alongside phrases will prove valuable for the phrase-based alignment task as well.

⁵When the number of sentences in a text segment $|\mathcal{S}_r| > 1$, we assume the dependency parses of all its sentences share a common ROOT node.

⁶This is equivalent to each token being represented at most once as a *dependent* in an alignment relation.

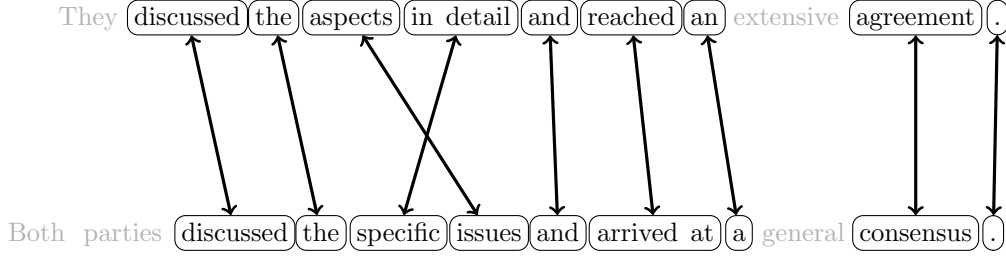


Figure 8.4: A potential phrase-based alignment solution \mathbf{v} to the example from Figure 8.1. Faded tokens do not participate in the alignment structure.

We now describe a joint objective for phrase-based and dependency-based alignment. Let v_{ijkl} denote a binary variable which indicates whether the phrases $\langle t_i^1, \dots, t_j^1 \rangle \in \mathcal{V}_1$ and $\langle t_k^2, \dots, t_l^2 \rangle \in \mathcal{V}_2$ are aligned in a phrase-based alignment and let $\Delta_{\text{phr}}(\mathcal{S}_1, \mathcal{S}_2, i, j, k, l)$ denote the corresponding feature-based scoring function. Similarly, let w_{jl} denote a binary variable which indicates whether the sole dependency edge from \mathcal{Z}_1 which features $t_j^1 \in \mathcal{T}_1$ as a dependent is aligned to the edge from \mathcal{Z}_2 which has $t_l^2 \in \mathcal{T}_2$ as a dependent, with $\Delta_{\text{edg}}(\mathcal{S}_1, \mathcal{S}_2, j, l)$ representing the corresponding edge alignment score. We can now write the objective from (8.1) in terms of these indicator variables as follows.

$$\begin{aligned}
\hat{A} &= \arg \max_{\mathbf{v}, \mathbf{w}} \sum_{\substack{i,j: \\ \langle t_i^1 \dots t_j^1 \rangle \in \mathcal{V}_1}} \sum_{\substack{k,l: \\ \langle t_k^2 \dots t_l^2 \rangle \in \mathcal{V}_2}} v_{ijkl} \cdot \Delta_{\text{phr}}(\mathcal{S}_1, \mathcal{S}_2, i, j, k, l) \\
&\quad + \sum_{j: t_j^1 \in \mathcal{T}_1} \sum_{l: t_l^2 \in \mathcal{T}_2} w_{jl} \cdot \Delta_{\text{edg}}(\mathcal{S}_1, \mathcal{S}_2, j, l) \\
&= \arg \max_{\mathbf{v}, \mathbf{w}} \mathbf{v}^\top \mathbf{\Delta}_{\text{phr}} + \mathbf{w}^\top \mathbf{\Delta}_{\text{edg}}
\end{aligned} \tag{8.2}$$

where the indicator vector $\mathbf{v} \triangleq \langle v_{ijkl} \rangle_{\langle t_i^1 \dots t_j^1 \rangle \in \mathcal{V}_1, \langle t_k^2 \dots t_l^2 \rangle \in \mathcal{V}_2}$ compactly represents a full phrase-based alignment configuration, the indicator vector $\mathbf{w} \triangleq \langle w_{jl} \rangle_{t_j^1 \in \mathcal{T}_1, t_l^2 \in \mathcal{T}_2}$ represents a dependency alignment configuration, and $\mathbf{\Delta}_{\text{phr}}$ and $\mathbf{\Delta}_{\text{edg}}$ denote corresponding vectors of scores for phrase and dependency edge alignment respectively.

Naturally, structural considerations prevent efficient optimization of this objective. Specifically, the following conditions must hold for valid and consistent alignments.

- Every input token is present in at most one phrase alignment v_{ijkl}
- Every input dependency is present in at most one dependency edge alignment w_{ij}

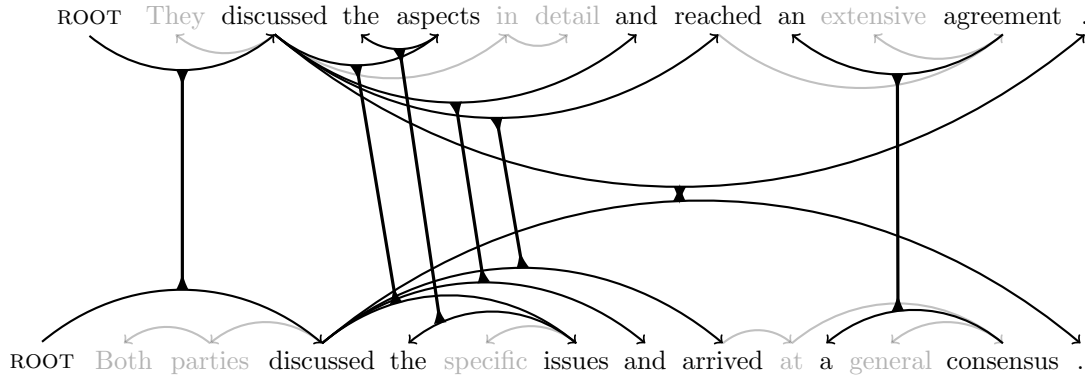


Figure 8.5: A potential dependency alignment solution \mathbf{w} to the example from Figure 8.1 which is consistent with the phrase-based alignment solution from Figure 8.4. Faded tokens and edges do not participate in the dependency alignment.

- Phrase and dependency alignments are consistent, i.e., governors and dependents of aligned dependency edges must also appear in aligned phrase pairs.

8.2.3 Inference via ILP

Unlike the text-to-text generation tasks addressed previously, these conditions are local to pairs of phrases, dependencies or tokens across the input sentences. Valid and consistent alignment structure can therefore easily be enforced through integer and linear constraints in an ILP. We enumerate the necessary constraints here.

Alignment relations over phrase pairs must avoid overlapping in tokens in order to ensure a consistent segmentation for phrase-based alignment. This can be expressed by straightforward constraints on all phrase variables that cover a particular token.

$$\sum_{\substack{i,j: \\ i \leq p \leq j}} \sum_{k,l} v_{ijkl} \leq 1, \quad \forall 1 \leq p \leq n_1 \quad (8.3)$$

$$\sum_{\substack{k,l: \\ k \leq q \leq l}} \sum_{i,j} v_{ijkl} \leq 1, \quad \forall 1 \leq q \leq n_2 \quad (8.4)$$

These constraints—along with the integer constraints on v variables—are sufficient to guarantee a valid phrase-based alignment along the lines of previous work (DeNero and Klein, 2008; MacCartney et al., 2008; Thadani and McKeown, 2011a). A valid dependency align-

ment requires similar constraints on all edge alignment variables w associated with a dependency in the input.

$$\sum_l w_{jl} \leq 1, \quad \forall 1 \leq j \leq n_1 \quad (8.5)$$

$$\sum_j w_{jl} \leq 1, \quad \forall 1 \leq l \leq n_2 \quad (8.6)$$

It is generally as important to avoid spurious relations in alignment problems as it is to identify appropriate ones. For this reason, it is often helpful to distinguish phrases and dependencies which participate in an output alignment from *poor* alignment candidates in the parameterization, as is done in MacCartney et al. (2008). Practically, this merely requires the inclusion of additional binary variables which indicate the *absence* of phrases and dependency edges in an output alignment.

Specifically, we can use binary variables $v_{ij..}$ to indicate that the phrase $\langle t_i^1, \dots, t_j^1 \rangle \in \mathcal{V}_1$ remains unaligned⁷ and similarly $v_{..kl}$ to indicate that $\langle t_k^2, \dots, t_l^2 \rangle \in \mathcal{V}_2$ remains unaligned, with corresponding scores $\Delta_{\text{phr}}(\mathcal{S}_1, \mathcal{S}_2, i, j, \epsilon, \epsilon)$ and $\Delta_{\text{phr}}(\mathcal{S}_1, \mathcal{S}_2, \epsilon, \epsilon, k, l)$ in the objective from (8.2). The phrase alignment constraints (8.3)–(8.4) can now be replaced by the following equality constraints which ensure that every token participates in either a phrase alignment or in an unaligned phrase.

$$\sum_{\substack{i,j: \\ i \leq p \leq j}} v_{ij..} + \sum_{k,l} v_{ijkl} = 1, \quad \forall 1 \leq p \leq n_1 \quad (8.7)$$

$$\sum_{\substack{k,l: \\ k \leq q \leq l}} v_{..kl} + \sum_{i,j} v_{ijkl} = 1, \quad \forall 1 \leq q \leq n_2 \quad (8.8)$$

A similar modification can be introduced to the dependency alignment problem using binary variables $w_j.$ and $w_{.l}$ to denote unaligned dependencies from \mathcal{Z}_1 and \mathcal{Z}_2 respectively, a corresponding revision to the edge alignment scoring function Δ_{edg} and the following

