Data-Driven Solutions to Bottlenecks in Natural Language Generation

Or Biran

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

COLUMBIA UNIVERSITY

2017

ABSTRACT

Data-Driven Solutions to Bottlenecks in Natural Language Generation

Or Biran

Concept-to-text generation suffers from what can be called *generation bottlenecks* - aspects of the generated text which should change for different subject domains, and which are usually hard to obtain or require manual work. Some examples are domain-specific content, a type system, a dictionary, discourse style and lexical style. These bottlenecks have stifled attempts to create generation systems that are generic, or at least apply to a wide range of domains in non-trivial applications.

This thesis is comprised of two parts. In the first, we propose data-driven solutions that automate obtaining the information and models required to solve some of these bottlenecks. Specifically, we present an approach to mining domain-specific paraphrasal templates from a simple text corpus; an approach to extracting a domain-specific taxonomic thesaurus from Wikipedia; and a novel document planning model which determines both ordering and discourse relations, and which can be extracted from a domain corpus. We evaluate each solution individually and independently from its ultimate use in generation, and show significant improvements in each.

In the second part of the thesis, we describe a framework for creating generation systems that rely on these solutions, as well as on hybrid concept-to-text and text-to-text generation, and which can be automatically adapted to any domain using only a domain-specific corpus. We illustrate the breadth of applications that this framework applies to with three examples: *biography generation* and *company description generation*, which we use to evaluate the framework itself and the contribution of our solutions; and *justification of machine learning predictions*, a novel application which we evaluate in a task-based study to show its importance to users.

Table of Contents

List of Figures

List of Tables

Acknowledgments

When I started the journey that ends with this dissertation, I could not possibly have appreciated the time and effort that would go into it. At this point, I am compelled to express the profound effect of the guidance, support and inspiration provided by others.

First, I would like to express my deep gratitude to my advisor, Kathy McKeown. Kathy has been a patient, supportive, and above all insightful mentor. This thesis would not have been possible without her guidance and advice, and the weight of her contribution to my experience at Columbia and my growth as a researcher cannot be overstated. I also thank Owen Rambow, who was an inspiring MS advisor and who continued to provide particularly helpful comments and ideas throughout my PhD program. Finally, I would like to thank the rest of my committee - Mike Collins, Julia Hirschberg and Drago Radev - for their very thoughtful feedback on this thesis as well as their overall positive impact on me throughout graduate school.

Many thanks to my collaborators - students, colleagues and mentors - beginning with Terra Blevins, who was instrumental to a chapter of this thesis, and with Alfio Gliozzo who impacted it through many fascinating conversations. To Jacob Andreas, Sam Brody, Noemie Elhadad, Sara Rosenthal, Sid Patwardhan, Gavin Saldanha and Marlies Santos Deas for rewarding research collaborations. My thanks go also to other past and present faculty members of the NLP group and CCLS - Mona Diab, Nizar Habash, Smara Muresan and Becky Passonneau - for their support and interest over the years.

As graduate students do, I spent a good part of my time chatting with colleagues about our work and other topics, and in some cases forming lasting friendships. I want to thank Kapil Thadani and Ioannis Paparrizos in particular for many hours spent brainstorming, debating and obsessing together. Apoorv Agarwal, Yassine Benajiba, Debanjan Ghosh, Chris Kedzie, Injung Kim, Hyunmin Lee, Jessica Ouyang, Yves Petinot, Vinod Prabhakaran and Karl Stratos all deserve special thanks for special interactions, both professional and personal. That said, I greatly enjoyed my interactions with many others at Columbia, including but certainly not limited to Mohamed AlTantawy, Daniel Bauer, Shay Cohen, Bob Coyne, Noura Farra, Hagen Fürstenau, Weiwei Guo, Chris Hidey, Mukund Jha, Melody Ju, Weiyun Ma, Kevin McInerny, Kristen Parton, Raphael Pelossof, Axinia Radeva, Mohammad Rasooli, Alexander Rush, Cem Sübakan, Swabha Swayamdipta, Olivia Winn and Boyi Xie, as well as all participants of the FUSE project. I should also mention those outside the Columbia CS department with whom I spent many hours discussing my research and academic life in general: Courtenay Cotton, Luc Despaux, Lior Galanti and Shani Gluska.

Finally, I am grateful to my close friends and family for their love and support. To my parents, who were always enthusiastically supportive of anything I chose to do, and encouraged me to pursue my academic aspirations in particular. They have a part in all of my accomplishments. To my brother, Noam, whose voice is my second voice and whose counsel I am fortunate to have. To my wonderful friends, who made sure I keep enjoying the real world while taking this dive. To Clara - my favorite - who every day listens, supports, advises and challenges me. I thank her for always reminding me of the point. And to Kai, for being here. I dedicate this work to you.

Chapter 1

Introduction

Natural Language Generation (NLG) is the field of research concerned with automatically producing human-readable text suited for a particular application. There are two main approaches: traditional NLG, often simply referred to as *generation* and more recently as *concept-to-text generation*, in which text is generated from abstract representations; and *text-to-text generation* in which existing text is extracted and manipulated to generate a desired text.

Concept-to-text generation (C2T) usually follows, at least to some extent, the model of a generation pipeline. The traditional generation pipeline (Rambow and Korelsky, 1992; Reiter and Dale, 1997) consists of three major independent steps: *document planning* (often split into *content selection* and *discourse planning*), where the entire "document" to be generated is planned, i.e. the (abstract, semantic space) messages that comprise it are chosen and arranged; *micro-planning*, where sentence boundaries are determined; and *realization*, where the text is made concrete with lexical and syntactic choices. Underlying a C2T system is its knowledge base, which consists of concepts and relations (in some form) and often a type system.

C2T is more flexible than text-to-text generation (T2T), since any text can be generated regardless of what is available. In practice, however, there are two major problems with C2T: first, abstract semantic structures and knowledge bases must be created for each new application domain; and second, rules for generating text documents from these abstract structures must be created, and are often also domain-dependent. As a result, C2T is difficult to use except in special circumstances: small, closed domains; applications where generated text is essentially templated; applications where the generated text is very short; etc.

Starting in the early 2000's, when Natural Language Processing as a field moved towards corpus-driven approaches, T2T was gaining popularity as a way to leverage existing data and generate complex, human-readable text for certain applications (in particular, summarization and question answering). T2T is also relatively domain-agnostic, which makes it more attractive. The main drawback of T2T is its inability to generate text for applications where the expected output is not already available. In summarization, a full text exists (either as one cohesive document or as snippets from separate documents) and needs to be summarized; in question answering (QA), the answer exists somewhere (possibly in multiple places) and needs to be found and presented as an answer (and possibly aggregated or otherwise inferred). In most other applications, however, we want to automatically generate text that simply does not exist anywhere yet, and in these cases we must resort to C2T with its difficulties.

Note that applications such as summarization and QA describe the *function* of the generated text as opposed to its *subject domain*. One of the main advantages of T2T is that it is domain agnostic: the text retrieved is already conceptually, lexically and stylistically within the domain, and usually nothing special needs to be done when using a T2T system in a new subject domain.[1]

Applications that are typically only possible to approach with a C2T method are also defined mainly by the function of the generated text. In some cases - indeed many of the cases explored so far in the literature - the function is so specific that it specifies the subject domain as well (e.g., description of software structures (Lavoie et al., 1997); weather forecast (Reiter et al., 2005); football match logs (Bouayad-Agha et al., 2011)). In other, more challenging applications, the domain is open-ended to some extent (e.g., biography

[1]It is important to distinguish between the subject domain and the *genre* of the text: while T2T approaches are not always genre-independent (e.g., they may be specialized for newswire text, encyclopedia text etc.), they are generally domain-independent (e.g., it does not matter whether the news articles discuss finance or foreign affairs).

(Duboue et al., 2003); product description (Androutsopoulos et al., 2013)).

At this point, we should be more specific about the difficulties of C2T that we mentioned earlier. C2T generation pipelines suffer from what we will refer to in this thesis as *generation bottlenecks*. A bottleneck is a part of the generation pipeline that must be created *for each subject domain*, even for the same application function. These bottlenecks exist in various levels of the pipeline: the knowledge base that must contain concepts, entities and relations relevant to the domain; the content selector, which should select messages important in the domain; the discourse planner, which should know the discourse structures common in the domain; and the realization component which must produce text using the style of the domain. With a few notable exceptions (see Section 1.1), these bottlenecks are handled in previous work either manually or with an automated approach specific to a subject domain.

For an application function where the goal is to create text that does not yet exist anywhere, full T2T approaches are not useful. Corpus-driven and even partially T2T approaches, however, can be. Consider the example application of biography generation: while it is true that it is in general not possible to find existing text describing most people, it is in fact possible to find existing text on relevant domain entities; for example, there is probably text describing the university a person graduated from (independently of that person). In other words, while we must use C2T to create the overall structure and to generate much of the central text for an application, we can use a general corpus in the subject domain to extract domain-specific text (that is, specific to the subject domain, but agnostic to the application function). Our generation framework in Chapter 5 uses such a hybrid approach to tackle the bottleneck of getting domain-specific information by complementing a base C2T pipeline with (manipulated) sentences extracted from a domain corpus.

Similarly, though less straightforwardly, a general corpus in the subject domain also contains information that can help with the other bottlenecks we mentioned earlier. The relevant entities and relations of the domain (e.g. cities, universities, political parties, and relations between them), unique lexical terms (e.g. domain jargon and rare word senses), and domain-specific patterns at both the sentence level (templates) and the document level (discourse and ordering) are all hidden within the corpus.

In this thesis, we propose three domain-agnostic corpus-driven approaches to solving key generation bottlenecks: a method for mining paraphrasal templates (which can be used in language realization) from a simple text corpus; a method for inferring taxonomic relations between concepts (which assists in building a type system and in finding synonyms of known concepts); and a method for extracting a statistical discourse model from a simple text corpus (relying on our work on discourse parsing) and using it to improve discourse planning. Each of these methods is a stand-alone contribution, and is evaluated by itself. We then introduce a family of generation applications which we call Generation Endeavors with Modular Subjects (GEMS), and which includes many interesting NLG applications. This family is unique in that while each application is a closed *application function* (e.g. biography), it can be applied to an open set of *subject domains* (politicians, scientists...). We introduce a framework for building hybrid C2T/T2T generation systems for these applications that adapts to new subject domains completely automatically, and describe three example applications: *biography generation*, *company description generation*, and *Machine Learning prediction justification*. We use the first two applications to evaluate our framework, as well as the three methods we propose in the context of a larger generation framework. The third is a novel contribution in its own right, and we evaluate its usefulness.

Table 1.1 lists the major contributions of this thesis, along with the corresponding chapters and publications.

Contribution		Chapters	Publications
Data-driven generation bottleneck solutions	Paraphrasal template mining for realization	Chapter 2	(Biran et al., 2016)
	Taxonomy induction	Chapter 3	(Biran and McKeown, 2013b)
	Corpus-driven discourse planning + end-to-end discourse parsing	Chapter 4	(Biran and McKeown, 2013a; Biran and McKeown, 2015b; Biran and McKeown, 2015a)
Hybrid, domain-agnostic generation framework for GEMS		Chapters 5 - 6	TBD
ML prediction justification		Chapter 7	(Biran and McKeown, 2014)

Table 1.1: List of contributions, with chapters and publications

The thesis is organized as follows. In the first part, we describe our three data-driven bottleneck solutions. Chapter 2 describes paraphrasal template mining; Chapter 3 introduces taxonomy extraction; and Chapter 4 discusses discourse planning and our work on discourse parsing supporting it.

In the second part, we show an example of how our work can be used for generation. Chapter 5 defines the GEMS family of generation applications and describes the generation framework. Chapter 6 describes two straightforward applications, *biography* and *company descriptions*, which rely on RDF data, and evaluates the generation framework using these applications. Chapter 7 introduces the *Machine Learning prediction justification* application. Finally, in Chapter 8 we conclude and discuss both the limitations of our work and potential future work.

1.1 Background

Natural Language Generation as a task dates back at least as far as the 1960's, when early approaches appeared, focusing on generating individual sentences. These were mostly created for purely academic purposes (Yngve, 1961; Klein et al., 1963; Weizenbaum, 1966) but also for some real-world applications like question answering (see survey in Simmons (1970)) and robotics (Coles, 1969). In the decade that followed, the focus slowly shifted towards generation of longer texts and applications such as description generation (Carbonell and Collina, 1973), story writing (Klein, 1973; Meehan, 1977) and expert system explanation (Shortliffe and Buchanan, 1975; Swartout, 1981), as well as early proposals for generic generation methods (Mann and Moore, 1981).

The next decade saw the appearance of more robust, generic approaches to generation, and a focus on document planning. McKeown (1985) introduced *schemata*, recursive discourse plans that could be used to generate many kinds of texts, and was the first to incorporate a discourse planning stage in generation. Schemata became a standard approach to generation and were used by many systems in the years that followed (Hovy, 1987; Paris, 1988; McCoy, 1989; McKeown et al., 1997). Rhetorical Structure Theory (RST) was introduced by Mann and Thompson (1987), and was subsequently used as a discourse planning framework by many systems as well (Hovy, 1991; Hovy, 1993; Moore and Paris, 1993; Mellish et al., 1998; Power, 2000; Bouayad-Agha et al., 2000). By the 1990's, the standard three-layered generation pipeline (consisting of *document planning*, *micro-planning* and *realization*) was widely acknowledged and eventually formalized and elaborated on by Reiter and Dale (1997), who proposed an abstract framework intended to generalize over virtually all existing concept-to-text generation systems. In general, one of the clear trends before the 2000's was increasing generality, where systems and frameworks were proposed that could handle increasingly many types of applications and a broader range of domains (note the similarity in goal to this thesis). At the same time, standardized sentence-level generation frameworks and libraries appeared and became widely used: notable examples include KPML (Bateman, 1997), RealPro (Lavoie and Rambow, 1997) and the SURGE realizer (Elhadad and Robin, 1996) based on Functional Unification Grammar (Shieber, 1986).

The movement towards generality has largely stopped in the 2000's (with some exceptions - a generic architecture was proposed by Mellish et al. (2006), for example, but the field was not eager to follow up). At the time, another trend was rapidly taking the lead - corpus-driven, statistical methods, which rose due to the increasing availability of large text corpora and were seen as a way around some generation bottlenecks. Concept-to-text generation systems first started significantly incorporating statistical information derived from text data by using it to score various possibilities of generated structures (such as document plans, parse trees and lexical choices) and make probabilistic selections based on similarity to the corpus, thereby solving the bottleneck requirement of having a generation plan for any possible situation (Knight and Hatzivassiloglou, 1995; Langkilde and Knight, 1998). These approaches still relied on structured knowledge bases and static document plans, but used corpus statistics to "close the gaps" where they were missing or incomplete. In the years that followed, NLG took a more statistical, less structured direction. Fully statistical (learned from data) approaches were proposed in the concept-to-text literature for content selection (Duboue and McKeown, 2003; Barzilay and Lapata, 2005a; Kelly et al., 2009), sentence ordering (Duboue and McKeown, 2001; Duboue and McKeown, 2002; Dimitromanolaki and Androutsopoulos, 2003; Barzilay and Lee, 2004) and realization (Bangalore and Rambow, 2000; Langkilde-Geary, 2002; Guo et al., 2008; Filippova and Strube, 2009; White and Rajkumar, 2009).

Another facet of the corpus-driven trend was the rise of text-to-text generation, especially for *summarization*. T2T was actually explored very early on, both for summarization/abstraction of database texts (Luhn, 1958; Edmundson and Oswald, 1959) and for sentence transformation and shortening (Klein, 1965). However, these early attempts were rare and unique, and despite some interest in specific applications by the Information Retrieval community (Young and Hayes, 1985; Rau, 1988), it was not until the 1990's that T2T became a central part of the generation literature. Jones (1993) presented a first solid definition of summarization (not yet acknowledged as a single application of the wider text-to-text generation task) and proposed a framework for pursuing it. Early work following that focused on single document summarization (Baldwin and Morton, 1998; Barzilay and Elhadad, 1999; Berger and Mittal, 2000; Hatzivassiloglou et al., 2001), followed by multi-document summarization as the web became prevalent (Radev and McKeown, 1998; Amitay and Paris, 2000; Barzilay et al., 2002; Erkan and Radev, 2004; Conroy et al., 2006; Daumé and Marcu, 2006; Biadsy et al., 2008). Although summarization has been the primary application, T2T methods were quickly leveraged for other tasks, most notably question answering (Blair-Goldensohn et al., 2003; Chali and Joty, 2008) and essay grading and feedback (Burstein and Marcu, 2000). Statistical methods were proposed for subtasks within text-to-text as well, in particular sentence ordering (Lapata, 2003; Bollegala et al., 2005; Ji and Pulman, 2006; Donghong and Yu, 2008), sentence compression (McDonald, 2006; Clarke and Lapata, 2006; Filippova and Strube, 2008a; Clarke and Lapata, 2008; Napoles et al., 2011) and sentence fusion (Filippova and Strube, 2008b; Thadani and McKeown, 2011; Thadani and McKeown, 2013).

Recent work on C2T generation, meanwhile, includes the development of several interesting trajectories. One has been an increasing focus on generating text from raw data. Although earlier examples exist (Kukich, 1983; Bourbeau et al., 1990; Sripada et al., 2004), *data-to-text generation* became newly popularized by Reiter (2007) as an extension of the traditional pipeline (knowledge base $\rightarrow$ messages $\rightarrow$ document plan $\rightarrow$ sentence plan $\rightarrow$ text) with components that first transform raw numeric data into digestible conceptual information. This has expanded the range of NLG applications to include, for example, sensor data description (Gatt et al., 2009; Van Der Meulen et al., 2010). In Chapter 7, we describe the data-to-text application of justifying machine learning predictions. A somewhat related body of research includes recent work on generating descriptions of images and videos (Kazemzadeh et al., 2014; Gkatzia et al., 2015).

Most relevant to our work, there has been some rekindled interest in generic NLG approaches, although different in flavor from the highly abstract specification frameworks of Reiter and Dale (1997) and Mellish et al. (2006). Specifically, researchers have begun looking at corpus-driven ways to automatically adapt generation systems for different domains. Of particular note in that regard are the approaches of Angeli et al. (2010) and Kondadadi et al. (2013) who leverage aligned text-data corpora to mine domain-specific templates for realization and learn how to order sentences in the domain. These approaches tackle problems similar to the ones we do in this thesis, but they rely on a corpus of sample texts (of the kind that they are expected to generate) and knowledge bases corresponding to these texts. In contrast, we focus on solving generation problems for which sample texts do not exist at all. Other recent work that focuses on automatic system adaptation has looked at adapting generation output to different user groups (Janarthanam and Lemon, 2010; Gkatzia et al., 2014); adapting summarization systems to different genres (Lloret and Boldrini, 2015); adapting dialog generation systems to different applications (Rieser and Lemon, 2011) and different domains (Walker et al., 2007); and parameterizing existing hand-crafted generation systems to increase the range of domains they can handle (Lukin et al., 2015).

A related trend, enabled by access to increasingly large semantic web repositories (Auer et al., 2007; Suchanek et al., 2007; Bollacker et al., 2008) is concept-to-text generation from RDF, OWL and similar semantic web data formats. Because so much diverse data is available in these formats, there has been some work that puts particular emphasis on presumably generic aspects of generation from semantic web data, such as sentence realization and content selection (Power and Third, 2010; Bouayad-Agha et al., 2012; Dannélls et al., 2012; Androutsopoulos et al., 2013). While this literature is not concerned with domain adaptation and solving bottlenecks specifically, it complements our work well because semantic web repositories are available for virtually any domain. In Chapter 6, we describe two applications that generate texts from RDF data; while in our work we focused on areas relevant to the bottlenecks we aim to solve, it can certainly be complemented by RDF-specific approaches to, for example, realization.

Finally, another line of research that has come to prominence in recent years is concerned with joint end-to-end generation methods, particularly using neural nets. This scheme replaces the traditional pipeline architecture with a single statistical model that learns to map structured data to generated text. This has typically been done with neural architectures such as LSTM (Potash et al., 2015; Wen et al., 2015; Wang et al., 2016) or other attention-based neural models (Rush et al., 2015), but there have also been proposals for new task-specific models (Yin et al., 2016), and for joint generation methods that do not rely on neural models (Lampouras and Androutsopoulos, 2013; Dušek and Jurcicek, 2015). In the dialog generation literature, active learning (Mairesse et al., 2010) has also been used. While this line of work represents an alternative, unrelated approach to the one we operate in (i.e., the pipeline approach), its prominence highlights current interest in our common goal of fully automated creation of natural language generation systems for any domain.

1.2 Our Work in the Context of the Field

In the previous section, we discussed specific trends of NLG in their historical context. In this section, we take a bird's eye view of these trends, focusing on the overall trajectories rather than on specific trends and short term focuses. Using this more abstract view, we describe the evolution of NLG as a field and discuss its current direction, as well as our expectation for its future directions, and place our work (in particular, the framework described in Chapter 5 and the general approach of data-driven domain adaptation for dealing with NLG bottlenecks) in the context of these directions.

In the 60 or so years since its origination, NLG has largely followed three general trajectories: increased generated text **size** (and associated complexity); increased reliance on statistical **modeling** and data-driven approaches as opposed to expert knowledge and hand crafted rules; and systems that are increasingly generic and broader in **scope** (that is, they are capable of handling more genres and subject domains without reliance on manual work or rare resources).

In each of the three aspects, new phases introduced new challenges. In the text size aspect, for example, multi-sentence documents introduced the challenge of document planning, which did not exist when NLG was concerned with single sentences. NLP researchers have only recently started looking at narrative modeling (Ouyang and McKeown, 2015), the upcoming challenge for generating even longer documents such as books and large-scale essays. Similarly, increased reliance on statistics poses challenges in modeling, which started with relatively simple likelihood models (Knight and Hatzivassiloglou, 1995), through the last two decades of mostly supervised, unstructured machine learning model use, and most recently, end-to-end structured models (Lampouras and Androutsopoulos, 2013; Rush et al., 2015). Finally, increasingly generic systems require increasing layers of abstraction (Reiter and Dale, 1997; Mellish et al., 2006) and, more recently, reliance on data-driven domain adaptation (Angeli et al., 2010). Table 1.2 shows a timeline of the phases we have identified in each of the trajectories, along with the challenges of each phase. The work in this thesis belongs to the second phase in text size, the third phase in modeling, and the third stage in scope.

It is interesting to explore the interaction among the three trajectories. Clearly, phases in certain trends rely on progress in another: Phase 3 of the scope aspect (concrete, data-driven generic frameworks) was not possible before Phase 3 of the modeling aspect (ML task solutions) was already established. In addition to full dependence of this sort, there are also cases of soft dependence, where new phases in certain trajectories would likely not have been explored if it were not for the needs of new phases in another trajectory. For example, it is unlikely that we would have seen scope move beyond Phase 1, and modeling move beyond Phase 3, if we were still at Phase 1 of text size (i.e., if NLG was only concerned with generating single sentences).

Often, when a new phase in a particular trajectory was first explored, authors have reverted to earlier phases in one or both others. For example, the first statistical models were applied to syntactic tree generation in single sentences in the 90's (Knight and Hatzivassiloglou, 1995; Langkilde and Knight, 1998), even though NLG had been at the phase

Year	Text size	Modeling	Scope
1960 - 1965	Phase 1: Generation of	Phase 1: Hand-	Phase 1: Manual
1965 - 1970	single sentences, including	crafted rules	domain adaptation,
1970 - 1975	short multi-sentence texts		non-generic systems
1975 - 1980	without ordering/selection		
1980 - 1985	(syntax, lexical choice)		
1985 - 1990	Phase 2: Multi-		
1990 - 1995	sentence documents		
1995 - 2000	(content selection,	Phase 2: statistical	Phase 2: Descriptive
	discourse planning,	likelihood models	generic frameworks
2000 - 2005	micro-planning)	Phase 3: ML task	(task abstraction)
2005 - 2010		solutions (mainly	
2010 - 2015		unstructured)	Phase 3: Concrete
Present		Entering Phase 4:	generic frameworks
		Towards end-to-end	(domain adaptation)
Future	Phase 3: Books, large-	structured models	Phase 4: Fully
	scale essays (narrative)		generic systems

Table 1.2: Timeline of phases in three key aspects of NLG

of generating long documents for over a decade. The first statistical models for sentence ordering and discourse came several years later (Duboue and McKeown, 2001; Duboue and McKeown, 2002). Similarly, the first generic frameworks focused on syntax of single sentences, based on hand-crafted grammars (Elhadad and Robin, 1996; Bateman, 1997; Lavoie and Rambow, 1997). *Descriptions* of generic frameworks for document-level generation systems rose to prominence around the same time (Reiter and Dale, 1997), but generic *concrete* frameworks (which, unlike simply descriptive frameworks, are capable of producing systems adapted to a particular domain or genre with minimal or no manual work) which rely on aligned datasets have only recently started to appear (Angeli et al., 2010; Kondadadi et al., 2013). Our work introduces the first, to our knowledge, concrete domain-adaptable framework which relies only on simple text data.

The introduction of a concrete framework which relies only on simple data sets is a crucial evolution within the current phase of the scope trajectory. Until fully generic systems become possible, the best approach for reducing manual work in the creation of new generation systems remains data-driven domain-adaptable frameworks. In order for such frameworks to be relevant for a wide range of applications, they cannot rely on aligned data (as they have so far) or on otherwise special, rare data sets. Future concrete domain-adaptable frameworks, if they are to be widely useful, will have to find ways (as we have with our bottleneck solutions and hybrid generation approach) to extract the information they need from easily obtainable simple text resources.

Part I

Data-Driven Methods for Generation Bottlenecks

Chapter 2

Paraphrasal Template Extraction

One of the main difficulties in Natural Language Generation is the *surface realization* of messages: transforming a message from its internal representation to a natural language phrase, sentence or larger structure expressing it. Often the simplest way to realize messages is though the use of templates. For example, any message about the birth year and place of any person can be expressed with the template “[Person] was born in [Place] in [Year]”.

Templates have the advantage that the generation system does not have to deal with the internal syntax and coherence of each template, and can instead focus on document-level discourse coherence and on local coreference issues. On the other hand, templates have two major disadvantages. First, having a human manually compose a template for each possible message is costly, especially when a generation system is relatively open-ended or is expected to deal with many domains. In addition, a text generated using templates often lacks variation, which means the system’s output will be repetitive, unlike natural text produced by a human.

In this chapter we tackle a task aimed at solving both problems: automatically mining paraphrasal templates, i.e. groups of templates which share the same slot types and which, if their slots are filled with the same entities, result in paraphrases. We introduce an unsupervised approach to paraphrasal template mining from the text of Wikipedia articles. The work described in this chapter is also described in (Biran et al., 2016).

Most previous work on paraphrase detection focuses either on a corpus of aligned paraphrase candidates or on such candidates extracted from a parallel or comparable corpus. In contrast, we are concerned with a very large dataset of templates extracted from a single corpus, where any two templates are potential paraphrases. Specifically, paraphrasal templates can be extracted from sentences which are not in fact paraphrases; for example, the sentences "The population of Missouri includes more than 1 million African Americans" and "Roughly 185,000 Japanese Americans reside in Hawaii" can produce the templated paraphrases "The population of [american state] includes more than [number] [ethnic group]" and "Roughly [number] [ethnic group] reside in [american state]". Looking for paraphrases among templates, instead of among sentences, allows us to avoid using an aligned corpus.

Our approach consists of three stages. First, we process the entire corpus and determine slot locations, transforming the sentences to templates (Section 2.2.1). Next, we find the most appropriate *type* for each slot using a large taxonomy, and group together templates which share the same set of types as potential paraphrases (Section 2.2.2). Finally, we cluster the templates in each group into sets of paraphrasal templates (Section 2.2.3).

2.1 Related Work

To our knowledge, although several approaches exist which utilize paraphrasal templates in some way, the task of extracting them has not been defined as such in the literature. The reason seems to be a difference in priorities. In the context of NLG, Angeli et al. (2010) as well as Kondadadi et al. (2013) used paraphrasal templates extracted from aligned corpora of text and data representations in specific domains, which were grouped by the data types they relate to. Duma and Klein (2013) extract templates from Wikipedia pages aligned with RDF information from DBPedia, and although they do not explicitly mention aligning multiple templates to the same set of RDF templates, the possibility seems to exist in their framework. In contrast, we are interested in extracting paraphrasal templates from non-aligned text for general NLG, as aligned corpora are difficult to obtain for most domains.

While template extraction has been a relatively small part of NLG research, it is very prominent in the field of Information Extraction (IE), beginning with Hearst (1992). There, however, the goal is to extract good data and not to extract templates that are good for generation. Many pattern extraction (as it is more commonly referred to in IE) approaches focus on semantic patterns that are not coherent lexically or syntactically, and the idea of paraphrasal templates is not important (Chambers and Jurafsky, 2011). One exception which explicitly contains a paraphrase detection component is (Sekine, 2006).

Meanwhile, independently of templates, detecting paraphrases is an important, difficult and well-researched problem of Natural Language Processing. It has implications for the general study of semantics as well as many specific applications such as Question Answering and Summarization. Research that focuses on mining paraphrases from large text corpora is especially relevant for our work. Typically, these approaches utilize a parallel (Barzilay and McKeown, 2001; Ibrahim et al., 2003; Pang et al., 2003; Quirk et al., 2004; Fujita et al., 2012; Regneri and Wang, 2012) or comparable corpus (Shinyama et al., 2002; Barzilay and Lee, 2003; Sekine, 2005; Shen et al., 2006; Zhao et al., 2009; Wang and Callison-Burch, 2011), and there have been approaches that leverage bilingual aligned corpora as well (Bannard and Callison-Burch, 2005; Madnani et al., 2008).