⁷Note that unaligned text need not be parameterized as full phrases; for instance, MacCartney et al. (2008) considers fixed-length phrases for alignment relations alongside the tokens in unaligned text. In general, the flexibility of an ILP formulation allows us to consider phrases of varying length as well as *discontinuous* phrases as needed.

equality constraints to replace (8.5)–(8.6).

$$w_j + \sum_l w_{jl} = 1, \quad \forall 1 \leq j \leq n_1 \quad (8.9)$$

$$w_l + \sum_j w_{jl} = 1, \quad \forall 1 \leq l \leq n_2 \quad (8.10)$$

Finally, we turn to the problem of making the phrase-based and dependency alignments consistent with each other. In order to connect the two configurations, we introduce auxiliary binary variables ξ_{pq} which indicate that the token $t_p^1 \in \mathcal{T}_1$ and the token $t_q^2 \in \mathcal{T}_2$ participate in an aligned phrase pair.

$$\xi_{pq} = \begin{cases} 1, & \text{iff } \exists i, j, k, l \text{ s.t. } v_{ijkl} = 1, \ i \leq p \leq j, \ k \leq q \leq l \\ 0, & \text{otherwise} \end{cases} \quad (8.11)$$

Additional constraints are required to impose this definition on ξ variables. By definition, an active token pair indicator must participate in exactly one phrase alignment relation. Equivalently, a phrase alignment relation must activate token pair indicators for all its covered tokens.

$$\sum_{\substack{i,j: \\ i \leq p \leq j}} \sum_{\substack{k,l: \\ k \leq q \leq l}} v_{ijkl} = \xi_{pq}, \quad \begin{array}{l} \forall 1 \leq p \leq n_1, \\ 1 \leq q \leq n_2 \end{array} \quad (8.12)$$

In order to ensure that phrase-based and dependency-based alignments remain consistent, the edge alignment relations must activate token pair indicators for their governors and dependents. In combination with (8.12), this ensures that these governors and dependents also participate in phrases that are aligned to each other.

$$w_{jl} - \xi_{jl} \leq 0, \quad \begin{array}{l} \forall 1 \leq j \leq n_1, \\ 1 \leq l \leq n_2 \end{array} \quad (8.13)$$

$$w_{jl} - \xi_{\pi(j)\pi(l)} \leq 0, \quad \begin{array}{l} \forall 1 \leq j \leq n_1, \\ 1 \leq l \leq n_2 \end{array} \quad (8.14)$$

where $\pi(i)$ indicates the index of the token which governs t_i^r in the dependency parse \mathcal{Z}_r of the respective input sentence. The root nodes of both input sentences are assumed to always be aligned, i.e., $\xi_{00} = 1$.

Note however that the inverse of this implication is not enforced by the above constraints, i.e., if the governors and dependents of two dependencies are aligned, the dependencies may not necessarily be aligned to each other. An additional set of constraints is needed to enforce this behavior.

$$\xi_{jl} + \xi_{\pi(j)\pi(l)} - w_{jl} \leq 1, \quad \forall 1 \leq j \leq n_1, \quad (8.15)$$

$$1 \leq l \leq n_2$$

With these constraints, the solution to an ILP with the objective from (8.2) is guaranteed to contain a valid phrase-based alignment in \mathbf{v} and a valid dependency alignment in \mathbf{w} such that dependency edges are only aligned if the tokens they link participate in aligned phrases. Assuming phrases are contiguous, the ILP requires $O(n_1n_2)$ constraints and $O(\delta^2n_1n_2)$ variables where δ is the maximum number of tokens in an alignable phrase.

8.3 Features

The scoring function in (8.2) that guides alignment inference is a linear function over the feature maps for each substructure, i.e., phrase alignment features ϕ_{phr} and edge matching features ϕ_{edg} .

8.3.1 Phrase alignment features

The feature templates enumerated below are used in the feature map $\phi_{\text{phr}}(\mathcal{S}_1, \mathcal{S}_2, i, j, k, l)$ for an alignment between $\langle t_i^1, \dots, t_j^1 \rangle$ from text segment \mathcal{S}_1 to $\langle t_k^2, \dots, t_l^2 \rangle$ from \mathcal{S}_2 . We assume without loss of generality that these phrases are contiguous and bounded in length by some pre-determined limit $\delta \geq 1$ and that phrases which remain unaligned are limited to single tokens, e.g., for an unaligned token from \mathcal{S}_1 , $i = j$ while $k = l = \epsilon$. These features are drawn largely from MacCartney et al. (2008) and were also used in our initial work on phrase-based alignment in Thadani and McKeown (2011a). They consist of the following templates over the input phrases:

- Whether one of the spans does not participate in the alignment, i.e., whether either i and j or k and l are set to ϵ .

- The average size of the two token spans if both participate in the alignment.
- The difference in size between the two token spans if both participate in the alignment.
- Whether $\langle t_i^1, \dots, t_j^1 \rangle$ forms a constituent in a shallow parse (chunking) of \mathcal{S}_1 , and separately whether $\langle t_k^2, \dots, t_l^2 \rangle$ forms a constituent in a shallow parse of \mathcal{S}_2 . We use TagChunk⁸ to generate POS tags and chunks over the input text segments.
- If the two phrases consist of single tokens, whether these tokens are near-identical—determined by stem/lemma overlap, abbreviation fitting and Wordnet synonymy as in §7.2.2.1.
- The maximum similarity between the two phrases using the following normalized similarity measures: (a) a case-insensitive string match of the phrases, (b) for phrases which consist of single tokens, the previous measure for matching near-identical tokens, (c) for phrases which consist of single tokens with at least one referring to a collapsed named entity, the Jaccard coefficient over their components, (d) for phrases of sufficient length, e^{-d} where d is the normalized Levenshtein edit distance between the phrases, (e) the maximum Wordnet path similarity and (f) the maximum Jiang-Conrath similarity over all pairs of Wordnet synsets evoked by the input phrases.

as well as features that capture the context of these phrases:

- The maximum similarity measure over the tokens t_{i-1}^1 and t_{k-1}^2 which precede the input phrases in \mathcal{S}_1 and \mathcal{S}_2 respectively.
- The maximum similarity measure over the tokens t_{j+1}^1 and t_{l+1}^2 which follow the phrases in \mathcal{S}_1 and \mathcal{S}_2 respectively.
- A measure of positional distortion expressed as the absolute distance between the relative positions of the midpoints of the phrases in their respective text segments, i.e., $|(i + j)/2n_1 - (k + l)/2n_2|$ where n_1 and n_2 are the number of tokens in \mathcal{S}_1 and \mathcal{S}_2 respectively.

⁸TagChunk: <http://www.umiacs.umd.edu/~hal/TagChunk/>

8.3.2 Edge matching features

The following feature templates are used in the feature map $\phi_{\text{edg}}(\mathcal{S}_1, \mathcal{S}_2, i, k)$ for an alignment between dependency edges $\langle t_i^1, t_j^1 \rangle$ from the Stanford dependency parse of \mathcal{S}_1 to $\langle t_k^2, t_l^2 \rangle$ from the parse of \mathcal{S}_2 . These features are based entirely on the dependency labels of the input dependencies, inspired in part by the corpus analysis from §8.1.1. To limit overfitting, we group labels together according to the hierarchy of Stanford dependency labels.⁹ Specifically, we group the descendants of *subj*, *obj*, *comp* excluding *obj* and its descendants, *mod* excluding $\{det, predet\}$ and $\{poss, possessive\}$ —each of which are paired as shown. These are used in edge indicator features which record:

- For a non-alignment relation (i.e., $j = \epsilon$ or $l = \epsilon$), the deepest label group in the collapsed hierarchy described above which contains the label of this edge.
- For an alignment relation, a conjugation of the deepest label groups from the hierarchy above which contain their labels.

While the evaluation in §8.4 proceeds with these features, it should be noted that we have identified promising new features in followup experiments.¹⁰ Specifically, we observed notable improvements in results when including dependency features based on POS tags, local structure and similarity measures over token stems. We intend to conduct a more comprehensive and controlled examination of alignment features—including variations over the underlying parsing tools and formalisms—in future work on this task.

8.4 Experiments

To evaluate the proposed approach to multi-structured alignment, we conducted alignment experiments over the Edinburgh corpus described in §8.1. The corpus features 715 training instances and 305 test instances and we used 30% of the training dataset for development experiments. We report macro-averaged precision, recall and F_1 for all token pairs within

⁹Stanford dependencies manual (de Marneffe and Manning, 2008): http://nlp.stanford.edu/software/dependencies_manual.pdf

¹⁰We owe these observations to additional experiments by Michael White, a collaborator on this work.

aligned phrases as well as aligned dependency edge pairs obtained by projecting reference alignments onto the Stanford parses of \mathcal{S}_1 and \mathcal{S}_2 . In addition, we include the percentage of results which exactly match the phrase-based alignment of the references.

The following systems were considered in the experiments:

- **Meteor**: The Meteor alignment tool (Denkowski and Lavie, 2011) using its precision-focused *max accuracy* setting, which we found to yield higher F_1 on the training dataset than the *max coverage* option.
- **ILP-phr**: An ILP for phrase-based alignment which ignores dependency edges, equivalent to the approach presented in Thadani and McKeown (2011a).¹¹ This is an optimal variant of the inference objective from MacCartney et al. (2008) which outperforms a number of other alignment techniques (Och and Ney, 2003; Liang et al., 2006b; Chambers et al., 2007).
- **ILP-phr-edg**: The full ILP for multi-structured alignment over phrases and dependency edges as described in §8.2.3.