Of the above, two are particularly relevant. Barzilay and Lee (2003) produce *slotted lattices* that are in some ways similar to templates, and their work can be seen as the most closely related to ours. However, as they rely on a comparable corpus and produce untyped slots, it is not directly comparable. In our approach, it is precisely the fact that we use a rich type system that allows us to extract paraphrasal templates from sentences that are not, by themselves, paraphrases and avoid using a comparable corpus. Sekine (2005) produces typed phrase templates, but the approach does not allow learning non-trivial paraphrases (that is, paraphrases that do not share the exact same keywords) from sentences that do not share the same entities (thus remaining dependent on a comparable corpus), and the type system is not very rich. In addition, that approach is limited to learning short paraphrases of relations between two entities.

Another line of research is based on contextual similarity (Lin and Pantel, 2001; Paşca and Dienes, 2005; Bhagat and Ravichandran, 2008). Here, shorter (phrase-level) paraphrases are extracted from a single corpus when they appear in a similar lexical (and in later approaches, also syntactic) context. The main drawbacks of these methods are their inability to handle longer paraphrases and their tendency to find phrase pairs that are semantically related but not real paraphrases (e.g. antonyms or taxonomic siblings).

More recent work on paraphrase detection has, for the most part, focused on classifying provided sentence pairs as paraphrases or not, using the Microsoft Paraphrase Corpus (Dolan et al., 2004). Mihalcea et al. (2006) evaluated a wide range of lexical and semantic measures of similarity and introduced a combined metric that outperformed all previous measures. Madnani et al. (2012) showed that metrics from Machine Translation can be used to find paraphrases with high accuracy. Another line of research uses the similarity of texts in a latent space created through matrix factorization (Guo and Diab, 2012; Ji and Eisenstein, 2013). Other approaches that have been explored are explicit alignment models (Das and Smith, 2009), distributional memory tensors (Baroni and Lenci, 2010) and syntax-aware representations of multi-word phrases using word embeddings (Socher et al., 2011). Word embeddings were also used by Milajevs et al. (2014). These approaches are not comparable to ours because they focus on classification, as opposed to mining, of paraphrases.

Detecting paraphrases is closely related to research on the mathematical representation of sentences and other short texts, which draws on a vast literature on semantics, including but not limited to lexical, distributional and knowledge-based semantics. Of particular interest to us is the work of Blacoe and Lapata (2012), which show that simple combination methods (e.g., vector multiplication) in classic vector space representations outperform more sophisticated alternatives which take into account syntax and which use deep representations (e.g. word embeddings, or the distributional memory approach). This finding is appealing since classic vector space representation (distributional vectors) are easy to obtain and are interpretable, making it possible to drill into errors.

2.2 Method

Our method relies on a type system which links entities to one another in a taxonomy. We use a combination of WordNet (Fellbaum, 1998) and DBPedia (Auer et al., 2007), which provides both a rich top-level type system with lexicalizations of multiple senses and a large database of entities linked through the type system (the top-level DBPedia categories all have cognates in WordNet, which make the two easy to combine). Leveraging the fact that

DBPedia entities have corresponding Wikipedia pages, we also use the *redirect* terms for those pages as alternative lexicalizations of the entity (e.g., the Wikipedia article "United States" has "USA" as a redirect term, among others).

The three steps of our approach are described in detail below.

2.2.1 Creating Templates

The first step to creating the templates is to find entities, which are candidates to becoming slots in the templates. Since we are trying to find sentence-level paraphrasal templates, each sentence in the corpus is a potential template.

Entities are found in one of two ways. First, we use regular expressions to find *open set* entities: dates, percentages, currencies, counters (e.g., "9th") and general numbers. Those special cases are immediately given their known type (e.g., "date" or "percentage"). These are the only types of entities we allow without having the entity in the taxonomy.

Next, after POS-tagging the entire corpus, we look for candidate *closed set* entities in the following patterns: terms that contain only NNP (including NNPS) tags; terms that begin and end with an NNP and contain only NNP, TO, IN and DT tags; and terms that contain only capitalized words, regardless of the POS tags. Of these candidates, we only keep ones that appear in the taxonomy. Unlike the special cases above, the type of the slots created from these general entities is not yet known and will be decided in the next step.

At the end of this step, we have a set of partially-typed templates: one made from each sentence in the corpus, with its slots (but not their types in most cases) defined by the location of entities. We remove from this set all templates which have fewer than two slots as these are not likely to be interesting, and all templates which have more than five slots to avoid excessively complicated templates.

We originally experimented with simply accepting any term that appears in the taxonomy as an entity. That method, however, resulted in a large number of both errors and entities that were too general to be useful (e.g, "table", "world" and similar terms are in the taxonomy). Note that NER approaches, even relatively fine-grained ones, would not give us the same richness of types that directly comparing to the taxonomy allows. The next step, which is concerned with making decisions about the types of entities, requires that each entity we handle exist in the taxonomy.

2.2.2 Template Typing and Grouping

Determining the type of a slot in the template presents two difficulties. First, there is a sense disambiuation problem, as many lexical terms have more than one sense (that is, they can correspond to more than one entry in the taxonomy). Second, even if the sense is known, it is not clear which level of the taxonomy the type should be chosen from. For example, consider the sentence "[JFK] is [New York]'s largest airport" (the terms in square brackets will become slots once their types are determined). "JFK" is ambiguous: it can be an airport, a president, a school, etc. The first step in this process is, then, to determine which of the possible senses of the term best fits the sentence. But once we determine that the sense of "JFK" here is of an airport, there are different types we can choose. JFK is a New York Airport, which is a type of Airport, which is a type of Air Field, which is a type of Facility and so on. The specificity of the type we choose will determine the correctness of the template, and also which other templates we can consider as potential paraphrases.

Our solution is a two-stage distributional approach: *choosing the sense*, and then *choosing the type level* that best fit the context of the slot. In each stage, we construct a $pseudo-sentence$ (a collection of words in arbitrary, non-grammatical order) from words used in the taxonomy to describe each option (a sense in the first stage, and a type level in the second stage), and then use their vector representations to find the option that best matches the context.

Following the observation of Blacoe and Lapata (2012) that simple similarity metrics in traditional vector representations match and even outperform more sophisticated representations in finding relations among short texts as long as multiplication is used in forming vector representations for the texts, we use traditional context vectors as the basis of our comparisons in both stages. We collect context vectors from the entire English Wikipedia corpus, with a token window of 5. To avoid noise from rarely occurring words and reduce the size of the vectors, we remove any feature with a count below a threshold of $log_{10}(\Sigma)$ where Σ is the sum of all feature counts in the vector. Finally, the vector features are weighted with (normalized) TF*IDF.[1]

For a multi-word collection (e.g. a pseudo-sentence) ψ, we define the features of the combined vector $\mathcal{V}_\psi$ using the vectors of member words $\mathcal{V}_w$ as:

$$\mathcal{V}_{j\psi} = (\prod_{w\in\psi} \mathcal{V}_{jw})^{\frac{1}{|S|}} \tag{2.1}$$

Where $\mathcal{V}_{jw}$ is the value of the jth feature of $\mathcal{V}_w$.

To choose the sense of the slot (the first stage), we start with S, the set of all possible senses (in the taxonomy) for the entity in the slot. We create a pseudo-sentence ψ_s from the primary lexicalizations of all types in the hierarchy above each sense s - e.g., for the airport sense of JFK we create a single pseudo-sentence $\psi_{JFK-airport-sense}$ consisting of the terms "New York airport", "airport", "air field", "facility" and so on.[2] We create a vector representation $\mathcal{V}_{\psi_s}$ for each ψ_s using Equation 2.1. Then, we create a pseudo-sentence $\psi_{context}$ for the context of the slot, composed of the words in a 5-word window to the left and right of the slot in the original sentence, and create the vector $\mathcal{V}_{\psi_{context}}$. We choose the sense $\hat{s}$ with the highest cosine similarity to the context:

$$\hat{s} = \operatorname*{argmax}_{s\in S} \cos(\mathcal{V}_{\psi_s}, \mathcal{V}_{\psi_{context}})$$

Note that this is a deep similarity - the similarity of the (corpus) context of the sense and the (corpus) context of the slot context; the words in the sentence themselves are not used directly.

We use the lexicalizations of all types in the hierarchy to achieve a more robust vector representation that has higher values for features that co-occur with many levels in the sense's hierarchy. For example, we can imagine that "airplane" will co-occur with many of the types for the JFK airport sense, but "La Guardia" will not (helping to lower the score of the first, too-specific sense of "New York airport") and neither will features that co-occur with other senses of a particular type - e.g., "Apple" for the "airport" type.[3]

[1] A "term" being a single feature count, and a "document" being a vector

[2] But we exclude a fixed, small set of the most abstract types from the first few levels of the WordNet hierarchy, as these turn out to never be useful

[3] AirPort is the name of an Apple product

Once the sense is chosen, we choose the proper type level to use (the second stage). Here we create a pseudo-sentence for each type level separately, composed of all possible lexicalizations for the type. For example, the "air field" type contains the lexicalizations "air field", "landing field", and "flying field". These pseudo-sentences are then compared to the context in the same way as above, and the one with highest similarity is chosen. The reason for using all lexicalizations is similar to the one for using all types when determining the sense: to create a more robust representation that down-scores arbitrary co-occurrences.

At the end of this step, the templates are fully typed. Before continuing to the next step of finding paraphrases, we group all *potential* paraphrases together. Potential paraphrases are simply groups of templates which share exactly the same set of slot types (regardless of ordering).

2.2.3 Finding Paraphrases within Groups

Each group of potential paraphrases may contain multiple sub-groups such that each of the members of the subgroup is a paraphrase of all the others. In this last stage, we use a clustering algorithm to find these sub-groups.

We define the distance between any two templates in a group as the Euclidean distance between the vectors (created using Equation 2.1) of the two templates with the entity slots removed (that is, the pseudo-sentences created with all words in the template outside of the slots). We tried other distance metrics as well (for example, averaging the distances between the contexts surrounding each pair of corresponding slots in both templates) but the Euclidean distance seemed to work best.

Using this metric, we apply single-linkage agglomerative clustering, with the stopping criteria defined as a threshold τ for the maximum sum of squared errors (SSE) within any cluster. Specifically, the algorithm stops linking if the cluster C that would be created by the next link satisfies:

$$log(\sum_{v}^{C} d(v, \mu_C)^2) \geq \tau$$

Where μ_C is the centroid of C and d is the Euclidean distance. The logarithm is added for convenience, since the SSE can get quite large and we want to keep τ on a smaller scale.

The intuition behind this algorithm is that some paraphrases will be very similar (lexically or on a deeper level) and easy to find, while some will be more difficult to distinguish from template pairs that are related but not paraphrasal. The single-linkage approach is essentially transductive, allowing the most obvious clusters to emerge first and avoiding the creation of a central model that will become less precise over time. The threshold is a direct mechanism for controlling the trade-off between precision and recall.

At the end of this step, any pair of templates within the same cluster is considered a paraphrase. Clusters that contain only a single template are discarded (in groups that have high distances among their member templates, often the entire group is discarded since even a single link violates the threshold).

2.3 Evaluation

To evaluate our method, we applied it to the six domains described in Table 2.1. We tried to choose a set of domains that are diverse in topic, size and degree of repeated structure across documents. For each domain, we collected a corpus composed of relevant Wikipedia articles (as described in the table) and used the method described in Sections 2.2.1-2.2.3 to extract paraphrasal templates. We used Wikipedia for convenience, since it allows us to easily select domain corpora, but there is nothing in our approach that is specific to Wikipedia; it can be applied to any text corpus.

We sampled 400 pairs of paraphrases extracted from each domain and used this set of 2400 pairs to conduct a crowd-sourced human evaluation on CrowdFlower. For each template pair, we randomly selected one and used its original entities in both templates to create two sentences about the same set of entities. The annotators were presented with this pair and asked to score the extent to which they are paraphrases on a scale from 1 to 5. Table 2.2 shows the labels and a brief version of the explanations provided for each. To ensure the quality of annotations, we used a set of hidden test questions throughout the evaluation and rejected the contributions of annotators which did not get at least 70% of the test questions correctly. Of those that did perform well on the test questions, we had three annotators score each pair and used the average as the final score for the pair. In

Domain	Description	Size	Source article link
NBA	NBA teams	30	`National_Basketball_Association`
States	US states	50	N/A
AuMa	Automobile manufacturers	241	`List_of_automobile_manufacturers`
Metal	Heavy Metal bands (original movement, 1967-1981)	291	`List_of_heavy_metal_bands`
CWB	Battles of the American Civil War	446	`List_of_American_Civil_War_battles`
Marvel	Superheroes from the Marvel Comics universe	932	`Category:Marvel_Comics_superheroes`

Table 2.1: Evaluation domains. Article links preceded by https://en.wikipedia.org/wiki/

Score	Label	Explanation
5	Perfect Paraphrase	The two sentences are equivalent in meaning (but allow differences in e.g. tense, wordiness or sentiment)
4	Almost Paraphrase	The two sentences are equivalent in meaning with one minor difference (e.g., change or remove one word)
3	Somewhat Paraphrase	The two sentences are equivalent in meaning with a few minor differences, or are complex sentences with a part that is a paraphrase and a part that is not
2	Related	The sentences are related in meaning, but are not paraphrases
1	Unrelated	The meanings of the sentences are unrelated

Table 2.2: Annotation score labels and explanations

39.4% of the cases, all three annotators agreed; two annotators agreed in another 47% of the cases, and in the remaining 13.6% there was complete disagreement. The inter-annotator agreement for the two annotators that had the highest overlap (27 annotated pairs), using Cohen's Kappa, was $\kappa = 0.35$.

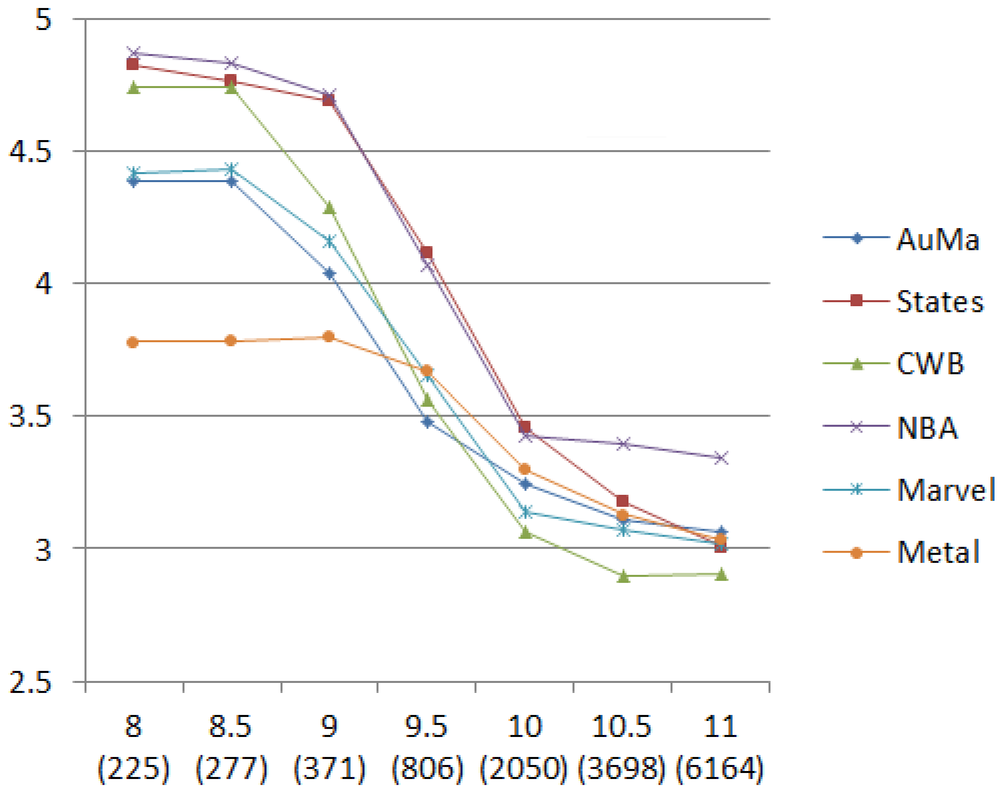


Figure 2.1: The average scores for each domain, for a range of threshold choices. The number in parentheses for each threshold is the number of paraphrases generated

The overall results are shown in Figure 2.1. Note that because of our clustering approach, we have a choice of similarity threshold. The results are shown across a range of thresholds from 8 to 11 - it is clear from the figure that the threshold provides a way to control the trade-off between the number of paraphrases generated and their precision. Table 2.3 shows the numeric details, including the number of paraphrases generated, for each domain with each threshold, highlighting those of our preferred threshold of 9.5.

The number of paraphrase clusters found changes with the threshold. For the 9.5 thresh-

Thr.	Domain	Size	Avg.	%3+	%4+	Thr.	Domain	Size	Avg.	%3+	%4+
8	NBA	5	4.9	100%	100%	8.5	NBA	11	4.8	100%	100%
	States	64	4.8	100%	99%		States	85	4.8	100%	96%
	AuMa	9	4.4	100%	89%		AuMa	9	4.4	100%	89%
	Metal	61	3.8	88%	67%		Metal	63	3.8	88%	67%
	CWB	17	4.7	96%	96%		CWB	19	4.7	96%	96%
	Marvel	133	4.4	97%	87%		Marvel	150	4.4	97%	87%
	All	290	4.3	94%	82%		All	337	4.3	95%	83%
9	NBA	14	4.7	100%	96%	9.5	NBA	**30**	**4.1**	**88%**	**70%**
	States	103	4.7	99%	93%		States	**171**	**4.1**	**86%**	**76%**
	AuMa	13	4.0	96%	71%		AuMa	**58**	**3.5**	**80%**	**50%**
	Metal	67	3.8	88%	67%		Metal	**98**	**3.7**	**83%**	**63%**
	CWB	25	4.3	90%	79%		CWB	**81**	**3.6**	**75%**	**56%**
	Marvel	209	4.2	93%	78%		Marvel	**428**	**3.7**	**83%**	**63%**
	All	431	4.2	93%	79%		All	**866**	**3.8**	**82%**	**63%**
10	NBA	125	3.4	77%	51%	10.5	NBA	255	3.4	78%	49%
	States	360	3.5	72%	54%		States	630	3.2	63%	46%
	AuMa	204	3.2	75%	42%		AuMa	371	3.1	68%	39%
	Metal	223	3.3	74%	47%		Metal	405	3.2	70%	43%
	CWB	260	3.1	61%	36%		CWB	424	2.9	58%	31%
	Marvel	942	3.2	67%	40%		Marvel	1630	3.1	65%	35%
	All	2114	3.3	71%	45%		All	3715	3.1	66%	40%
11	NBA	406	3.3	76%	48%						
	States	1019	3.0	59%	40%						
	AuMa	654	3.1	68%	36%						
	Metal	642	3.0	66%	37%						
	CWB	712	2.9	60%	30%						
	Marvel	2754	3.0	63%	33%						
	All	6187	3.1	65%	37%						

Table 2.3: Size (number of paraphrase pairs generated), average score, % of pairs with a score above 3 (paraphrases), and % of pairs with a score above 4 (high quality paraphrases) for the different domains with different thresholds

old we find 512 clusters over all domains, a little over 60% of the number of paraphrases. The distribution of their sizes is Zipfian: a few very large clusters, dozens of increasingly smaller medium-sized ones and a long tail of clusters that contain only two templates.

The vast majority of paraphrase pairs come from sentences that were not originally paraphrases (i.e, sentences that originally had different entities). With a 9.5 threshold, 86% of paraphrases answer that criteria. While that number varies somewhat across thresholds, it is always above 80% and does not consistently increase or decrease as the threshold increases.

While we wanted to show a meaningful comparison with another method from previous work, none of them do what we are doing here - extraction of sentence-size paraphrasal templates from a non-aligned corpus - and so a comparison using the same data would not be fair (and in most cases, not possible). While it seems that providing the results of human evaluation without comparison to prior methods is the norm in most relevant prior work (Ibrahim et al., 2003; Paşca and Dienes, 2005; Bannard and Callison-Burch, 2005; Fujita et al., 2012), we wanted to at least get some sense of where we stand in comparison to other methods, and so we provide a list of (not directly comparable) results reported by other authors in Table 2.4.[4] While it is impossible to meaningfully compare and rate such different methods, these numbers support the conclusion that our single-corpus, domain-agnostic approach achieves a precision that is similar to or better than other methods. We also include the *paraphrase per sentence (PPS)* value - the ratio of paraphrases extracted to the number of input sentences of the corpus - for each method in the table. We intend this figure as the closest thing to recall that we can conceive for mining paraphrases. However, keep in mind that it is not a comparable figure across the methods, since different corpora are used. In particular, it is expected to be significantly higher for parallel corpora, where the entire corpus consists of potential paraphrases (and that fact is reflected in Table 2.4, where some methods that use parallel corpora have a PPS that is an order of magnitude

[4]We always show the results of the best system described. Where needed, if results were reported in a different way than simple percentages, we use averages and other appropriate measures. Some previous work defines related sentences (as opposed to paraphrases) as positives and some does not; we do not change their numbers to fit a single definition, but we use the harsher measure for our own results

higher than other methods).

	Corpus type	Prec.	PPS
This paper, $\tau = 8$	Unaligned	94%	0.005
This paper, $\tau = 9.5$	Unaligned	82%	0.013
This paper, $\tau = 11$	Unaligned	65%	0.1
Barzilay and McKeown (2001)	Parallel	86.5%	0.1 *
Ibrahim et al. (2003)	Parallel	41.2%	0.11 *
Pang et al. (2003)	Parallel	81.5%	0.33
Barzilay and Lee (2003)	Comparable	78.5%	0.07
Bannard and Callison-Burch (2005)	Parallel bilingual	61.9%	n/a **
Zhao et al. (2009)	Parallel or Comparable	70.6%	n/a **
Wang and Callison-Burch (2011)	Comparable	67%	0.01
Fujita et al. (2012)	Parallel bilingual + unaligned	58%	0.34
Regneri and Wang (2012)	Parallel	79%	0.17

* These papers do not report the number of sentences in the corpus, but do report enough for us to estimate it (e.g. the number of documents or the size in MB)

**These papers do not report the number of paraphrases extracted, or such a number does not exist in their approach

Table 2.4: Comparison with the precision and paraphrases generated per input sentence (PPS) of relevant prior work

2.3.1 Discussion and Examples

The first thing to note about the results shown in Figure 2.1 is that even for the highest threshold considered, which gives us approximately a $\times 20$ improvement in size over the smallest threshold considered, all domains except CWB achieve an average score higher than 3, meaning most of the pairs extracted are paraphrases (CWB is close - a little over 2.9 on average). For the lowest threshold considered, all domains are at a precision above 88%, and for three of them it is 100%. In general, across all domains, there seems to be a significant drop in precision (and a significant boost in size) for thresholds between 9 and 10, while the precisions and sizes are fairly stable for thresholds between 8 and 9 and between 10 and 11. This result is encouraging: since the method seems to behave fairly similarly for different domains with regard to changes in the threshold, we should be able to expect similar behavior for new domains as the threshold is adjusted.

The magnitude of precision across domains is another matter. It is clear from the results that some domains are more difficult than others. The Metal domain seems to be the hardest: it never achieves an average score higher than 3.8. For the highest threshold, however, Metal is not different from most of the others, while CWB is significantly lower in precision. The reason seems to be the styles of the domain articles: some domains tend to have a more structured form. For example, each article in the States domain will discuss the economy, demographics, formation etc. of the state, and we are more likely to find paraphrases there (simply by virtue of there being 50×49 possible candidates for each of these). Articles in the Metal domain are much less structured, and there are fewer obvious paraphrase candidates. In CWB articles, there are a few repetitive themes: the outcome of the battle, the casualties, the generals involved etc., but beyond that it is fairly unstructured. This "structurality" of the domain also affects the number of paraphrases that can be found, as evident from the number of paraphrases found in the states domain in Table 2.3 as compared with the (much larger) Metal and CWB domains.

Table 2.5 shows a number of examples from each domain, along with the score given to each by the annotators. In an informal error analysis, we saw a few scenarios recurring in low-scored pairs. The Metal example at the bottom of Table 2.5 is a double case of bad sense disambiguation: the *album* in the second sentence ("Pyromania" in the original) happened to have a name that is also a pathological state. In addition, the *number* in the second sentence really was a date ("1980"). If we had correctly assigned the senses, these two templates would not be paraphrase candidates. The process of grouping by type is an important part of improving precision: two unrelated sentences can be misleadingly similar in the vector space, but it is less likely to have two sentences with the exact same entity types and a high vector similarity that are not close in meaning.

Another scenario is the one seen in the NBA example that was scored as 1. Here the

Sc.	Domain	Templates
5	States	Per dollar of federal tax collected in [date], [american state] citizens received approximately [money] in the way of federal spending.
		In [date] the federal government spent [money] on [american state] for every dollar of tax revenue collected from the state.
	AuMa	Designed as a competitor to the [car 1], [car 2] and [car 3].
		It is expected to rival the [car 1], [car 2], and [car 3].
4	CWB	Federal casualties were heavy with at least [number 1] killed or mortally wounded, [number 2] wounded , and [number 3] made prisoner.
		Federal losses were [number 1] killed, [number 2] wounded, and [number 3] unaccounted for – primarily prisoners.
	NBA	For the [date] season, the [basketball team] moved into their new arena , the [place], with a seating capacity of [number].
		As a result of their success on the court, the [basketball team] moved into the [place] in [date], which seats over [number] fans.
3	Marvel	[imaginary being 1] approached [imaginary being 2], hunting for leads about the whereabouts of the X-Men.
		[imaginary being 1] and [imaginary being 2] eventually found the X-Men and became full time members.
	Metal	In [date], [band] recorded their third studio album, "[album]", which was produced by Kornelije Kovač.
		[band] released their next full-length studio album, "[album]" in [date].
2	Auma	[company] and its subsidiaries created a variety of initiatives in the social sphere, initially in [country] and then internationally as the company expanded.
		[company] participated in [country]'s unprecedented economic growth of the 1950s and 1960s.
	Marvel	Using her powers of psychological deduction, she picked up on [first name 1]'s attraction towards her, and then [first name 2] admits she is attracted to him as well.
		While [first name 1] became shy, reserved and bookish, [first name 2] became athletically inclined, aggressive, and arrogant.
1	NBA	Though the [date] 76ers exceeded many on-court expectations, there was a great deal of behind-the-scenes tension between [person], his players, and the front office.
		After an [date] start, with [person] already hurt, these critics seemed to have been proven right.
	Metal	Within [number] hours of the statement, he died of bronchial pneumonia, which was brought on as a complication of [pathological state].
		With the album's massive success, "[pathological state]" was the catalyst for the [number] pop-metal movement.

Table 2.5: Examples of template pairs and their scores

senses were chosen correctly, but the level of the hierarchy chosen for the *person* slot was too high. If instead we had chosen *basketball coach* and *basketball player* for the two sentences respectively, they would not be considered as paraphrase candidates (and note that both meanings are implied by the templates). This sort of error does not create a problem (in our evaluation, at least) if the more accurate sense is the same in both sentences - for example, in the other NBA example (which scored 4), the *place* slot could be more accurately replaced with *sports arena* in both templates.

Cases where the types are chosen correctly do not always result in perfect paraphrases, but are typically at least related (e.g. in the examples that scored 2, and to a lesser extent those that scored 3). That scenario can be controlled using a lower threshold, with the downside that the number of paraphrases found decreases.

2.4 Conclusion and Future Work

In this chapter, we developed a method for extracting paraphrasal templates from a plain text corpus in three steps: templatizing the sentences of the corpus; finding the most appropriate type for each slot; and clustering groups of templates that share the same set of types into paraphrasal sub-groups. We conducted a crowd-sourced human evaluation and showed that our method performs similarly to or better than prior work on mining paraphrases, with three major improvements:

1. We do not rely on a parallel or comparable corpus, which are not as easily obtained

2. We produce typed templates that utilize a rich, fine-grained type system, which can make them more suitable for generation

3. By using such a type system we are able to find paraphrases from sentence pairs that are not, before templatization, really paraphrases

Many, if not most, of the worst misidentifications seem to be the result of errors in the second stage of the approach - disambiguating the sense and specificity of the slot types. In this paper we focused on a traditional distributional approach that has the advantage of being explainable, but it would be interesting and useful to explore other options such as word embeddings, matrix factorization and semantic similarity metrics. We leave these to future work.

Another task for future work is semantic alignment. Our approach discovers paraphrasal templates without aligning them to a semantic meaning representation. These are perfectly usable by summarization, question answering, and other text-to-text generation applications; Chapter 5 describes how we use them within a hybrid concept-to-text and text-to-text generation pipeline. However, it would be useful for traditional concept-to-text generation and other applications to have each cluster of templates aligned to a semantic representation of the meaning expressed. Since we already discover all the entity types involved, all that is missing is the proposition (or frame, or set of propositions); this seems to be a straightforward, though not necessarily easy, task to tackle in the future.

Chapter 3

Taxonomy Induction

Thesauri are useful resources for many NLP applications. Most relevant to our purposes, taxonomic thesauri which contain synonymy and hypernymy relations are important for NLG systems which must make decisions regarding lexical choice, aggregation and message construction. WordNet (Fellbaum, 1998) is one such thesaurus which has many uses in generation (Jing, 1998), but its set of concepts (called *synsets*) is quite limited. It does not contain many domain-specific concepts, nor does it contain technical and cultural concepts that emerged very recently. This chapter describes our approach to automatically building a large, WordNet-like taxonomic thesaurus from Wikipedia. This work is also described in (Biran and McKeown, 2013b).