We trained models using the structured perceptron from §3.5 for 20 epochs.¹² Separate models were trained on the training dataset with just the SURE alignments and with the SURE+POSSIBLE alignments.¹³

8.4.1 Confident alignments

Table 8.1 contains macro-averaged results from the various systems over the test dataset when using only SURE alignments for training and evaluation. It is evident that the supervised aligners have much higher recall than **Meteor**, with some unsurprising loss in precision due to the conservative *max accuracy* matching. This naturally leads to a lower F_1 for recovering both token alignments as well as dependency alignments projected onto Stanford parses, in addition to fewer overall cases of perfect phrase-based alignment.

¹¹Specifically, the ILP *without* syntactic constraints.

¹²Models for multi-structured alignment were trained with the original structured perceptron of Collins (2002) described in Algorithm 1 and not the minibatched variant of Zhao and Huang (2013).

¹³Note that all alignments are considered equally when evaluating on the SURE+POSSIBLE alignments.

Inference	Token pairs			Projected dependencies			All phrases
	P%	R%	F ₁ %	P%	R%	F ₁ %	Acc%
Meteor (DL11)	81.82	71.90	75.49	84.64	58.03	65.60	11.22
ILP-phr (TM11)	74.83	83.25	77.85	76.07	78.42	75.10	12.21
ILP-phr-edg	76.57	83.79	79.20	73.56	84.27	76.30	12.21

Table 8.1: Macro-averaged results for prediction of SURE alignments on the test dataset with respect to aligned token pairs, projected dependency alignments over Stanford parses and entire phrase-based alignment configurations. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

Compellingly, the multi-structured **ILP-phr-edg** improves in both precision and recall on aligned tokens over the phrase-only **ILP-phr**, with the improved precision largely responsible for a statistically significant increase in F_1 . On projected dependency alignments, however, **ILP-phr** retains greater precision and the F_1 improvement for **ILP-phr-edg** is driven entirely by a nearly 6% gain in recall. It therefore appears that parameterizing dependency edges in the objective leads to additional alignments that are more often accurate than noisy and also boosts the precision of aligned token pairs. However, the rate of perfect alignments in the test dataset is statistically indistinguishable for these two systems.

8.4.2 All alignments

We also considered the POSSIBLE alignments in the corpus which are produced with lower annotator agreement and often involve paraphrasing or logical inference. Table 8.2 reports results over the test dataset with SURE+POSSIBLE alignments for training and evaluation. The trends observed in the SURE-only evaluation are largely repeated in this setting albeit with system differences appearing somewhat exaggerated. For instance, **Meteor** once again leads the aligners in terms of precision, recovering more than 90% of dependency alignments projected from SURE+POSSIBLE phrase alignments. However, the larger fraction of paraphrasal alignments in this setting leads to lower recall than in the SURE-only setting; consequently all F_1 scores as well as the rate of perfect alignments remain lower for **Meteor** than those of the supervised systems.

Inference	Token pairs			Projected dependencies			All phrases
	P%	R%	F ₁ %	P%	R%	F ₁ %	Acc%
Meteor (DL11)	85.40	64.76	72.32	91.19	51.80	62.57	10.56
ILP-phr (TM11)	70.84	82.54	75.37	80.09	80.74	78.79	13.53
ILP-phr-edg	73.03	84.60	77.57	77.04	88.76	80.92	14.85

Table 8.2: Macro-averaged results for prediction of SURE+POSSIBLE alignments on the test dataset with respect to aligned token pairs, projected dependency alignments over Stanford parses and entire phrase-based alignment configurations. Boldfaced entries are statistically undistinguished from the best result within each column under Wilcoxon’s signed rank test ($p < 0.05$).

As expected, the increase in recall for ILP-phr-edg over ILP-phr is greater in this setting with a statistically significant increase of 2% for token alignments and an 8% improvement for dependency alignments. Differences in precision remain largely unchanged so this translates to higher F₁ on both measures for ILP-phr-edg as well as a significant improvement in the rate of perfect phrase-based alignment solutions. Token alignment F₁ for these systems is lower than in the SURE-only scenario owing to the ambiguity of POSSIBLE alignments—manifested here as lower precision since these systems are less conservative in introducing alignments than Meteor under *max accuracy*. On the other hand, F₁ for projected dependency alignments is higher because the additional dependency alignments induced by POSSIBLE phrase alignments appear to be easier to identify using shallow label features, as illustrated by the example in Figure 8.3.

8.5 Remarks

The main contribution of this chapter is a multi-structured approach to monolingual phrase-based alignment which jointly identifies phrasal and syntactic dependency alignments using exact ILP inference and discriminative structured prediction. Our alignment technique shows significant gains over the well-known Meteor aligner and well-studied phrase-based aligners (MacCartney et al., 2008; Thadani and McKeown, 2011a). While further research in token alignment (Yao et al., 2013a; Sultan et al., 2014) and phrase-based alignment (Yao et al., 2013b) has surpassed our results in recent years, we believe that the more general

multi-structured approach is capable of improving further. Initial experiments¹⁴ with improved features for dependency tree alignments demonstrate that significant performance improvements can be obtained with better features and a more rigorous tuning methodology.

As with the text-to-text generation techniques discussed in the previous chapters, runtime is a concern for ILP-based alignment inference. However, the multi-structured formulation can easily be factored into two subproblems: phrase-based alignment over \mathbf{v} , which is NP-hard (DeNero and Klein, 2008), and dependency alignment over \mathbf{w} , which is a straightforward assignment problem that can be solved in polynomial time with the Hungarian algorithm (Kuhn, 1955; Munkres, 1957). This resembles the extractive compression task—also composed of two sub-problems of which only one is tractable—thereby introducing the possibility of following Chapter 4 in applying dual decomposition to speed up multi-structured alignment through the use of Lagrange multipliers over the ξ variables.

While standalone monolingual alignment is an interesting testbed for multi-structured inference, it also has many practical applications in text-to-text generation tasks. For instance, recovering the derivations of reference solutions for compression (cf. §3.5.2), fusion (cf. §7.3.3) and other text-to-text problems such as paraphrase generation are all variations on the monolingual alignment problem. Alignment can also be used in the identification of redundant content for fusion problems (cf. 7.2.2) instead of clustering in order to recover concepts which are phrases and not simply open-class tokens. Finally, statistical alignment can also be used in the evaluation of system output when working with reference solutions that are not reachable under lexically-constrained models, i.e., those which do not admit paraphrases and reordering. We aim to better exploit the natural synergies of alignment and generation problems through further integration of systems for the two tasks. As an example, we expect that alignment can be used to acquire additional training data for compression and that compression able to generate additional training examples for alignment, raising the question of whether it is possible to simultaneously train or adapt semi-supervised systems for *both* tasks in resource-constrained scenarios.

¹⁴Experiments conducted by Michael White exhibit a 4-6% improvement in token alignment F_1 over the Edinburgh corpus thanks to improved corpus pre-processing, with further improvements indicated in preliminary experiments using novel syntactic features.

Chapter 9

Conclusions

In this dissertation, we examine the role of common structural representations and combinations thereof in high-level structured prediction problems such as sentence compression, sentence fusion and text alignment. In particular, we largely focus on joint inference over phrasal or n-gram factorizations of a token sequence as well as edge and second-order factorizations of its dependency tree. We presented a number of models that synthesize new sentences—and, in one case, alignments over sentences—using these joint factorizations and show that they often capture properties of text which are lacking in single-perspective models. A brief overview of the broad technical contributions follows:

- In Chapter 3, we describe a multi-structured objective for the *sentence compression* problem which assembles an output sentence through both a sequence of n-grams and an edge-factored dependency tree. In order to recover feasible output sentences, we develop a flexible integer linear program (ILP) which guarantees optimal solutions for this joint objective. Even though n-gram features and dependency features have access to the same underlying signals (lexical information, POS tags and dependency labels), our experiments indicate that joint inference over both structures results in improved performance on all corpora over single-structure approaches. This advantage is observed to persist when evaluating at different compression rates, and we note that multi-structured inference is particularly beneficial when compression rates are aggressive. To our knowledge, this technique yields state-of-the-art performance for

extractive compression over the datasets described in §3.1.¹

- In order to overcome the practical limitations of the previous compression approach—namely its dependence on ILPs whose general intractability can hamper real-world implementations—Chapter 4 presents approximate inference algorithms for the same objective by separately addressing the n-gram and dependency subproblems through dual decomposition. While the former subproblem can be addressed efficiently through a restriction to order-preserving bigrams, the problem of recovering optimal non-projective subtrees in directed graphs remains intractable and must be approximated. Our experiments indicate that the use of subgradient-based dual decomposition for multi-structured inference results in a decrease in average runtime over the ILP approach—which relies on a highly optimized multi-core solver—despite no effort toward code optimization.
- In both these compression approaches, exact solutions cannot be guaranteed in polynomial time because they accommodate non-projectivity in the output dependency trees and can also support reordering of the input tokens. However, these aspects are not needed for extractive compression in a mostly projective language like English. Chapter 5 extends the order-preserving assumption to both output structures and describes a generalization of a well-known dynamic programming algorithm for projective dependency parsing that can recover an optimal compressed dependency tree as well as a corresponding bigram factorization for an input sentence. This polynomial-time approach leads to significantly improved runtime for multi-structured compression with no meaningful decrease in output quality. An extension to second-order dependencies is also viable with no asymptotic increase in runtime, resulting in moderate performance improvements which we believe can be further improved with additional training data.
- While the relaxed and tractable techniques discussed previously are motivated by improving runtime, the flexible integer programming approach readily supports ex-

¹Qian and Liu (2014) report similarly strong results for an independent implementation of our approach using different features and a different split of the WN corpus.

tensions such as the use of additional structured representations of text. In Chapter 6, we attempt to incorporate predicate-argument structures for the output sentence by extending the flow-based inference strategy to various forms of directed acyclic graphs (DAGs). This representation remains compatible with various well-known shallow semantic representations including the predicate-based Propbank, its recent extension to full-sentence semantics in the Abstract Meaning Representation (AMR), and the frame-semantic formalism underlying the Framenet project. In addition, we conduct experiments incorporating Framenet structures alongside n-grams and dependency trees. Although we do not find improvements over the extractive compression datasets, we conjecture that performance gains remain a possibility for representations which are closer to surface lexicalizations such as AMR, which can be characterized by DAG-structured parses over concepts and their arguments.