The English Wikipedia has over 4 million articles, and over 8.6 million titles if *redirects*, which are alternative titles for the articles, are included. These titles are essentially lexical terms referring to concepts. Crucially, it contains articles describing concepts from a large variety of subject domains and is very quickly updated with articles about new concepts. In this chapter, we'll use the *science* domain as an example - an interesting domain for a couple of reasons. First, it is a fast-paced domain: many new technologies came to prominence in the last decade, and WordNet does not contain those. For example, Wikipedia contains articles with titles such as *Supersymmetric String Theory*, *Gorilla Glass* and *Sentiment Analysis*, all of which are missing from WordNet. Second, unlike many domains which are more entity-focused, it heavily involves concepts in addition to entities.

While there have been attempts to build ontologies from Wikipedia, these tended to focus (in their optimization and evaluation) on entities such as people, places and events. There is still a need for a WordNet-like taxonomy which contains accurate synonymy and hypernymy relations for highly specialized terms from specialized domains such as the various sub-domains of science.

Unlike most previous approaches, which tend to rely on WordNet's hierarchy and/or on Wikipedia's pseudo-hierarchy of *categories*, we frame the problem as a binary classification task for a pair of Wikipedia article titles - deciding whether the term representing the concept in the first article is a hypernym of the term representing the second or not. This enables us to handle specialized concepts which are far from the established concepts in the WordNet hierarchy.

WordNet-like taxonomies behave in some ways as a dictionary, and in others as an ontology. To avoid confusion, we define the main terms we use in this chapter and what they correspond to:

- A *concept* in computational ontologies is a unique semantic entity. We assume that WordNet synsets correspond to concepts. Another assumption we make is that each Wikipedia article describes something analogous to a concept; this assumption does not work for some types of articles (e.g. Template articles), and we remove such articles before processing, as explained in Section 3.2.

- A *term* is a lexical entity (word or combination of words) used to refer to a concept. Each WordNet synset contains multiple lexicalizations (synonyms) which all refer to the concept represented by the synset. We treat Wikipedia article titles as terms referring to the concept described in the article. In addition to the main title, Wikipedia has multiple additional *redirect titles* referring to each article. We do not *a priori* treat these as synonyms, as they are often hypernyms, hyponyms or even terms referring to distinct (though related) concepts (for example, as of the time we performed the experiments described in this chapter, *Disambiguation* redirects to *Word Sense Disambiguation*; *nano-SIM* redirects to *Subscriber Identity Module* (SIM); and *Sheep Sounds* redirects to *Sheep*).

- *Relations* in this chapter are semantic relations between pairs of terms - specifically, synonymy and hypernymy. This is in contrast to the use of the word in ontologies where relations occur between pairs of concepts.

The following are a few examples of relations from the science domain that do not appear in WordNet and which our method correctly finds:

- *Gene Silencing* is a hypernym of *RNA Interference*
- *Graph Property* is a hypernym of *Clustering Coefficient*
- *Conditional Random Field* and *CRF* are synonyms

We will use these examples to illustrate the limitations of other methods in the following section.

3.1 Related Work

There have been several endeavors to extend WordNet with concepts from Wikipedia. Because WordNet has some of the properties of an ontology, most work on extending WordNet with Wikipedia concepts was in the context of creating an ontology. Although our work is different in that we focus on extending only the taxonomic relations between the terms, this related work is still very relevant. There have also been attempts to create ontologies directly from Wikipedia in various ways, and we discuss those as well.

There have been three main approaches to building ontologies from Wikipedia. The first is the one which was used to build Yago (Suchanek et al., 2007), a large ontology (over 10 million concepts) based on WordNet and extended with concepts from Wikipedia and other resources. Its hypernymy hierarchy (a relation called *subClassOf*) is derived by matching articles with existing WordNet synsets using the lexical and syntactic properties of the title. This approach works well for some complex entities: a title like "American people in Japan" contains the head compound *people* which matches the WordNet synset *Person/Human*. It does not work as well for concepts that are not entity-focused (e.g., sets of entities or entity types), where titles tend to be less clearly related. For example, Yago contains the concepts *Clustering Coefficient* and *RNA Interference*, because they are titles of Wikipedia articles; but these concepts are not part of the *subClassOf* hierarchy, because their titles are not lexically similar to *Graph Property* and *Gene Silencing*, respectively.

Another approach is to build ontologies from the *infoboxes* of Wikipedia articles. Perhaps the most famous is DBPedia (Auer et al., 2007), an ongoing community effort maintaining a knowledge base of over 4.5 million entities. DBPedia utilizes scraping algorithms to extract (as well as post-process) structured data from the infoboxes of Wikipedia pages and transform them to RDF triples. Wu and Weld (2008) use Markov Logic Networks to link entities and properties extracted from infoboxes to WordNet and create a full ontology (the DBPedia team does this, and other linking of DBPedia with other resources, manually). While infoboxes are very useful, as they contain structural human-curated information, their main drawback is their relative sparsity. They commonly occur in articles of popular entities (people, places and so on), but not in the articles of less popular entities, and not in most non-entity concepts (for example, the three scientific concepts we mentioned at the end of the last chapter cannot be found in any infobox-derived ontology).

The third approach is to use the *category hierarchy* of Wikipedia. Categories are hand-curated tags given to Wikipedia articles, which have a somewhat hierarchical structure (they are not a true hierarchy: there are cross-references and even cycles in the category graph). Ponzetto and Navigli (2009) link Wikipedia categories to existing WordNet synsets, leveraging the category structure to enrich WordNet with concepts from Wikipedia. Wikipedia categories are mostly thematic, with no strict hierarchical structure and do not represent a taxonomy, but they do tend to be somewhat hierarchical for concepts low in the hierarchy (i.e., more specific concepts). For example, *Public transport in Stockholm* is in the category *Public transport in Sweden* which is in the category *Public transport*, and the latter corresponds to a synset in WordNet. However, this is not true for many concepts in specialized domains (e.g. scientific domains), where even the more general concept does not appear in WordNet. For example, *Clustering coefficient* is in the category *Graph invariants*, but the categories above that are purely thematic, and WordNet does not contain a synset for *Graph invariant*. Similarly, the term *CRF* is the title of a disambiguation page, which does not belong to any categories and so would not be linked to *Conditional Random Field*.

Syed and Finin (2010) build a taxonomy by matching each Wikipedia article to a Word-

Net synset as a hypernym-like superclass. Their method relies on the synset-category mappings of (Ponzetto and Navigli, 2009), extending it with information obtained from the hyperlink structure of the Wikipedia articles. However, this approach is still limited by the choice of categories for each article. In addition, it does not work as well for articles with a small number of hyperlinks, which is typical of more specialized domain articles.

In addition to not being optimal for specialized domains, these three approaches all have in common that in attempting to extend WordNet using Wikipedia they rely on the structural information in WordNet directly (the only exception is DBPedia, where the linking was done manually). This generally means that the further down the hierarchy a term is (that is, the further it gets from the most specific hypernym available in WordNet) the less accurate the constructed taxonomy becomes with regard to its relations. This again works well for some entities, where WordNet contains reasonably specific concepts (e.g., occupations and nationalities for people, industries for organizations) but not too well for specialized domain concepts.

In contrast, in our approach, WordNet is only used to provide the labels for very few relations ($5,000$) that are used in training and (separately) in evaluation. However, these relations are all considered individually. We do not rely on the WordNet hierarchical structure as a whole; instead, we learn to classify the relation between a pair of terms using only information from their Wikipedia article content. This makes our method more robust with regard to very specific concepts. Evaluating other methods using gold data from WordNet may be biased, because concepts from WordNet (even if they are not used directly in ontology construction) are inevitably close to other concepts in WordNet. It can be expected that for more highly specialized concepts, these methods will not perform as well. In our approach, there is nothing special about a relation whose concepts appear in WordNet, and performance on those should give a good indication of performance on other relations (perhaps with the caveat that concepts which appear in WordNet may have larger corresponding articles on average).

One other work that (like us) takes the approach of classifying the relation between two terms is (Do and Roth, 2010). However, their method relies on Wikipedia's categories, which as mentioned earlier is problematic for specialized domain concepts. In addition, they do not utilize the important link structure of Wikipedia.

In addition to ontology and taxonomy building, there is work mapping words from Wikipedia articles to particular senses within WordNet using WSD techniques (Mihalcea, 2007; Milne and Witten, 2008). While superficially related, these are not relevant since they only match articles to existing WordNet synsets. In contrast, we create a thesaurus specifically containing terms that are not in WordNet.

There is also literature on classifying taxonomic relations based on textual patterns in any corpus (Caraballo, 1999; Girju et al., 2003; Buitelaar et al., 2005; Snow et al., 2006). While these methods have the potential of reaching concepts and entities that ours cannot (since we are limited by the articles that exist in Wikipedia), they are far less precise and would be difficult to use for most NLP needs (e.g., for generation) in practice.

3.2 Method

Since we want our terms from Wikipedia to refer to concepts, we remove from the Wikipedia corpus all the pages whose title begins with a Wikipedia special prefix. These prefixes are single words followed by a colon, and denote a special type of Wikipedia page, such as *File*, *Category* or *Template*. We also remove all pages whose title does not contain at least one English letter character.

We define a Wikipedia term as any Wikipedia article title and any redirect title which passes the filters above. This lexical definition is motivated by the need to find synonymy and hypernymy. It also makes evaluation (using the gold data in WordNet) more straightforward. To keep things simple, we ignore senses, assuming that each lexical term corresponds to a singular sense or to the most common sense. While word sense disambiguation has been a major part of some related work, it is less crucial for our purposes since the Wikipedia article almost always describes the most common sense (Mihalcea, 2007). In addition, specific terms (i.e., terms which are not in WordNet) are less likely to be ambiguous than general terms. We hypothesize that the Wikipedia article itself describes the concept that is referred to by the term.

We define a WordNet term as any term (synonym) participating in any noun synset in WordNet. Wikipedia terms are then matched to WordNet terms lexically, with some pre-processing: we lowercase the Wikipedia titles, replace underscores with spaces, remove diacritics from unicode characters and remove text in parentheses (which are commonly used in Wikipedia to disambiguate senses).

Using our definition, there are 117,092 WordNet terms. The total number of potential terms from Wikipedia[1] is 9,096,022, which covers 73.62% of the WordNet terms. WordNet has 494,892 hypernym and synonym relations between all terms. The set of all potential relations from the Wikipedia term set (which is $9,096,022^2$ in size) covers 63.71% of those.

To conveniently define our task, we introduce the taxonomic relation of *synoronymy* (from ancient Greek *σύνορο - boundary*). Synoronymy designates a term as the lower boundary on the taxonomic line for another term, i.e., term A is at least a synonym, and possibly a hypernym, of term B. Logically, synoronymy is defined as:

$$Synoronym(A,B) \iff Synonym(A,B) \vee Hypernym(A,B)$$

Using that definition, we formulate our task as a binary classification over all potential relations from the Wikipedia term set: for each ordered pair of terms, we classify whether the first is a synoronym of the second or not. We perform this classification for both possible orderings of each term pair, and derive the usual taxonomic relations from the two results: two-way synoronymy is synonymy, and one-way synoronymy is hypernymy (in the same direction). We evaluate our performance in this task on a dataset sampled from that subset of the Wikipedia terms which also exist in WordNet.

To determine the relations for all Wikipedia terms, we would be required to evaluate over 82 trillion potential relation data points, and be forced to either somehow reduce the space or parallelize aggressively (and still need to be very patient). For our purposes, we will always work with significantly smaller domains, where the space is small enough to work with (for example, see the way we use this method for lexical choice and taxonomic message building in Chapter 5). In our evaluation in this chapter we present results both on domain-specific subsets and on general sampled subsets.

[1]As of April, 2013

For the remainder of this section, we describe the features used by our classifier. In the next section we will describe the experimental setup in more detail.

3.2.1 Features

We extract fourteen features of four general types. For most of these, it is essential that each term in the pair corresponds to a Wikipedia article. Each term matches either the article title, or a redirect title that redirects to the article. From here on, we will refer to terms and articles interchangeably (e.g., "the first article" instead of "the article referred to by the first term" in a pair of terms to be classified), with the understanding that the article in question is always the one whose title (or redirect title) is the term. The features are described in this section and summarized in Table 3.1.

Type	Feature
Hyperlink	First links to second
	Second links to first
	Outgoing link similarity
	Outgoing links in first shared by second
	Outgoing links in second shared by first
	Incoming link similarity
	Incoming links in first shared by second
	Incoming links in second shared by first
Text	Bag-of-words similarity
Redirect	First redirects to second
	Second redirects to first
	Both redirect to third
Titles	Length difference
	Word overlap

Table 3.1: The features used in the taxonomic relation classifier

3.2.1.1 Features from the hyperlink structure

One of the powerful aspects of Wikipedia is its hyperlink structure. Given the assumption that article A links to article B only if the information in B is related to or somehow assists in understanding the information in A, the intuition is that two articles having a semantic relation will more often link to one another, as well as to the same (third) articles, than will two unrelated articles. The Wikipedia hyperlink structure has been used to compute similarity between articles, for example by Syed and Finin (2010) and Yazdani and Popescu-Belis (2010).

We utilize the natural graph structure of hyperlinks between Wikipedia articles to build the following eight features:

1. First article links to second (yes or no)
2. Second article links to first (yes or no)
3. The cosine similarity between the outgoing links of the articles
4. The ratio of outgoing links in the first article shared by the second article
5. The ratio of outgoing links in the second article shared by the first article
6. The cosine similarity between the incoming links of the articles
7. The ratio of incoming links in the first article shared by the second article
8. The ratio of incoming links in the second article shared by the first article

Wikipedia links contain two bits of information: the title of the article they link to, and the text of the hyperlink as it appears in the referring article. For features (1) and (2), we allow both: that is, even if a hyperlink links to a third article, but uses the relevant article's title in the text,[2] we count that as a link to the relevant article. For the other features, we use only the title of the actual linked articles. The reason is that in features (1) and (2) we want to measure something different than in the rest: whether or not one of the articles

[2]For example, a link for the article *International Phonetic Alphabet* may have only "pronunciation" in the text, which is the title of an article about the phenomenon which is modeled by the IPA

mentions the other directly (hyponyms often mention their hypernyms, while hypernyms sometimes list their hyponyms). An article being mentioned by name in a hyperlink, even when the link goes elsewhere, answers that criteria. The other features are intended to capture the similarity of the two articles based on how related the links to/from them are, and so using the text is less relevant (and that information would be captured to some extent by the feature in the next category instead).

3.2.1.2 Features from the text of the articles

For each article, we build a bag-of-words vector. These vectors are used to compute the cosine similarity between the two articles of a pair, which we use as a feature.

The intuition behind this central feature is that articles having a semantic similarity will also have a higher lexical similarity. This is the same intuition behind distributional similarity (Church and Hanks, 1990), which is that terms surrounded by similar context tend to be semantically related. In this case, the context does not surround the terms but is in the body of the articles corresponding to them. Lexical similarity between Wikipedia articles has been used successfully to link articles, for example by Yazdani and Popescu-Belis (2010).

3.2.1.3 Features from the redirect structure

Wikipedia contains a list of redirects from multiple alternative titles to each article. We use those to build three binary features:

1. The first term redirects to the second term's article (yes or no)
2. The second term redirects to the first term's article (yes or no)
3. Both terms redirect to the same, third article (yes or no)

As mentioned earlier, redirect titles are often synonyms, hypernyms or hyponyms of the main title of the article they redirect to. While it is not consistent enough to use as a rule, this structure can be taken advantage of in features.

3.2.1.4 Features from the terms (i.e. the article titles)

We derive two features from the terms themselves (which correspond to article titles or redirect titles):

1. The difference between the number of words in the two terms
2. The number of words which overlap in the two terms

In some cases, the terms themselves can point at the relation among them. In particular, hypernyms are sometimes lexical subsets of their hyponyms (*String Theory* is a hypernym of *Super String Theory*; *Leukemia* is a hypernym of *Lymphocytic Leukemia* which in turn is a hypernym of *B-cell Chronic Lymphocytic Leukemia*).

3.3 Evaluation

Our training, development and test data sets all consist of ordered pairs of terms from Wikipedia which also appear in WordNet. The label is positive if the first term in the pair is a synoronym of the second. The positive instances are taken from the hypernymy and synonymy relations in WordNet. To get negative samples, we randomly paired terms from WordNet that have no relation between them.

We train a linear-kernel SVM classifier on a small balanced training set of $5,000$ labeled pairs. We initially experimented with a much larger set of $100,000$ pairs, but found the difference in results on the development set to be insignificant. We used a balanced development set of $186,000$ pairs in tuning. We evaluate on a large unbalanced test dataset of 10 million pairs. Using the number of WordNet's total potential relations ($117,092^2$) and the number of its true relations (494,892), we estimate the ratio of real relations in the natural set of all potential relations to be around 0.0036%. We multiply this fraction by $1,000$ to make the ratio of positive samples in our test set more significant (while still keeping it small). This multiplication can be excused as an estimate of the order of magnitude of the expected ratio is in a domain-specific dataset (as shown in the next paragraph). The test set is then built using 360,000 sampled true relations from WordNet, while the rest are randomly paired concepts (which appear in WordNet but have no relation between them).

To illustrate our performance on a domain-specific subset, we also constructed a *science* test set using Wikipedia's category hierarchy. In this data set, we included only terms with corresponding articles in a category which is a descendent of the *Science* category at a depth of no more than 20, but are *not* descendants of one of the following categories with a depth of 5 or less: *People*, *Places*, *History*, *Chronology*, *Music*, *Film* and *Sports*. These exclusions are required because descendants of the *Science* category include articles for entities such as scientists and universities, certain historical dates/eras, and expansions of the technologies used in the music, film and sports industries to include entities from these fields (songs, bands, movies...) which then completely overwhelm the data set in size. The depth restrictions are necessary because the category graph is cyclic. In addition to illustrating our system's performance in a specific domain, this test set is important in that it features negative samples that are not entirely random, since they are at least thematically related.[3] The size of this set is 258,971, and it is unbalanced with about 10% positive samples. Note that we use the same classifier (trained on the same unrestricted training set) when evaluating on all test sets, including this one.

To illustrate our approach's advantage over naive methods, we include the results for two baselines. The first uses only the term names and makes predictions based on the Levenshtein distance between them (predicting synonym for distance < 8, hypernym for distance < 12, and none otherwise). The second predicts the relation type based on the lexical cosine similarity between the articles (predicting synonym for similarity > 0.1, hypernym for > 0.05, and none otherwise). The thresholds in both baselines were manually tuned to optimize f-measure on the development set.

In addition, we compare our performance with that provided by querying two leading publicly available ontologies that were constructed using Wikipedia's category hierarchy and infoboxes: Yago (Suchanek et al., 2007) and DBPedia (Auer et al., 2007).

We show three binary evaluations for each data set. The main evaluation, where a positive answer means the (ordered) pair has a synoronymy relation, is shown in Table 3.2.

[3]Some examples of thematically related concepts that are not taxonomically linked are siblings (*Canine* : *Feline*), intersecting but distinct concepts (*Biochemistry* : *Bioengineering*), and meronyms (*Vertebrate* : *Vertebra*)

The hypernymy evaluation in Table 3.3 and the synonymy evaluation in Table 3.4 are additional evaluations over those pairs that were judged as having a synoronymy relation in the first evaluation, and a positive answer means the pair is a hypernym or a synonym, respectively. These evaluations show the actual taxonomic accuracy we achieve. Recall that we mark as synonyms those pairs that are determined to have synoronymy in both directions, while those that have it only in one direction are hypernyms. The synonymy evaluation does not include the ontologies as baselines, since they do not contain synonyms. We found the results to be statistically significant using a standard t-test.

3.4 Analysis

The first thing to note is that in general, the SVM classifier operates as high-precision, lower-recall system. On the synonymy task, precision is extremely high (just under 100%) while retaining a reasonable recall even on the large test set. This is important, since a high precision is crucial to maintaining coherence in tasks such as lexical choice.

The classifier beats both baselines on the main task. The lexical baseline does quite well on the synonymy task, but its performance deteriorates on the unbalanced test sets while the classifiers' performance actually significantly increases due to its high-precision nature. The lexical baseline is much worse at the hypernymy task than at the synonymy task.

While the ontologies (Yago and DBPedia) offer incredibly high precision in all cases, their recall is low (often less than 1% in DBPedia). This is because they focus on entities that are well defined through the category hierarchy and/or infoboxes, which most Wikipedia articles are not.

Overall, the classifier beats both baselines and both ontologies in both tasks on both test sets. Most importantly, we achieve a relatively high performance on the domain-specific test set, which is our main goal: an approach that can automatically build accurate domain-specific taxonomies for use in generation tasks such as lexical choice, aggregation and message building.

Crucially, the training data does not have to be domain-specific: our approach achieves good performance on a domain-specific test set, even when trained on a small data set

	Dev P	Dev R	Dev F	Test P	Test R	Test F	Sci P	Sci R	Sci F
Naive base	57.41	69.44	62.85	4.76	69.38	8.91	13.74	80.08	23.45
Lex. base	97.14	17.89	30.21	54.31	16.23	24.99	70.22	19.13	30.06
DBPedia	100	0.25	0.5	96.33	0.26	0.52	98.72	1.78	3.5
Yago	100	15.23	26.44	99.96	14.5	25.33	100	29.19	45.19
SVM	98.75	46.18	**62.93**	66.03	42.95	**52.05**	64.81	61.23	**62.97**

Table 3.2: **P**recision, **R**ecall and **F**-measure obtained for each data set for the **main task (synoronymy)**. Results are shown for the development set, the large general test set, and the science test set.

	Dev P	Dev R	Dev F	Test P	Test R	Test F	Sci P	Sci R	Sci F
Naive base	26.46	37	30.85	1.81	37.78	3.46	7.15	38.22	12.04
Lex. base	55.84	0.89	1.75	10.25	0.88	1.63	31.52	2.24	4.19
DBPedia	99.54	0.49	0.97	95.98	0.48	0.95	98.72	2.36	4.6
Yago	99.68	29.67	45.73	99.41	26.42	41.74	99.91	38.34	55.42
SVM	46.12	99.96	**63.12**	27.74	99.79	**43.41**	55.47	99.33	**71.19**

Table 3.3: **P**recision, **R**ecall and **F**-measure obtained for each data set for the **hypernymy** task. Results are shown for the development set, the large general test set, and the science test set.

	Dev P	Dev R	Dev F	Test P	Test R	Test F	Sci P	Sci R	Sci F
Naive base	50.76	68.61	58.35	7.02	66.19	12.7	7.65	64.26	13.67
Lex. base	68.41	97.83	**80.52**	43.49	97.75	60.2	23.99	92.31	38.08
SVM	99.92	30.15	46.33	99.65	44.58	**61.6**	97.65	56.12	**71.28**

Table 3.4: **P**recision, **R**ecall and **F**-measure obtained for each data set for the **synonymy** task. Results are shown for the development set, the large general test set, and the science test set.

sampled from the entire set of WordNet lexicalizations (this $5,000$ instance set is very unlikely to have more than a few scientific concepts, at most).

3.4.1 Feature Analysis

To better understand how the different features contribute to classification, we evaluated classifiers that were trained with a single feature on the science test set (for the main synoronymy task). The results are shown in Table 3.5.

	P	R	F
First links to second	96.9	29.53	45.27
Second links to first	96.38	27.93	43.31
Outgoing link similarity	9.98	100	18.15
Out. links in 1st shared by 2nd	60.58	37.43	46.27
Out. links in 2nd shared by 1st	9.98	100	18.15
Incoming link similarity	77.35	43.09	55.34
Inc. links in 1st shared by 2nd	83.2	35.72	49.98
Inc. links in 2nd shared by 1st	75.93	18.89	30.25
Bag-of-words similarity	80.29	15.77	26.36
First redirects to second	100	0.26	0.53
Second redirects to first	100	0.31	0.61
Both redirect to third	100	0.29	0.57
Length difference	10.16	95.4	18.36
Word overlap	96.89	14.96	25.92

Table 3.5: **P**recision, **R**ecall and **F**-measure obtained with a single feature for the main task on the science test set.

One pattern that emerges is that link features are the most useful, with the exception of two of the outgoing link features: *outgoing link similarity* and *outgoing links in second shared by first*. In fact, for those two, the classifier simply marks every pair as a synoronym. The third outgoing link feature, *outgoing links in first shared by second*, is useful. This makes sense: the outgoing links from the potential hypernym article (the "first") should be repeated by the hyponym article (the "second") to a high degree. We expect the article "Marsupial", for example, to contain many of the links in "Mammal": general biology and evolutionary terms, terms related to all vertebrates, etc; but the opposite is not true - many links in "Marsupial" will be specific to that type of mammal rather than to mammals in general.

Outgoing links in some sense model what the article is about; incoming link features, on the other hand, are all important because they model something different: what the article is relevant to. Hyponyms and their hypernyms will often be relevant to the same things, and indeed the most powerful feature is exactly that - the *incoming link similarity.* Finally, the two features looking at whether or not the articles link to one another are important indicators for the obvious reason that articles that are taxonomically related are more likely to be linked.

Most other features, namely the three redirect features, *word overlap* and to some extend also *bag-of-words similarity* operate as high-precision, low recall assisting features. These features are good at identifying particular special cases - high lexical similarity, highly similar title names (which are often specifications, e.g. "Physics" and "Particle Physics"), and cases where the terms are tied by a redirect relationship. Those are all strong but uncommon indicators for a taxonomic relationship. The last feature, *length difference*, does not give much improvement over marking all pairs as having a relation.

3.5 Output and Error Analysis

In this section we provide some examples of correct and incorrect decisions made by the classifier, not necessarily from the scientific domain, and provide some informal error analysis. The examples are presented in Table 3.6.

The lower five rows of Table 3.6 are examples of incorrect decisions. In two of these cases, we believe that it is in fact WordNet that is wrong in its classification: *Werlhof's Disease* or *Idiopathic Thrombocytopenic Purpura* is indeed a specific type of *Thrombocytopenic Purpura*, as our classifier predicts, and *Dinornis Giganteus* is a subspecies of *Giant Moa* (Dinornis).

First term	Second term	Real rel.	Predicted rel.
arthropod genus	genus phalangium	hypernym	hypernym
metacarpophalangeal joint	knuckle joint	hypernym	hypernym
living dead	zombie	synonym	synonym
vldl	very low density lipoprotein	synonym	synonym
bleeding	haemorrhage	synonym	synonym
gear	rigging	hypernym	none
articles of confederation	mahican	none	hypernym
giant moa	dinornis giganteus	synonym	none
thrombocytopenic purpura	werlhof's disease	synonym	hypernym
shopping list	grocery list	hypernym	synonym

Table 3.6: Examples of correct and incorrect decisions made by classifier.

Such errors in WordNet may have some effect on our results.

In the case of *Articles of Confederation* and *Mahican*, a Native American tribe, it seems that the classifier was thwarted by accidentally similar content in the two articles, which both describe the American Revolutionary war and the 13 first states (and contain many similar links to these topics). In the case of *gear* and *rigging*, the fact we chose to ignore senses comes into play: the sense in which *gear* is a hypernym is not the common one (which the Wikipedia article describes). We believe that in more specialized concepts, this problem will not occur often. *Grocery List*, in Wikipedia, is a redirect title which redirects to the article *Shopping List*. In this case, Wikipedia lacks the semantic subtleties that are captured in WordNet.

3.6 Conclusion and Future Work

In this chapter we described a simple supervised method of classifying pairs of Wikipedia article titles in terms of the relation among them, covering synonymy and hypernymy. Our approach significantly outperforms the baselines on simulated target data, and achieves very high precision. Unlike previously described approaches, it does not rely on the WordNet hierarchy as a whole, but only on the properties of the individual pair. It retains a good performance on a domain-specific data set, even when trained on general, out-of-domain data.

Our method can be used for a number of NLG tasks. Chapter 5 describes how we use it for lexical choice and for creating taxonomic messages. In addition, although we do not pursue it in this thesis, it could be used for aggregation or fusion, which is an interesting direction for future work.

Aggregation is the task of deciding that multiple messages should be realized in the same sentence because of semantic relatedness (Dalianis and Hovy, 1996). In text-to-text generation, a related problem is *fusion* decision: the task of deciding whether two sentences are related enough to merge. Given two such sentences, we can find the noun terms in each and use our classifier to discover the relations between all cross-sentence pairs, which is useful in determining the similarity of content between the sentences. There is no need to construct a taxonomy in advance for this particular application. The relations can also be used in realizing the aggregation: synonyms can be collapsed into one sentence with a conjunction, and two terms sharing a hypernym can be collapsed in the same way, being replaced with the hypernym. For example, given two sentences such as:

- *RNA interference has become a valuable research tool, both in cell culture and in living organisms*
- *siRNA transfection has become a major instrument in research with potential applications in gene-therapy*

and the knowledge that *gene silencing* is a hypernym of both *RNA interference* and *siRNA transfection*, as well as the knowledge that *tool* and *instrument* are synonyms, we may want to produce a sentence like

- *Various forms of gene silencing have become valuable research tools*

Chapter 4

Discourse Planning

Discourse planning is a subtask of Natural Language Generation, concerned with determining the ordering of messages in a document and the discourse relations that hold among them (Reiter and Dale, 2000). In pipeline-style architectures of NLG, it is often placed after content selection but before micro-planning. Thus, the input is a set of unordered, unrealized messages, and the output is a set of ordered unrealized messages with specified discourse relations between them.

Early approaches used manually written rules, often based on schemas (McKeown, 1985) or on Rhetorical Structure Theory (RST) (Mann and Thompson, 1987; Hovy, 1993; Power, 2000). In the 2000's, various statistical approaches to discourse planning have emerged (Duboue and McKeown, 2001; Dimitromanolaki and Androutsopoulos, 2003; Soricut and Marcu, 2006; Konstas and Lapata, 2013), while statistical approaches to content ordering also became popular in the summarization literature (Barzilay et al., 2001; Lapata, 2003; Bollegala et al., 2005). These approaches overwhelmingly focus on determining the best order of messages using semantic conent, while discourse relations are in most cases either determined by manually-written derivation rules (or, uniquely in the case of Konstas and Lapata (2013), by derivation rules with weights learned from an aligned corpus) or completely ignored.