- Extending the multi-structured compression approach in another direction, Chapter 7 explores the related task of *sentence fusion*, which combines multiple input sentences. Developing empirical models for multi-structured requires a source of training data—which we produce using the human annotations from shared-task datasets for summarization evaluation—and an inference approach which can draw information from multiple input sentences and order it into a coherent sentence, which we find in the compression system discussed previously. Rather than align the input sentences prior to lexicalization (and consequently risk propagating errors), we propose a single-stage inference approach which identifies redundancy in the input sentences in order to define features that aid content selection, while restricting redundancy in the output sentence through constraints over groups of similar input tokens. Our experiments indicate that multi-structured fusion offers a strong improvement over bigram-based fusion and that dependency-only models are overwhelmed by the ambiguity in the fusion task. We also uncover mild performance gains by parameterizing dependencies differently based on their orientation in an output sentence.
- Finally, we consider the utility of a multi-structured approach to inference for *phrase-based alignment*—a task closely related to many aspects of text-to-text generation—in

Chapter 8. Our supervised framework recovers phrase-based alignments between two sentences alongside a consistent mapping of dependency edges from the parse trees over the sentences. We find that the inclusion of syntactic structure in the objective results in clear improvements over phrase-based alignment in isolation, which has itself previously been shown to outperform a number of monolingual and cross-lingual alignment techniques. This formulation appears amenable to further improvements on evaluation measures—through better syntactic features—as well as improvements in runtime with an approximate inference strategy based on dual decomposition.

9.1 Limitations

Our investigation of multi-structured inference was limited in the following aspects owing to time and resource constraints. We hope to address these issues in future research.

9.1.1 Datasets

We limited our compression experiments to the commonly used datasets of Clarke and Lapata (2006b) and Clarke and Lapata (2007). However, recent work by Filippova and Altun (2013) has raised the possibility of using large-scale automatically-constructed datasets in compression research, in this case by using the headlines of news articles to reliably guide automated compressions of their leading sentences. Although this approach produces reference compressions using a subtree-pruning assumption—observed to be violated frequently in the BN and WN datasets by §3.1.1 and Qian and Liu (2014)—it is possible that growing the dataset by two orders of magnitude will compensate for noise in reference compressions.

We cannot immediately infer the performance of multi-structured techniques when a large amount of training data is available. It is possible that simpler models with richer features will perform competitively in such a scenario, thereby restricting the utility of rich multi-structured inference to scenarios in which training resources are limited. However, we suspect that higher-order models such as the trigram models from §3.6.5 and the second-order dependency models from §5.3.3 will also see performance gains when feature sparsity is not a concern. Higher-order models have been observed to improve performance in tasks

such as dependency parsing (McDonald and Pereira, 2006; Koo and Collins, 2010; Pitler, 2014) even when training data is relatively plentiful.

The same caveat applies to the problems of sentence fusion and phrase-based alignment, which rely on corpora which are approximately as large as the compression datasets used here. Automated acquisition of sentence fusion data may be viable from datasets of news articles containing lightly-edited newswire text while monolingual alignment data can possibly be acquired from parallel datasets used in machine translation. We expect that the flexible multi-structured techniques can be modified to take advantage of increased training resources in both tasks, although the efficiency tradeoffs involved in using higher-order models will require further consideration.

9.1.2 Features

The performance of all our supervised discriminative approaches is naturally closely tied to the features defined over various substructures. For our experiments on compression and fusion, we attempted to isolate useful features in ablation tests over the development datasets with varying amounts of training data. However, feature tuning is an inexact science and we expect that further refinement may lead to features with better performance and generalization, especially if accompanied by additional datasets from new domains.

Different feature sets may also result in varying benefits for particular structured representations. For instance, refining the feature set and reference derivation approach for our previously published research on the compression task (Thadani and McKeown, 2013a; Thadani, 2014) led to an outsized gain for the n-gram models relative to the dependency-based models. Although we made an effort to represent various linguistic attributes—open-class words, morphology, POS tags and dependency labels—equally when assigning features for each structured representation,² it nevertheless remains possible that novel features may favor models based on a particular structured representation.

Furthermore, although these linguistic attributes comprise standard sources of features

²Note that this is not the case for the features described in §8.3 for multi-structured alignment. Here, we used a well-studied feature set for phrase-based inference and augmented it with complementary but relatively naïve features for dependency mappings.

in statistical natural language problems, additional indicators based on the distributional properties of text could also be considered within these models. For instance, word clusters based on lexical co-occurrence frequencies over text collections (Brown et al., 1992) have been found to be useful for features in structured tasks such as dependency parsing (Koo et al., 2008). Recent research in lexical semantics has also yielded fast and accurate techniques to recover vector embeddings of words in Euclidean spaces with intriguing regularities under linear transformations (Mikolov et al., 2013a; Mikolov et al., 2013b; Mikolov et al., 2013c). The utility of these unsupervised lexical representations in higher-level text-to-text generation problems remains an open question and a point of interest for future work.

9.1.3 Representations

We use dependency structures throughout this dissertation and rely on the Stanford parser to recover dependency parses that serve as gold-standard dependency trees for training and evaluation. However, since our reference trees are derived from imperfect statistical models, it is likely that the use of alternative statistical parsers will yield different results for the structured prediction tasks considered here. A similar caveat holds for the frame-semantic parses provided by SEMAFOR in Chapter 6 which was observed to be inconsistent in labeling frame elements 6.1.4 across original and compressed sentences—a phenomenon which may partially explain the absence of performance improvements when frames and frame elements are included in the compression objective.

We also did not pursue alternative notions of syntactic representation such as context-free grammars (CFGs) for constituent parse trees. Synchronous grammars have been employed previously for sentence compression (Galley and McKeown, 2007; Cohn and Lapata, 2008; Cohn and Lapata, 2009; Ganitkevitch et al., 2013) and quasi-synchronous grammars (Smith and Eisner, 2006) have been used to incorporate paraphrases in surface realizations for other text-to-text tasks (Woodsend et al., 2010; Woodsend and Lapata, 2011; Woodsend and Lapata, 2012). Although constituency parse structures can technically be accommodated within our ILP framework for text-to-text generation—with non-terminal nodes potentially acquiring coefficients drawn from the parameters of a synchronous CFG—the resulting increase in program size and relatively minimal increase in representative power

renders experimentation in this direction unappealing. However, these models may prove useful as an alternative source of syntactic compression solutions in the dual decomposition technique described in Chapter 4.

9.1.4 Learning algorithms

We trained all models in this dissertation with the structured perceptron, which is described in §3.5.1. While this algorithm is straightforward to implement and analyze, it is limited to optimizing a 0/1 loss over the structured output space and—despite the use of parameter averaging—may have poor generalization. Large-margin techniques such as the structured SVM (Tsochantaridis et al., 2004) and MIRA (Crammer and Singer, 2003; McDonald et al., 2005a) trade off ease of implementation for better generalization and the ability to incorporate non-differentiable cost functions as well as multiple inference hypotheses in the learning procedure. In addition, we do not control the learning rate of the structured perceptron although recent research indicates the benefit of specifying per-feature error rates in gradient updates to better exploit rare but predictive features (Duchi et al., 2011). We expect that a principled application of more advanced learning techniques will improve evaluation results for all inference techniques discussed here.

Furthermore, all our datasets are composed solely of *reachable* problems, i.e., problems with structured output spaces which contain the reference solution. However, such datasets are less likely to exist in text-to-text generation tasks which require lexical choice or paraphrasing. Although paraphrases can be incorporated relatively easily in the inference techniques proposed in this work, the structured perceptron’s 0/1 loss is unsuited to training over unreachable instances. Replacing the loss function with one which is less sensitive to these issues—such as structured ramp loss (Gimpel and Smith, 2012)—is another important consideration for extending these supervised inference techniques to new problems and datasets.