Discourse planning, particularly with non-trivial discourse relation planning, is difficult because it is as domain-dependent as realization and content determination, albeit in subtler ways (see, for example, Kittredge et al. (1991)). Different genres (e.g., newswire, encyclopedia, blogs) have different styles of discourse in general, and different subject domains within the genres (e.g., financial news vs. entertainment news; biographic encyclopedia entries vs. entries describing scientific concepts; personal blogs vs. political blogs) have different styles specifically. Most discourse planning approaches either rely on a manually created set of domain-specific rules to generate discourse structures, or feature corpus-driven approaches to content ordering that largely ignore discourse structure.

Meanwhile, researchers working on discourse relation disambiguation have observed that the sequence of discourse relations itself, independently of content, helps in disambiguating adjacent relations (Wellner et al., 2006; Pitler et al., 2008). Sequential discourse information has been used successfully in discourse parsing (Ghosh et al., 2011; Feng and Hirst, 2014), and discourse structure was shown to be as important for text coherence as entity-based content structure (Lin et al., 2011; Feng et al., 2014). Surprisingly, so far, discourse sequential information from existing discourse-annotated corpora, such as the Penn Discourse Treebank (PDTB) (Prasad et al., 2008) has not been used in generation.

In this chapter, we describe a corpus-driven approach to discourse planning that jointly determines the ordering of the messages and the discourse relations between them using a learned n-gram model of discourse relations (which, when learned from a domain-specific corpus, represents the discourse style of the domain through the sequences of relations commonly found in the corpus). We present an evaluation which shows the importance of discourse style for generation and the effect of the domain on the desirable style. This model and evaluation are also described in Biran and McKeown (2015a).

In order to make this approach generally useful for generation, particularly in the context of our framework presented in the second part on this thesis, we must be able to automatically extract the discourse structure of arbitrary domain corpora. Therefore, it is crucial to have an efficient way to perform non-hierarchical (sequential) parsing of discourse relations. Recently, this task has been referred to as *shallow discourse parsing*, in particular in the 2015 CoNLL shared task (Xue et al., 2015). We describe our work on shallow discourse parsing at the end of this chapter, in Section 4.5. It is also described in Biran and McKeown (2015b) and in Biran and McKeown (2013a).

4.1 Related Work

Planning the discourse structure of generated text as a task has been tackled since the 1980's. Prior to that, NLG was mainly focused on generating single phrases or sentences; early systems that generated longer texts (Meehan, 1977; Swartout, 1981; Mann and Moore, 1981) had no explicit notion of discourse, and ordering was either fixed (in canned text) or fell out of the content selection algorithm. The fact that there are recognizable patterns in discourse coherence which can be used in generation has been explored as early as (Hobbs, 1979), however.

Early approaches to discourse planning relied on manually prepared sets of rules that completely determined the discourse structure - ordering and relations - of the text. Schemata (McKeown, 1985) were the earliest discourse plans, utilizing a combination of rhetorical predicates described by Grimes (1975) and Williams (1890), and were used in many early systems (Hovy, 1987; Paris, 1988; McCoy, 1989; McKeown et al., 1997). Afterwards, discourse plans based on RST (Mann and Thompson, 1987) became popular (Hovy, 1991; Hovy, 1993; Moore and Paris, 1993; Mellish et al., 1998; Power, 2000; Bouayad-Agha et al., 2000). While not dealing with (large-scale) discourse plans specifically, it is also worth noting Elhadad and McKeown (1990) who presented a framework for generating discourse connectives individually, i.e. not as a way to lexicalize a rhetorical relation but as unique phenomenon, each with its own set of rules. Other early rule-based approaches to content planning focused more on achieving communicative goals and less on discourse coherence specifically (Appelt, 1985; Cawsey, 1992).

The first to describe a statistical approach to learning the ordering of generated messages were Duboue and McKeown (2001), who used an aligned semantic representation of the content of medical transcripts to extract content ordering patterns and constraints on the ordering of generated transcripts. In (Duboue and McKeown, 2002), they utilize genetic search algorithms to learn schema-like planners within the constraints (although the planners may include discourse information, it is not learned by this approach - the genetic algorithms are scored by a function that only takes ordering into account). Dimitromanolaki and Androutsopoulos (2003) learned to order a fixed number of facts with unique types by assigning a type classifier to each fact slot in the text and using features built from fact properties and from the sequence information.

Meanwhile, researchers working on multi-document summarization have explored approaches to sentence ordering in an output summary. Barzilay et al. (2001) defined the sentence ordering task and suggested baseline heuristic approaches based on chronology and content of the input documents. Heuristic approaches to sentence ordering were also employed by Kan and McKeown (2002) and by Okazaki et al. (2004). Statistical approaches followed: Lapata (2003) learned to order sentences by looking at feature co-occurrence in human-written texts, while Bollegala et al. (2005) defined five ordering "experts" strategies based on the previous approaches and learn the weights for each from a corpus of news articles aligned with human-selected summaries. Other approaches include Ji and Pulman (2006) who used the history of selected sentences to improve ordering and Donghong and Yu (2008) who proposed using sentence feature adjacency information in learning to order sentences. Nishikawa et al. (2010) use features from multiple previous approaches, and use Integer Linear Programming to jointly learn both sentence selection and ordering. It is worth noting that Madnani et al. (2007) concluded based on variability studies that in general, there are multiple equally good orderings for a set of sentences extracted from multiple documents, so learning ordering models directly from reference summaries may be flawed in that it implicitly treats as incorrect other possible orderings.

Work on content-based coherence models (Barzilay and Lee, 2004; Barzilay and Lapata, 2005b; Soricut and Marcu, 2006; Karamanis et al., 2009) has generally also been evaluated using the sentence ordering task. In this line of research, which is based on insight from Centering Theory (Grosz et al., 1995), ordering is determined purely on the basis of textual content - lexical items, entities, etc - as opposed to the contextual information that the approaches in the previous paragraph rely on (e.g., chronology or ordering in the input documents). This notion of coherence is often referred to as *local* coherence, as opposed to the *global* coherence which is captured by RST and other theories of document-level discourse. It has been asserted by Marcu (1997) that global coherence can be derived from local coherence. Later work on coherence (Lin et al., 2011; Feng et al., 2014) have focused on *local discourse coherence*: models of coherence that are based on local (sentence-level) discourse relations, as opposed to lexical content, and found that discourse information is as important to the notion of local coherence as entity and other lexical information.

Recently there has been more interest in end-to-end concept-to-text generation, especially approaches that learn to generate from aligned corpora of data records and text (Reiter et al., 2005; Belz, 2008; Chen and Mooney, 2008; Kim and Mooney, 2010; Angeli et al., 2010; Konstas and Lapata, 2012; Kondadadi et al., 2013). These approaches typically contain a number of learned models that are used in a sequence of local decisions, and overwhelmingly focus on content selection and realization, neglecting discourse and ordering (partially because they focus on narrow domains with a rigid structure, such as weather forecasts and sportscasting, where aligned corpora are available). One notable exception is Konstas and Lapata (2013), who use the aligned corpus to learn a set of global derivation rules based on RST as well as ordering rules based on content sequences.

Our work in this chapter differs from previous work in that we propose a statistical generative model based on local discourse coherence - i.e., a local model of coherence which is based not on content but on sentence-level discourse relations.

4.2 Motivation: Sequences of Discourse Relations

In this chapter, we utilize the discourse theory of the Penn Discourse Treebank (PDTB) (Prasad et al., 2008). The PDTB theory of discourse has three main advantages over other frameworks such as Rhetorical Structure Theory (RST) (Mann and Thompson, 1987) and Segmented Discourse Representation Theory (SDRT) (Asher and Lascarides, 2003): first, it is based on a predicate-argument representation, which means it does not rely on a fully hierarchical document structure. This lenience allows us to develop the more local model of discourse that we describe in the rest of this section and utilize in the rest of this chapter. Second, it features a hierarchical set of discourse relation categories - i.e., each relation can be viewed on an increasingly fine-grained (or abstract, going the other way) scale. This allows us to model discourse structure with varying levels of specificity, depending on the difficulty of the task or the extent to which we wish to focus on the specific relation categories. Table 4.1 shows the first two levels of the category hierarchy (the first level category is called the *class* and the second level category is called the *type* of the relation).

The third, most specific level (called the *subtype*) is not shown, and we do not use it in this chapter since many of its categories are too rare to be useful. Finally, the PDTB is the largest annotated corpus of discourse in existence, and is aligned to the Penn Treebank (Marcus et al., 1993) which contains gold syntax parses of the same documents.

Class (Level 1)	Type (Level 2)	
Comparison	Concession	Pragmatic Concession
	Contrast	Pragmatic Contrast
Contingency	Cause	Pragmatic Cause
	Condition	Pragmatic Condition
Expansion	Alternative	Instantiation
	Conjunction	List
	Exception	Restatement
Temporal	Asynchronous	Synchrony

Table 4.1: The PDTB relation category hierarchy, with level 1 classes and level 2 types. The level 3 subtypes are not shown

PDTB discourse relations can be viewed as a triple: relation type, argument 1 and argument 2 (using only these three properties, we can abstract over both explicit and implicit relations). While in principle, the discourse structure theory of PDTB allows for the two arguments of a discourse relation to be located anywhere, in practice 92.9% of the relations annotated either a) are wholly contained in a single sentence, or b) span two adjacent sentences, with each argument contained in one of the sentences.[1]

Given this information, we can reformulate (the vast majority of) the discourse structure of a document as two intertwined sequences of discourse relations: the sequence of intra-sentence relations and the sequence of between-sentence relations (with NoRel being a legal discourse relation type in both sequences: in the intra-sentence case, it denotes a sentence without an internal explicit relation). Figure 4.1 illustrates this view of the discourse

[1]It should be noted that by the definition given in the annotation manual, all implicit relations in PDTB exist between arguments contained within two adjacent sentences.

structure of a document.

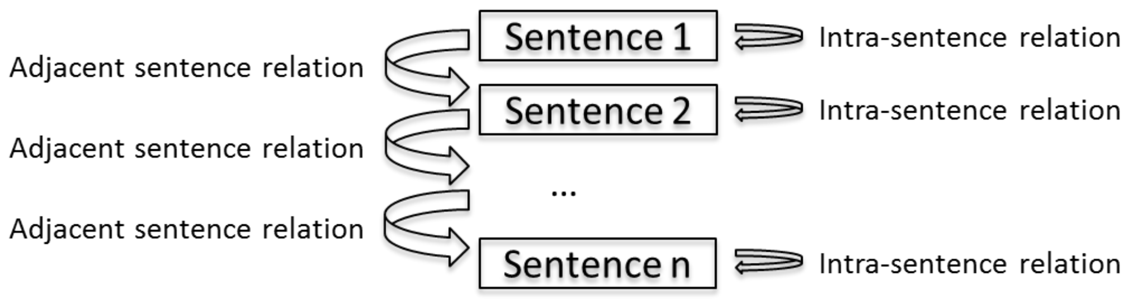


Figure 4.1: A sequential formulation of PDTB discourse structure

This view is useful because it transforms a complex structure into a simpler, linear one. It allows us to utilize a straightforward statistical model for discourse planning as described in the main part of this chapter, and to take advantage of sequential models for end-to-end discourse parsing as described in Section 4.5.

Table 4.2 shows the percentages with which each relation (using the second-level *type* of the relation) appear within sentences and across sentences in the PDTB. Clearly, many relations have a strong preference for appearing in one of the two forms.

To further motivate working with this formulation, we conducted a study of relation co-occurence in PDTB using the association ratio (Church and Hanks, 1990):

$$\alpha(x, y) = \log \frac{f(x, y)}{f(x)f(y)}$$

Where $f(x)$ is the frequency of relation x, and $f(x, y)$ is the frequency of the relation sequence x, y. The association ratio is reminiscent of mutual information, but is different in that it is ordered and non-transitive: $f(x, y) \neq f(y, x)$, and therefore $\alpha(x, y) \neq \alpha(y, x)$. We calculated the association ratio for all sequences of two relations (including their form - intra-sentence or across sentences) in the PDTB for both the first level relation *classes* and the second level relation *types*.

Table 4.3 shows all sequences of *class* level relations and their association ratio scores. Some interesting patterns emerge, and we discuss some examples below. Note that an association ratio of 0 is exactly random, while higher positive ratios mean stronger associations

Relation	# in PDTB	% intra-sentence	% across sentences
expansion.conjunction	7661	38%	62%
contingency.cause	5989	45%	55%
comparison.contrast	5367	39%	61%
entrel	5130	11%	89%
expansion.restatement	3101	28%	72%
temporal.asynchronous	2665	68%	32%
expansion.instantiation	1625	17%	83%
temporal.synchrony	1533	85%	15%
contingency.condition	1353	99%	1%
comparison.concession	1261	56%	44%
expansion.alternative	510	57%	43%
expansion.list	487	43%	57%
comparison (no *type* specified)	477	35%	65%
expansion (no *type* specified)	110	35%	65%
contingency.pragmatic_cause	71	49%	51%
contingency.pragmatic_condition	67	96%	4%
comparison.pragmatic_contrast	21	43%	57%
expansion.exception	16	69%	31%
comparison.pragmatic_concession	8	50%	50%
temporal (no *type* specified)	6	83%	17%
contingency (no *type* specified)	2	50%	50%

Table 4.2: The number of times a relation type appears in PDTB, and the percentage of time with which it appears inside a sentence and across adjacent sentences

Form	Relation 1	Relation 2	α	Form	Relation 1	Relation 2	α
across	temporal	temporal	1.624	intra	temporal	temporal	1.49
↓	comparison	contingency	1.19	↓	comparison	contingency	0.986
intra	contingency	comparison	1.124	across	contingency	comparison	0.973
	contingency	contingency	1.06		temporal	contingency	0.947
	temporal	expansion	0.93		temporal	comparison	0.935
	contingency	expansion	0.926		contingency	contingency	0.842
	contingency	temporal	0.924		expansion	comparison	0.841
	comparison	expansion	0.848		expansion	norel	0.742
	expansion	comparison	0.82		comparison	norel	0.725
	expansion	expansion	0.806		norel	expansion	0.715
	comparison	temporal	0.737		contingency	norel	0.713
	expansion	temporal	0.722		expansion	temporal	0.707
	norel	norel	0.719		temporal	expansion	0.698
	expansion	norel	0.677		norel	norel	0.692
	expansion	contingency	0.629		norel	temporal	0.664
	comparison	norel	0.626		comparison	comparison	0.662
	norel	comparison	0.618		norel	contingency	0.647
	comparison	comparison	0.616		norel	comparison	0.636
	norel	expansion	0.616		comparison	expansion	0.62
	norel	contingency	0.613		expansion	contingency	0.613
	temporal	norel	0.592		contingency	expansion	0.595
	contingency	norel	0.576		expansion	expansion	0.59
	norel	temporal	0.563		temporal	norel	0.587
	temporal	contingency	0.542		contingency	temporal	0.269
	temporal	comparison	0.371		comparison	temporal	0.233

Table 4.3: All *class* level relation sequences and their association ratio scores

and negative ratios mean anti-associations. There are no anti-associations in the *class* level sequences, presumably because of the small number of relations, but they do exist in the *type* level sequences.