9.1.5 Evaluation

An advantage of working on reachable structured problems is that it remains easy to quantify system performance using automated measures. This is especially true for the extractive

compression scenario in which output tokens do not change their relative ordering from the input sentence, where automated measures have been increasingly adopted (Napoles et al., 2011b). This approach sidesteps the cost and time required for human judgments, which are vulnerable to systematic biases without careful controls and must be discarded when systems are updated, e.g., when promising new features are included. It is therefore preferable to expend human effort on manual annotations, such as those underlying the datasets studied here, while relying on statistical measures for repeatable evaluations.

We can also see the drawbacks of this approach in the case of multi-way fusion in Chapter 7. Every instance in the dataset has a solitary reference fusion but a much vaster output space than an equivalent compression problem, so viable system-generated fusions may be unduly penalized for not recovering the exact surface realizations of their corresponding reference sentences. Although we compensate for this issue by additionally computing the evaluation measures over contributors drawn from the input, this highlights the limitations of working with datasets without multiple references per instance for non-trivial text-to-text problems, particularly if the references are not reachable. Therefore, although automated measures can suffice for comparing system performance on a particular dataset, human evaluations will ultimately be necessary if the output from these systems is intended to stand in for human-authored text.

9.2 Future Work

Despite the limitations enumerated above, multi-structured inference appears to be a useful notion for formulating and reasoning about practical natural language problems. The following categories summarize our ongoing and future extensions of this work.

9.2.1 Unifying text-to-text operations

This research is intended to form the foundational components of a generic monolingual text-to-text framework. By pairing supervised structured prediction with expressive features over natural language structures, many text-to-text applications can be addressed through the same approach given an adequate amount of training data. Furthermore, extensions to new

structured representations, learning algorithms and inference techniques will likely benefit more than one text-to-text application.

In this dissertation, we have largely considered challenges in *intrinsic* content selection—the identification of the most salient content within the input without external guidance—and syntactic manipulation—in order to produce grammatical output sentences which are similar to references. Content selection can easily be extended via salience features to extrinsic information such as queries in query-based fusion (Krahmer et al., 2008) or image descriptors for image caption generation (Yang et al., 2011; Mitchell et al., 2012). Similarly, since some of the proposed inference strategies accommodate full output spaces for n-grams and dependency trees, the only limitation for syntactic variation is the factorization of these structures chosen for a particular task. For instance, we expect that the second-order dependencies supported by the dynamic programs from Chapter 5 will suffice for most applications similar to sentence compression but, if more intricate transformations require higher-order dependencies or *all* siblings of a dependency to be parameterized, this can be incorporated into ILP and dual decomposition inference.

Furthermore, many practical text-to-text transformations rely on specific forms of paraphrasing, e.g., sentence simplification can broadly be viewed as sentence compression paired with lexical simplification. In addition to accommodating syntactic variation, all systems described in this work are capable of admitting lexical paraphrases, including—to a limited degree—the compressive parsing approach in Chapter 5. We can therefore incorporate paraphrase candidates drawn from training corpora, general resources such as the PPDB (Ganitkevitch et al., 2013) as well as task-specific directional paraphrases in text-to-text problems. This straightforward extension will make the proposed multi-structured framework applicable to a wide variety of problems including lexico-syntactic transformations like simplification and grammatical error correction as well as higher-level applications such as the generation of natural language entailments or stylistic variations on a text.

9.2.2 Direct applications

A natural application of our text-to-text framework is in single and multi-document summarization. A summarization algorithm that is able to compress sentences can make better

use of summary budgets—an observation that has motivated much research in combining sentence compression and selection for summaries (Daumé and Marcu, 2002; Clarke and Lapata, 2007; Madnani et al., 2007; Zajic et al., 2007; Gillick and Favre, 2009; Liu and Liu, 2009; Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Chali and Hasan, 2012; Genest and Lapalme, 2012; Woodsend and Lapata, 2012; Almeida and Martins, 2013; Li et al., 2013; Molina et al., 2013; Morita et al., 2013; Qian and Liu, 2013; Wang et al., 2013; Kikuchi et al., 2014; Li et al., 2014). Recent methods have focused on joint inference using ILPs or fast dual decomposition but rely on simple subtree-deletion models for the compression task itself. We are keen to investigate the incorporation of richer multi-structured compression models within a summarization objective.

Similarly, sentence fusion was originally devised to improve multi-document summarization (Barzilay and McKeown, 2005) by exploiting partial redundancy between related sentences from different documents. However, fusion in this context relies on pipelined stages and is therefore vulnerable to error propagation. We are interested in pursuing a joint formulation of fusion over multiple sentences within summarization over multiple documents which avoids these issues. Furthermore, recent work has proposed compelling extensions of the fusion task such as sentence *enhancement* (Cheung and Penn, 2014)—the combination of *dissimilar* sentences in abstractive summarization—which may also benefit from joint inference and supervised learning.

Other immediate applications of this framework include image caption generation, which may be viewed as a form of sentence fusion across captions from training images³ with image-specific features as an extrinsic source of salience, and grammatical error correction, which involves lexico-syntactic transformations which may benefit from joint inference under n-gram and syntactic structures. More generally, sentence realization in natural language generation (NLG) systems can be viewed as a multi-stage problem which starts with a DAG representation of semantic content and refines it into a syntactic tree which is then linearized into a sentence and finally modified with appropriate morphology (Bohnet et al., 2010). However, each of these stages can be combined into a single step following Chapter 6, thereby supporting the use of a common objective for general NLG problems.

³A suggestion by Hal Daumé III.

9.2.3 Task-based evaluations

As noted in §9.1.3, structural analyses over reference sentences can be noisy—as in the case of Stanford and SEMAFOR parses—and the choice of which representation or tool is used for gold-standard reference structures may affect output sentences in a measurable way. This naturally invites the question: which representations and tools are most appropriate for use in tasks such as text-to-text generation and alignment? Although this may be asked of any system which relies on syntactic *features* from these structures, we believe that structured inference tasks described here are far more sensitive to the variations across formalisms and the errors made by specific tools.

We aim to explore this by considering the effect of varying dependency parsers in a task-based evaluation with multi-structured monolingual alignment.⁴ Phrase-based and dependency alignment solutions are easy to evaluate and analyze given reference alignments. By varying the dependency structure used in multi-structured alignment, we can observe both the standalone contribution of a particular type of parse structure to the alignment problem as well as its marginal contribution when paired with phrase-based inference. Furthermore, this naturally leads to an exploration of which dependency structures make orthogonal alignment errors and whether multiple syntactic formalisms can be paired to reduce overall error on the alignment task.

9.2.4 Multi-task learning

Finally, although we have thus far discussed applications in isolation, it is not unreasonable to expect that learned parameters for different datasets, domains and even different text-to-text tasks may be largely similar. Saliency features may vary in each of these situations but the feature subspace which is concerned with the fluency of an output sentence may benefit from a shared parameterization and additional training data from multiple domains and tasks. The availability of an inference framework which applies to multiple text-to-text generation problems allows us to explore a number of hypotheses around shared or co-regularized parameters for structured output spaces.

⁴Joint work with Michael White.

Multi-task learning may also be considered *within* a particular structured representation. For instance, because an optimal dependency parse must be recovered within the space of all possible parse trees for an output sentence, multi-structured compression serves as a generalization of dependency parsing—a fact made explicit by the use of a well-studied parsing algorithm for the task in Chapter 5. This naturally invites the question of whether tree-banks can be directly used to refine the parameters of dependency edge variables rather than syntactic features drawn from parses over the input. We are therefore examining whether multi-task learning over text-to-text generation and problems such as dependency parsing can lead to performance improvements on either task.⁵ As further structured representations such as AMR are introduced within this text-to-text framework, structure-specific multi-task learning may become viable over the relevant training resources.

In our view, these extensions serve to further bridge the gap between natural language analysis and generation, allowing us to further both aspects over time and establish clear evidence of progress when working with newer structured formalisms. Much work remains to be done toward each of these problems but we remain hopeful that the foundational steps introduced in this dissertation remain flexible and general enough to serve these goals.

⁵An ongoing project with Alexander Rush.

Bibliography

- Miguel Almeida and André F. T. Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of ACL*, pages 196–206.
- Mandya Angrosh and Advaith Siddharthan. 2014. Text simplification using synchronous dependency grammars: Generalising automatically harvested rules. In *Proceedings of INLG*, pages 16–25.
- Mandya Angrosh, Tadashi Nomoto, and Advaith Siddharthan. 2014. Lexico-syntactic text simplification and compression with typed dependencies. In *Proceedings of COLING*, pages 1996–2006.
- Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley Framenet project. In *Proceedings of ACL-COLING*, pages 86–90.
- Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval’07 task 19: Frame semantic structure extraction. In *Proceedings of the International Workshop on Semantic Evaluations*, pages 99–104.
- Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186. Association for Computational Linguistics.
- Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of COLING*, pages 42–48.
- Srinivas Bangalore, Owen Rambow, and Steve Whittaker. 2000. Evaluation metrics for generation. In *Proceedings of INLG*, pages 1–8.
- Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of ACL*, pages 318–325.
- Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of ACL*, pages 597–604.
- Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of HLT-NAACL*, pages 16–23.
- Regina Barzilay and Kathleen R. McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of ACL*, pages 50–57.
- Regina Barzilay and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.
- Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of ACL*, pages 550–557.

- Anja Belz, Mike White, Josef van Genabith, Deirdre Hogan, and Amanda Stent. 2010. Finding common ground: Towards a surface realisation shared task. In *Proceedings of INLG*, pages 268–272.
- Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of ACL-HLT*, pages 481–490.
- Bernd Bohnet, Leo Wanner, Simon Mill, and Alicia Burga. 2010. Broad coverage multilingual deep sentence generation with a stochastic multi-level realizer. In *Proceedings of COLING*, pages 98–106.
- Stefan Bott and Horacio Saggion. 2011. An unsupervised alignment algorithm for text simplification corpus construction. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 20–26.
- Houda Bouamor, Aurélien Max, and Anne Vilnat. 2011. Monolingual alignment by edit rate computation on sentential paraphrase pairs. In *Proceedings of ACL-HLT*, pages 395–400.
- Florian Boudin and Emmanuel Morin. 2013. Keyphrase extraction for n-best reranking in multi-sentence compression. In *Proceedings of NAACL-HLT*, pages 298–305.
- Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the ACL-COLING Interactive Presentation Sessions*.
- Chris Brockett. 2007. Aligning the 2006 RTE corpus. In *Technical Report MSR-TR-2007-77, Microsoft Research*.
- Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.
- Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, pages 149–164.
- Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of EMNLP*, pages 196–205.
- Xavier Carreras and Lluís Màrquez. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL*, pages 89–97.
- Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL*, pages 152–164.
- Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of EMNLP-CoNLL*.
- John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying text for language-impaired readers. In *Proceedings of EACL*, pages 269–270.

- Yllias Chali and Sadid A. Hasan. 2012. On the effectiveness of using sentence compression models for query-focused multi-document summarization. In *Proceedings of COLING*, pages 457–474.
- Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yeh, and Christopher D. Manning. 2007. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 165–170.
- Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. 2010. Discriminative learning over constrained latent representations. In *Proceedings of HLT-NAACL*, pages 429–437.
- Jackie Chi Kit Cheung and Gerald Penn. 2014. Unsupervised sentence enhancement for automatic summarization. In *Proceedings of EMNLP*, pages 775–786.
- David Chiang, Jacob Andreas, Daniel Bauer, Moritz Karl Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proceedings of ACL*, pages 924–932.
- David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.
- Yoeng-jin Chu and Tseng-hong Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.
- James Clarke and Mirella Lapata. 2006a. Constraint-based sentence compression: an integer programming approach. In *Proceedings of ACL-COLING*, pages 144–151.
- James Clarke and Mirella Lapata. 2006b. Models for sentence compression: a comparison across domains, training requirements and evaluation measures. In *Proceedings of ACL-COLING*, pages 377–384.
- James Clarke and Mirella Lapata. 2007. Modelling compression with discourse constraints. In *Proceedings of EMNLP-CoNLL*, pages 1–11.
- James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: an integer linear programming approach. *Journal for Artificial Intelligence Research*, 31:399–429.
- James Clarke and Mirella Lapata. 2010. Discourse constraints for document compression. *Computational Linguistics*, 36(3):411–441.
- Trevor Cohn and Mirella Lapata. 2007. Large margin synchronous generation and its application to sentence compression. In *Proceedings of EMNLP*, pages 73–82.
- Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of COLING*, pages 137–144.
- Trevor Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34(1):637–674.

- Trevor Cohn, Chris Callison-Burch, and Mirella Lapata. 2008. Constructing corpora for the development and evaluation of paraphrase systems. *Computational Linguistics*, 34(4):597–614.
- Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*.
- Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8.
- Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.
- Dipanjan Das and Noah A. Smith. 2011. Semi-supervised frame-semantic parsing for unknown predicates. In *Proceedings of ACL*, pages 1435–1444.
- Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. Probabilistic frame-semantic parsing. In *Proceedings of HLT-NAACL*, pages 948–956.
- Dipanjan Das, André F. T. Martins, and Noah A. Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 209–217.
- Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2013. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.
- Hal Daumé, III and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of ACL*, pages 449–456.
- Hal Daumé III and Daniel Marcu. 2004. Generic sentence fusion is an ill-defined summarization task. In *Proceedings of the ACL Text Summarization Branches Out Workshop*, pages 96–103.
- Harold Charles Daumé III. 2006. *Practical Structured Learning Techniques for Natural Language Processing*. Ph.D. thesis, Los Angeles, CA, USA.
- Marie-Catherine de Marneffe and Christopher D. Manning, 2008. *Stanford Dependencies Manual*. Stanford University.
- John DeNero and Dan Klein. 2008. The complexity of phrase alignment problems. In *Proceedings of ACL-HLT*, pages 25–28.
- John DeNero and Klaus Macherey. 2011. Model-based aligner combination using dual decomposition. In *Proceedings of ACL-HLT*, pages 420–429.
- Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91.
- George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of HLT*, pages 138–145.

- Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge Trimmer: a parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL DUC03 Workshop on Text Summarization*, pages 1–8.
- Mark Dras. 1997. Reluctant paraphrase: Textual restructuring under an optimisation model. In *Proceedings of PacLing*, pages 98–104.
- Mark Dras. 1998. Search in constraint-based paraphrasing. In *Proceedings of the Second International Conference on Natural Language Processing and Industrial Applications*.
- John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.
- Jack R. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.
- Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of ACL*, pages 457–464.
- Jason Eisner and Noah A. Smith. 2010. Favor short dependencies: Parsing with soft and hard constraints on dependency length. pages 121–150. Springer Netherlands.
- Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of COLING*, pages 340–345.
- Micha Elsner and Deepak Santhanam. 2011. Learning to fuse disparate sentences. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 54–63.
- Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of EMNLP*, pages 1481–1491.
- Katja Filippova and Michael Strube. 2008a. Dependency tree based sentence compression. In *Proceedings of INLG*, pages 25–32.
- Katja Filippova and Michael Strube. 2008b. Sentence fusion via dependency graph compression. In *Proceedings of EMNLP*, pages 177–185.
- Katja Filippova. 2010. Multi-sentence compression: finding shortest paths in word graphs. In *Proceedings of COLING*, pages 322–330.
- Charles J. Fillmore, Charles Wooters, and Collin F. Baker. 2001. Building a large lexical databank which provides deep semantics. In *Proceedings of the Pacific Asia Conference on Language, Information and Computation*, pages 3–26.
- Charles J. Fillmore, Christopher R. Johnson, and Miriam R.L. Petruck. 2003. Background to framenet. *International Journal of Lexicography*, 16(3):235–250.
- Charles J. Fillmore. 1982. Frame semantics. *Linguistics in the Morning Calm*, pages 111–137.

- Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*, pages 363–370.
- Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of ACL*, pages 1426–1436.
- Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.
- Dimitrios Galanis and Ion Androutsopoulos. 2010. An extractive supervised two-stage method for sentence compression. In *Proceedings of HLT-NAACL*, pages 885–893.
- Dimitrios Galanis and Ion Androutsopoulos. 2011. A new sentence compression dataset and its use in an abstractive generate-and-rank sentence compressor. In *Proceedings of the UCNLG+Eval: Language Generation and Evaluation Workshop*, pages 1–11.
- Michel Galley and Kathleen McKeown. 2007. Lexicalized Markov grammars for sentence compression. In *Proceedings of HLT-NAACL*, pages 180–187.
- Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles, and Benjamin Van Durme. 2011. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proceedings of EMNLP*, pages 1168–1179.
- Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of HLT-NAACL*, pages 758–764.
- Pierre-Etienne Genest and Guy Lapalme. 2012. Fully abstractive approach to guided summarization. In *Proceedings of ACL*, pages 354–358.
- Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 10–18.
- Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proceedings of NAACL-HLT*, pages 221–231.
- Michael Heilman and Nitin Madnani. 2012. ETS: Discriminative edit models for paraphrase scoring. In *Proceedings of *SEM: The First Joint Conference on Lexical and Computational Semantics*, pages 529–535.
- Chiori Hori and Sadaoki Furui. 2004. Speech summarization: an approach through word extraction and a method for evaluation. *IEICE Transactions on Information and Systems*, E87-D(1):15–25.
- Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL*, pages 144–151.
- Liang Huang and Suphan Feyong. 2012. Structured perceptron with inexact search. In *Proceedings of HLT-NAACL*.

- Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of EMNLP*, pages 273–283.
- Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*.
- Hongyan Jing and Kathleen R. McKeown. 2000. Cut and paste based text summarization. In *Proceedings of NAACL*, pages 178–185.
- Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the Conference on Applied Natural Language Processing*, pages 310–315.
- Yuta Kikuchi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2014. Single document summarization based on nested tree structure. In *Proceedings of ACL*, pages 315–320.
- Beata Beigman Klebanov, Kevin Knight, and Daniel Marcu. 2004. Text simplification for information-seeking applications. *On the Move to Meaningful Internet Systems: Lecture Notes in Computer Science*, 3290:735–747.
- Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proceedings of AAAI*, pages 703–710.
- Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.
- Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 48–54.
- Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *Proceedings of ICCV*, pages 1–8.
- Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*, pages 1–11.
- Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-HLT*, pages 595–603.
- Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of EMNLP*, pages 1288–1298.
- Emiel Krahmer, Erwin Marsi, and Paul van Pelt. 2008. Query-based sentence fusion is better defined and leads to more preferred results than generic sentence fusion. In *Proceedings of ACL-HLT*, pages 193–196.
- Matthias Trautner Kromann. 2001. Optimality parsing and local cost functions in discontinuous grammar. *Electronic Notes in Computer Science*, 53:163–179.
- Harold W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

- Alex Kulesza and Fernando Pereira. 2007. Structured learning with approximate inference. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *Proceedings of NIPS*. Curran Associates, Inc.
- Hoong Chuin Lau, Trung Hieu Ngo, and Bao Nguyen Nguyen. 2006. Finding a length-constrained maximum-sum or maximum-density subtree and its application to logistics. *Discrete Optimization*, 3(4):385 – 391.
- Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *Proceedings of EMNLP*, pages 490–500.
- Chen Li, Yang Liu, Fei Liu, Lin Zhao, and Fuliang Weng. 2014. Improving multi-documents summarization by sentence compression based on expanded constituent parse trees. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 691–701.
- Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006a. An end-to-end discriminative approach to machine translation. In *Proceedings of ACL-COLING*, pages 761–768.
- Percy Liang, Ben Taskar, and Dan Klein. 2006b. Alignment by agreement. In *Proceedings of HLT-NAACL*, pages 104–111.
- Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of HLT-NAACL*, pages 912–920.
- Lucian Vlad Lita, Abe Ittycheriah, Salim Roukos, and Nanda Kambhatla. 2003. tRuEcas-Ing. In *Proceedings of ACL*, pages 152–159.
- Fei Liu and Yang Liu. 2009. From extractive to abstractive meeting summaries: Can it be done by sentence compression? In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 261–264.
- Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of ACL-COLING*, pages 609–616.
- Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of HLT-NAACL*, pages 41–48.
- Bill MacCartney, Michel Galley, and Christopher D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP*, pages 802–811.
- Nitin Madnani and Bonnie J. Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.
- Nitin Madnani, David Zajic, Bonnie Dorr, Necip Fazil Ayan, and Jimmy Lin. 2007. Multiple alternative sentence compressions for automatic text summarization. In *Proceedings of the Document Understanding Conference (DUC) at HLT-NAACL*.

- Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of HLT-NAACL*, pages 182–190.
- Thomas L. Magnanti and Laurence A. Wolsey. 1994. Optimal trees. In *Technical Report 290-94, Massachusetts Institute of Technology, Operations Research Center*.
- Erwin Marsi and Emiel Krahmer. 2005. Explorations in sentence fusion. In *Proceedings of the European Workshop on Natural Language Generation*, pages 109–117.
- Erwin Marsi, Emiel Krahmer, Iris Hendrickx, and Walter Daelemans. 2010. On the limits of sentence compression by deletion. In Emiel Krahmer and Mariët Theune, editors, *Empirical Methods in Natural Language Generation*, pages 45–66. Springer-Verlag, Berlin, Heidelberg.
- André F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 1–9.
- André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL-IJCNLP*, pages 342–350.
- André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proceedings of EMNLP*, pages 238–249.
- Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88.
- Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98.
- Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of EMNLP-HLT*, pages 523–530.
- Ryan McDonald, Keith Hall, and Gideon Mann. 2010. Distributed training strategies for the structured perceptron. In *Proceedings of HLT-NAACL*, pages 456–464.
- Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of EACL*, pages 297–304.
- Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of the European conference on IR research*, pages 557–564, Berlin, Heidelberg. Springer-Verlag.
- Kathleen McKeown, Sara Rosenthal, Kapil Thadani, and Coleman Moore. 2010. Time-efficient creation of an accurate sentence fusion corpus. In *Proceedings of HLT-NAACL*, pages 317–320.

- Kathleen R. McKeown. 1983. Paraphrasing questions using given and new information. *Computational Linguistics*, 9(1):1–10.
- Donald Metzler, Eduard Hovy, and Chunliang Zhang. 2011. An empirical evaluation of data-driven paraphrase generation techniques. In *Proceedings of ACL-HLT*, pages 546–551.
- Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The nombank project: An interim report. In Adam Meyers, editor, *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*.
- Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT*, pages 746–751.
- Margaret Mitchell, Xufeng Han, Jesse Dodge, Alyssa Mensch, Amit Goyal, Alex Berg, Kota Yamaguchi, Tamara Berg, Karl Stratos, and Hal Daumé, III. 2012. Midge: Generating image descriptions from computer vision detections. In *Proceedings of EACL*, pages 747–756.
- Alejandro Molina, Juan-Manuel Torres-Moreno, Eric SanJuan, Iria da Cunha, and Gerardo Eugenio Sierra Martínez. 2013. Discursive sentence compression. pages 394–407. Springer.
- Hajime Morita, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2013. Subtree extractive summarization via submodular maximization. In *Proceedings of ACL*, pages 1023–1032.
- James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38.
- Courtney Napoles, Chris Callison-Burch, Juri Ganitkevitch, and Benjamin Van Durme. 2011a. Paraphrastic sentence compression with a character-based metric: tightening without deletion. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 84–90.
- Courtney Napoles, Benjamin Van Durme, and Chris Callison-Burch. 2011b. Evaluating sentence compression: pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 91–97.
- Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, and Dawn Song. 2012. On the feasibility of internet-scale author identification. In *IEEE Symposium on Security and Privacy*.

- Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing*, 4(2).
- Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL*, pages 99–106.
- Tadashi Nomoto. 2007. Discriminative sentence compression with conditional random fields. *Information Processing and Management*, 43(6):1571–1587.
- Tadashi Nomoto. 2009. A comparison of model free versus model intensive approaches to sentence compression. In *Proceedings of EMNLP*, pages 391–399.
- Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29:19–51.
- Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.
- Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: extracting paraphrases and generating new sentences. In *Proceedings of HLT-NAACL*, pages 102–109.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.
- Emily Pitler. 2014. A crossing-sensitive third-order factorization for dependency parsing. *Transactions of the Association for Computational Linguistics*, 2:41–54.
- Xian Qian and Yang Liu. 2013. Fast joint compression and summarization via graph cuts. In *Proceedings of EMNLP*, pages 1492–1502.
- Xian Qian and Yang Liu. 2014. Polynomial time joint structural inference for sentence compression. In *Proceedings of ACL*, pages 327–332.
- Chris Quirk, Chris Brockett, and William B. Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of EMNLP*, pages 142–149.
- Sebastian Riedel and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of EMNLP*, pages 129–137.
- Stefan Riezler, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of HLT-NAACL*, pages 118–125.
- Frank Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.
- Alexander M. Rush and Michael Collins. 2011. Exact decoding of syntactic translation models through Lagrangian relaxation. In *Proceedings of ACL-HLT*, pages 72–82.

- Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of EMNLP*, pages 1–11.
- Kenji Sagae and Jun’ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proceedings of Coling*, pages 753–760.
- Natalie Schluter. 2014. On maximum spanning dag algorithms for semantic dag parsing. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 61–65.
- Advaith Siddharthan and Mandya Angrosh. 2014. Hybrid text simplification using synchronous dependency grammars with hand-written and automatically harvested rules. In *Proceedings of EACL*, pages 722–731.
- Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2004. Syntactic simplification for improving content selection in multi-document summarization. In *Proceedings of COLING*.
- Advaith Siddharthan. 2006. Syntactic simplification and text cohesion. *Research on Language and Computation*, 4(1):77–109.
- Advaith Siddharthan. 2010. Complex lexico-syntactic reformulation of sentences using typed dependency representations. In *Proceedings of INLG*, pages 125–133.
- Advaith Siddharthan. 2011. Text simplification using typed dependencies: A comparison of the robustness of different generation strategies. In *Proceedings of the European Workshop on Natural Language Generation*, pages 2–11.
- David A. Smith and Jason Eisner. 2006. Quasi-synchronous grammars: alignment by soft projection of syntactic dependencies. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 23–30.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642.
- Radu Soricut and Daniel Marcu. 2005. Towards developing generation algorithms for text-to-text applications. In *Proceedings of ACL*, pages 66–74.
- Radu Soricut and Daniel Marcu. 2006. Stochastic language generation using WIDL-expressions and its application in machine translation and summarization. In *Proceedings of ACL-COLING*, pages 1105–1112.
- Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of ICSLP*, pages 901–904.
- Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *Transactions of the Association for Computational Linguistics*, 2:219–230.

- Kapil Thadani and Kathleen McKeown. 2008. A framework for identifying textual redundancy. In *Proceedings of COLING*, pages 873–880.
- Kapil Thadani and Kathleen McKeown. 2011a. Optimal and syntactically-informed decoding for monolingual phrase-based alignment. In *Proceedings of ACL-HLT*, pages 254–259.
- Kapil Thadani and Kathleen McKeown. 2011b. Towards strict sentence intersection: decoding and evaluation strategies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 43–53.
- Kapil Thadani and Kathleen McKeown. 2013a. Sentence compression with joint structural inference. In *Proceedings of CoNLL*, pages 65–74.
- Kapil Thadani and Kathleen McKeown. 2013b. Supervised sentence fusion with single-stage inference. In *Proceedings of IJCNLP*, pages 1410–1418.
- Kapil Thadani, Scott Martin, and Michael White. 2012. A joint phrasal and dependency model for paraphrase alignment. In *Proceedings of COLING 2012*, pages 1229–1238.
- Kapil Thadani. 2014. Approximation strategies for multi-structure sentence compression. In *Proceedings of ACL*, pages 1241–1251.
- Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of ICML*, pages 104–.
- Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of ACL*, pages 290–297.
- Emmanouil Tzouridis, Jamal Nasir, and Ulf Brefeld. 2014. Learning to summarise related sentences. In *Proceedings of COLING*, pages 1636–1647.
- Yuya Unno, Takashi Ninomiya, Yusuke Miyao, and Jun’ichi Tsujii. 2006. Trimming CFG parse trees for sentence compression using machine learning approaches. In *Proceedings of ACL-COLING*, pages 850–857.
- Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of COLING*, pages 836–841.
- Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of ACL*, pages 1384–1394.
- Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Proceedings of ACL*, pages 565–574.
- Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of EMNLP*, pages 409–420.

- Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proceedings of EMNLP*, pages 233–243.
- Kristian Woodsend, Yansong Feng, and Mirella Lapata. 2010. Title generation with quasi-synchronous grammar. In *Proceedings of EMNLP*, pages 513–523.
- Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of ACL*, pages 1015–1024.
- Chang Xu, Dacheng Tao, and Chao Xu. 2013. A survey on multi-view learning. *CoRR*, abs/1304.5634.
- Yezhou Yang, Ching Lik Teo, Hal Daumé, III, and Yiannis Aloimonos. 2011. Corpus-guided sentence generation of natural images. In *Proceedings of EMNLP*, pages 444–454.
- Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013a. A lightweight and high performance monolingual word aligner. In *Proceedings of ACL*, pages 702–707.
- Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013b. Semi-markov phrase-based monolingual alignment. In *Proceedings of EMNLP*, pages 590–600.
- Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: unsupervised extraction of lexical simplifications from wikipedia. In *Proceedings of HLT-NAACL*, pages 365–368.
- David Zajic, Bonnie Dorr, and Richard Schwartz. 2002. Automatic headline generation for newspaper stories. In *Proceedings of the ACL Workshop on Automatic Summarization / Document Understanding Conference (DUC)*, pages 78–85.
- David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing and Management*, 43(6):1549–1570.
- Kai Zhao and Liang Huang. 2013. Minibatch and parallelization for online large margin structured learning. In *Proceedings of HLT-NAACL*, pages 370–379.
- Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *Proceedings of ACL-IJCNLP*, pages 834–842.
- Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of COLING*, pages 1353–1361.

Appendices

Appendix A








Glossary of Symbols

Symbol	Description	First occurrence
S	An input sentence	§3.2.1
\mathcal{S}	Set of input sentences	§7.2.1
C	Compression of an input sentence	§3.2.1
\hat{C}	Highest-scoring compression	§3.2.1
C_{ref}	Reference compression	§3.5
F	Fusion of multiple input sentences	§7.2.1
\hat{F}	Highest-scoring fusion	§7.2.1
A	Alignment between input sentences	§8.2.1
\hat{A}	Highest-scoring alignment	§8.2.1
n	Number of tokens in an input sentence	§3.2.1
m	Number of tokens in an output sentence	§4.1.2
ℓ	Number of semantic concepts in a sentence	§6.1.1
t	Token in a sentence	§3.2.1
\mathcal{T}	Set of tokens	§3.2.1
\mathcal{Y}	Set of n-grams	§3.2.2
\mathcal{Z}	Set of dependency edges	§3.2.2
b	Dependency edge label	§3.3.1
\mathcal{B}	Set of dependency edge labels	§3.3.1

Symbol	Description	First occurrence
f	Semantic frame in a sentence	§6.1.4
\mathcal{F}	Set of semantic concepts	§6.1.1
\mathcal{U}	Set of semantic relations	§6.1.1
\mathcal{G}	Set of similar tokens across input sentences	§7.2.2.1
\mathcal{H}	Set of similar tokens within a sentence	§7.2.2.1
\mathcal{V}	Set of phrases in a sentence	§8.2.1
Δ	Generic scoring function	§3.2.1
Δ_{tok}	Score of token in compression/fusion	§3.2.1
Δ_{ngr}	Score of n-gram in compression/fusion	§3.2.2
Δ_{dep}	Score of dependency in compression/fusion	§3.2.2
Δ_{sem}	Score of semantic relation in compression	§6.1.1
Δ_{phr}	Score of phrase pair in alignment	§8.2.2
Δ_{edg}	Score of dependency pair in alignment	§8.2.2
Δ_{tok}	Vector of scores for all tokens	§3.2.2
Δ_{ngr}	Vector of scores for all n-grams	§3.2.2
Δ_{dep}	Vector of scores for all dependencies	§3.2.2
Δ_{sem}	Vector of scores for all semantic relations	§6.1.1
Δ_{phr}	Vector of scores for all phrase alignments	§8.2.2
Δ_{edg}	Vector of scores for all dependency edge matchings	§8.2.2
ϕ	Generic vector of features	§3.2.1
ϕ_{tok}	Vector of token and frame features	§3.2.1
ϕ_{ngr}	Vector of n-gram features	§3.4
ϕ_{dep}	Vector of dependency features	§3.4
ϕ_{sem}	Vector of semantic relation features	§6.2.2
ϕ_{phr}	Vector of phrase alignment features	§8.3
ϕ_{edg}	Vector of dependency edge matching features	§8.3
θ	Generic vector of parameter weights	§3.2.1
θ_{tok}	Vector of parameters for ϕ_{tok}	§3.2.1

Symbol	Description	First occurrence
x	Indicator for token in compression/fusion	§3.2.1
y	Indicator for n-gram in compression/fusion	§3.2.2
z	Indicator for dependency in compression/fusion	§3.2.2
x^+	Indicator for token or concept in compression	§6.1.1
u	Indicator for semantic relation in compression	§6.1.1
v	Indicator for aligned phrases in alignment	§8.2.2
w	Indicator for matched edges in alignment	§8.2.2
ξ	Indicator for participating token pairs in alignment	§8.2.3
\mathbf{x}	Incidence vector for token solution	§3.2.1
\mathbf{y}	Incidence vector for n-gram solution	§3.2.2
\mathbf{z}	Incidence vector for dependency solution	§3.2.2
$\hat{\mathbf{y}}$	Highest-scoring n-gram solution	§4.1.1
$\hat{\mathbf{z}}$	Highest-scoring dependency solution	§4.1.1
$\tilde{\mathbf{x}}$	Relaxed token solution	§4.1.3
$\tilde{\mathbf{z}}$	Relaxed dependency solution	§4.1.3
\mathbf{x}^+	Incidence vector for token and concept solution	§6.1.1
\mathbf{u}	Incidence vector for semantic relation solution	§6.1.1
\mathbf{v}	Incidence vector for phrase alignment solution	§8.2.2
\mathbf{w}	Incidence vector for edge matching solution	§8.2.2
γ	Flow variable for dependency commodity	§3.3.1
γ'	Flow variable for adjacency commodity	§3.3.2
γ''	Flow variable for relational commodity	§6.1.2
Γ_{\max}	Maximum amount of dependency flow	§3.3.1
Γ'_{\max}	Maximum amount of adjacency flow	§3.3.2
Γ''_{\max}	Maximum amount of relational flow	§6.1.2
Υ_{\max}	Maximum number of incoming semantic relations	§6.1.2
$\boxed{\text{BN}}$	Compression corpus of broadcast news transcripts	§3.1
$\boxed{\text{WN}}$	Compression corpus of written news	§3.1

Symbol	Description	First occurrence
$\mathbf{0}$	Vector of all zeros	§3.5.1
$\mathbf{1}$	Vector of all ones	§3.3.3
λ	Lagrange multiplier	§4.1.2
λ_{\min}	Lower bound on Lagrange multiplier	§5.1.4.2
λ_{\max}	Upper bound on Lagrange multiplier	§5.1.4.2
$\boldsymbol{\lambda}$	Vector of Lagrange multipliers	§4.1.1
L	Lagrangian	§4.1.1
f	Subproblem for n-gram compression	§4.1.1
g	Subproblem for dependency compression	§4.1.1
h	Subproblem for variable-length compression	§5.1.4.2
\mathcal{C}	Set of nodes in a cycle	§3.3.1
μ	Order of n-grams	§3.3.2
ω	Compression rate	§3.3.3
ω_{\min}	Lower bound on compression rate	§3.3.3
ω_{\max}	Upper bound on compression rate	§3.3.3
\mathcal{D}	Training dataset	§3.5
\mathcal{L}	Loss function for training	§3.5
i_{\max}	Maximum number of iterations	§3.5.1
η	Learning rate setting	§3.5.1
$\boldsymbol{\eta}$	Learning rate schedule	§3.5.1
\mathcal{M}_p	Aligned tokens in a context window of size p	§3.5.2
ψ	Hyperparameter for token solution	§4.1.1
$\boldsymbol{\alpha}$	Token projection of n-gram solution	§4.1.1
$\boldsymbol{\beta}$	Token projection of dependency solution	§4.1.1
\emptyset	Empty set	§4.1.1
\mathcal{J}	Set of feasible solutions	§4.1.1
\mathcal{J}_{rep}	Set of repeated feasible solutions	§4.1.1
J_{\max}	Maximum number of repeated solutions	§4.1.1

Symbol	Description	First occurrence
Q	Dynamic programming chart	§4.1.2
G	Graph representing fractional tree solution	§4.1.3
σ	Type of item on the chart	§5.1.1
	Attaches rightward or completes right attachment	§5.1.1
	Attaches leftward or completes left attachment	§5.1.1
	Incomplete right attachment	§5.1.1
	Incomplete left attachment	§5.1.1
	Skips words to the right before becoming 	§5.1.2
	Incomplete sibling dependencies	§5.1.3
ϵ	Null or unassigned	§5.1.3
ε	Tolerance for bisection	§5.1.4.2
Ω	Support score for tokens	§7.2.2.1
δ	Maximum size of phrases in an input sentence	§8.2.1
π	Dependency governor for token	§8.2.3