Temporal relations are highly associated to themselves, in either form; expansion relations, on the other hand, are much more highly associated in the across $\rightarrow$ intra form than in the intra $\rightarrow$ across form (that is, in a cross-sentence expansion relation, the second sentence is more likely to contain an internal expansion relation than the first). Contingency and comparison are highly associated with each other in either form, while comparison is not associated with itself in either form and contingency is associated with itself more strongly in the across $\rightarrow$ intra form.

Across-sentences temporal relations (which are not as common as within-sentence ones, as seen in Table 4.2) are highly associated with sentences that are an internal expansion relation (in addition to those that have an internal temporal relation), and exhibit the weakest of all sequence associations to comparisons and contingencies. Intra-sentence temporal relations, however, are very strongly associated with cross-sentence comparisons and contingencies. This result in particular shows the value of viewing relations as we do - a sequence like *contingency-temporal* behaves in the complete opposite way in the two forms. To illustrate the difference between the forms, consider the following sentence pair:

1. The sun rises in the east.
2. Therefore, it is morning in New York before San Francisco.

These two sentences exhibit a cross-sentence contingency relation, and the second sentence contains a temporal relation. Compare that with the following sentence pair:

1. It is morning in New York.
2. It will only later be morning in San Francisco, because the sun rises in the east.

Here we have a cross-sentence temporal relation, and the second sentence contains a contingency relation. This pair is not as natural sounding (and other formulations that retain a cross-sentence temporal relation, for example placing the contingency within the first sentence or changing the connective from "because" to "therefore", will sound even less natural) and our analysis suggests that this is a general rule for discourse units that contain both a temporal and a contingency relation.

NoRel across sentences is much more associated with the first sentence having an expansion, contingency or comparison relation than a temporal relation. Sentences with internal temporal relations clearly tend to have relations with the sentences that follow them (especially temporal, contingency or comparison relations - not so much expansion relations). Sentences with intra NoRel relations (i.e., sentences that have no internal explicit relation) typically appear before sentences with which they have no relation or an expansion relation.

The results of this study are even more striking (albeit much more complex to follow because of the sheer number of possible sequences) for the more fine-grained *type* level relation sequences (and unlike the *class* level sequences, contains some anti-associations). *List* relations are, as expected, very strongly associated to one another in any form. Another related and strongly associated sequence is a cross-sentence *instantiation* with a *list* inside the second sentence; interestingly, the opposite form (intra-sentence *instantiation* and cross-sentence *list*) is strongly anti-associated. *Pragmatic_cause* across sentences is associated with *concession* or *instantiation* in the second sentence, and with temporal relations in the first sentence. Cross-sentence *concessions* are associated with *alternatives* in the second sentence and (separately) *restatements* in the first sentence (authors concede a point against an original idea and offer an alternative in the first case; in the second, they concede a point by restating it). There is also some strong anti-association in this set of sequences, notably cross-sentence *lists* with intra-sentence contingency relations and cross-sentence *synchrony* with *concessions* and *restatements*. The full list of association ratios for the *type* level relation sequences is quite long and can be found in Appendix A.

The structure shown in Figure 4.1 is very useful for our parsing approach, described at the end of this chapter, and the information in Table 4.2 and Table 4.3 motivates it and can be used in microplanning decisions. For the purpose of discourse planning, however, we want to simplify the structure further so we can work at the message level. Figure 4.2 shows a simplified version which does not include sentence boundary decisions. We will use this version as our definition of a discourse plan in this chapter.

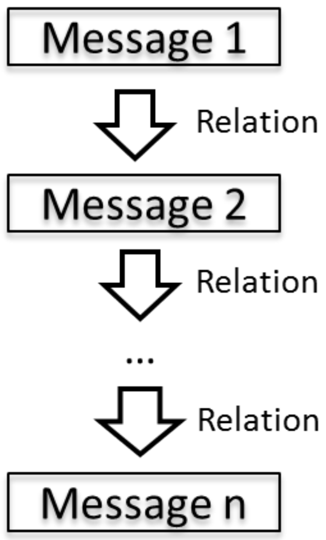


Figure 4.2: A linear discourse plan

While we lose some association information with this simplification, the relative rankings of most associations are still the same (with much smaller values), as shown in Table 4.4

4.3 Discourse Planning

One component of what makes a good discourse plan is the sequence of content: some content is more central and should appear earlier, for example; and some predicates and objects are semantically related and should appear near one another. Another component is the sequence of discourse relations, as shown in the previous section and as noted in previous studies of text coherence (Lin et al., 2011; Feng et al., 2014). Previous work on discourse planning has focused heavily on the first component, either completely ignoring the second or adding it as an afterthought. Motivated by the analysis in the previous section, we go the opposite route and define a discourse-focused sequential approach to discourse planning. In our method, we use only the sequence of relations to explicitly model the discourse plan, and expect that content coherence will be implicitly encoded in that sequence through the data structure we use to encode potential relations.

Relation 1	Relation 2	α	Relation 1	Relation 2	α
temporal	temporal	0.411	expansion	expansion	-0.003
comparison	contingency	0.285	expansion	comparison	-0.015
contingency	comparison	0.14	expansion	temporal	-0.029
norel	contingency	0.097	temporal	expansion	-0.052
norel	expansion	0.089	temporal	comparison	-0.063
comparison	norel	0.084	contingency	expansion	-0.078
contingency	contingency	0.059	comparison	expansion	-0.109
norel	temporal	0.055	norel	norel	-0.131
norel	comparison	0.054	comparison	comparison	-0.164
temporal	norel	0.049	temporal	contingency	-0.172
contingency	norel	0.049	contingency	temporal	-0.196
expansion	norel	0.046	expansion	contingency	-0.216
			comparison	temporal	-0.31

Table 4.4: Simplified version of the *class* level relation sequences, without sentence boundary decisions, and their association ratio scores

Our approach relies on a data structure called the Multigraph of Possible Relations (MPR). The idea behind this data structure is that while we do not know *a priori* the best discourse plan for a set of selected messages (i.e., the best order and set of discourse relations to impose on the messages), in concept-to-text systems we do know which relations should be *possible* between any pair of messages (and conversely, which relations would be impossible). Discourse relations can be either semantic (e.g., a real-world causality between events described in a pair of messages) or pragmatic (e.g., one of the messages describes an event that motivates including the other message in the text). Possible relations of the first type are implied in the semantics of the messages themselves, while possible relations of the second type are related to discourse intentions and therefore dependent on the application. The next section describes how we build the MPR in a simple system we created to evaluate our discourse planning approach, and Chapter 5 describes how we can build it more generally in a complex generation framework.

The MPR is a directed multigraph where each vertex corresponds to a message and each edge corresponds to a (directed) relation that can possibly exist between a pair of messages. Each edge has a type label, and multiple edges can exist in the same direction between a pair of messages if they have different types. Figure 4.3 shows a sample MPR with five messages. Note that while we do not show them in the figure, each pair of messages also has implicit *NoRel* edges going in both directions between them.

We formulate the discourse planning task as the task of finding the best Hamiltonian path through the MPR (because of the implicit *NoRel* relations, the MPR is fully connected so a Hamiltonian path always exists). Any Hamiltonian path in the MPR creates a complete discourse plan, as defined in Figure 4.2, containing all selected messages. The best path is simply the one that maximizes the likelihood of its associated discourse plan. In order to quantify the likelihood of a sequence of relations, we build an n-gram model of discourse relations from a discourse-annotated corpus.

An n-gram model measures the transitional probabilities for sequences of the units that the n-grams are composed of. In this case, the units are discourse relations. The probability of a particular sequence of relations of length $n+1$ given an existing subsequence of length n is computed as a fraction of the number of times it appears in the corpus and the number

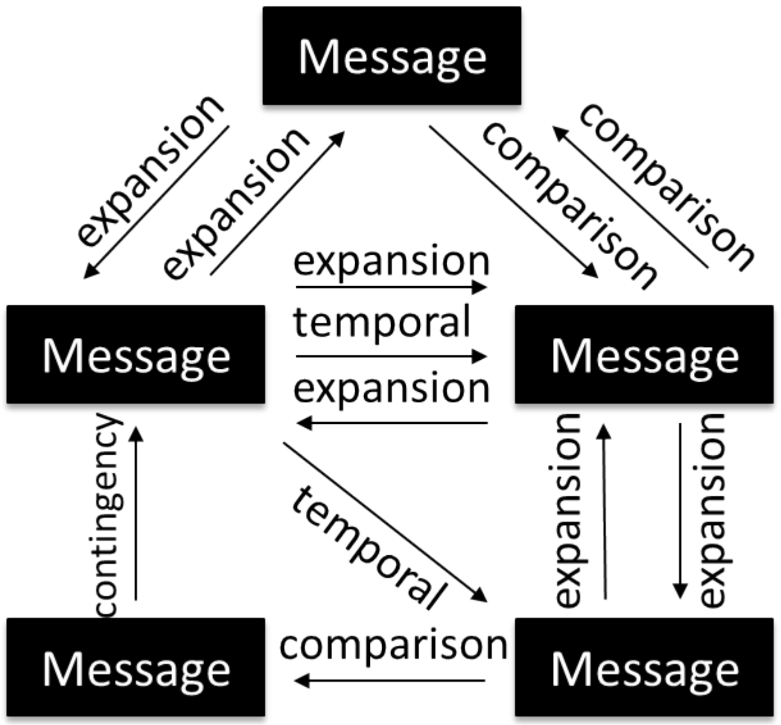


Figure 4.3: A sample Multigraph of Possible Relations

of times the subsequence appears in the corpus, i.e.

$$P(r_i|r_{i-n}, ..., r_{i-1}) = \frac{C(r_{i-n}, ..., r_{i-1}, r_i)}{C(r_{i-n}, ..., r_{i-1})}$$

Where $C(s)$ is the number of times sequence s appears in the corpus. Using this model to generate a discourse plan given a Multigraph of Possible Relations is a stochastic process: at each stage, we choose the next relation edge out of the last chosen message vertex based on the selected sequence of relation edges and the probabilities for the next relation in the model. When an edge is selected, the vertex it leads to is used and other edges leading to it can no longer be selected. The first vertex is often known in advance in real-life generation systems, as we discuss in Chapter 5; if it is not known, the process should be repeated with each possible initial vertex to determine the best sequence.

4.4 Evaluation: RDF Comparison Stories

To evaluate our discourse planning approach, we present an NLG framework that generates texts from existing semantic web ontologies using an n-gram model of discourse relations over constructed MPRs. Through a crowd-sourced human evaluation, we show that the ordering of our documents and the choice of discourse relations is significantly better when using this model, and that the genre of the corpus used to build the model makes a difference in human satisfaction.

In this section, we generate *comparison stories*, describing and comparing two similar entities, from an RDF ontology. The RDF semantic representation is commonly used in semantic web resources and free ontologies. An RDF message (called a *triple*) has three parts: a subject, a predicate and an object. For each story, we consider any triple whose subject is one of the participating entities as a potential message to be generated. We do only minimal processing on these messages: where two triples have the same subject and predicate but different objects, we merge them into a single message with multiple objects; and where two triples have the same subject and object but different predicates, we merge them into a single message with multiple predicates.

Next, we build the set of potential discourse relations between all messages. For the purposes of this study, we use the PDTB *class-level* relations, of which there are four: *expansion*, *comparison*, *contingency* and *temporal*. We do not differentiate between explicit and implicit relations, and treat *entrel* as a type of *expansion* when building the model.

Potential discourse relations are implied in the semantics of the triples: messages that contain the same predicate and object may have an expansion relation among them (e.g. "John has a ball. Mary also has a ball"). Messages that contain the same predicate but different subjects and objects may have a comparison relation (e.g. "John likes apples but Mary likes oranges").

Specific predicate pairs will also have specific potential relations among them - for example, "birth place" and "residence" have a temporal relation (when applied to the same subject). The same is true for contingency relations (e.g., "city" and "country" for the same subject - if the subject is in a city, it implies which country it is in). We manually annotated the 59 predicate pairs in the domains we evaluated that had potential temporal and contingency relations, as well as 8 pairs with special potential comparison relations (e.g., "birth place" and "residence" if the subject is the same but the object is not).

Once the potential relations are identified, we build the MPR - a directed multigraph where each vertex is a message and each edge is a potential relation.

Once the graph is ready, we perform content selection. Given a desired number of messages to generate, we choose the set of messages that maximizes the number of edges in the resulting subgraph (thus ensuring that the selected messages are discourse-coherent). If there are multiple such sets, we choose one at random.

After the content selection phase, we apply our discourse planning approach as described in the previous section (we discuss which n-gram models we use later in this section).

The next phase is microplanning. For each of the four discourse relations we use, we selected a few explicit connectives from the PDTB that are often used to convey them. We specifically chose connectives that apply to the entire range of class-level relations (e.g., for *comparison* we chose "while" - since it applies to both *contrast* and *concession* in the PDTB, but not "in contrast" which applies only to the former). We also chose only those connectives which have one of the following two structures:

1. ARG1 connective ARG2
2. ARG1. Connective, ARG2

During realization, we arbitrarily choose a connective to realize the relation.

Since the ordering and relations between messages is determined by the discourse plan, microplanning falls naturally out of it: sentence breaks occur where the connective pattern creates them, or where there is no relation between adjacent messages.

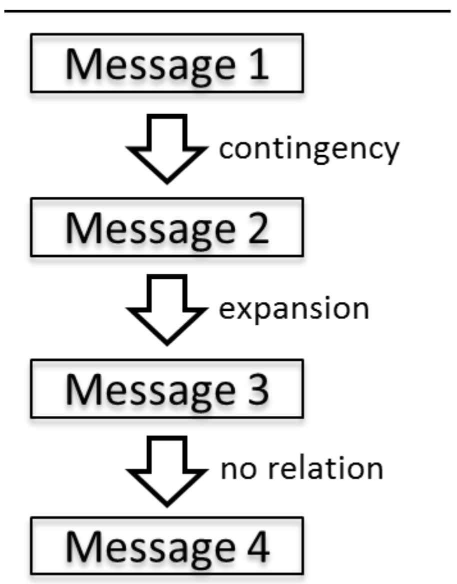
To realize the messages themselves, we follow a single pattern: "the [predicate(s)] of [subject] (is/are) [object(s)]". Simple rules are used to pluralize the predicate when there are multiple objects and to create lists of multiples objects or predicates where needed.

One method for evaluating a discourse plan independently of content is to produce pairs of generated short text documents, each containing the same content, but with different ordering and relations (as dictated by the discourse plan). The only obvious way to decide which text is better is to have human judges make that decision. It is important to minimize the effects of other qualities of the texts (differences in content, word choice, grammatical style, etc.) as much as possible, so that the judgment is based only on the differences in order and discourse. The basic formulations we described for the various stages of NLG produce texts that are rich enough to be acceptable for human readers, but which have relatively little variation in grammatical and lexical quality. This crucial combination allows us to perform a human study to specifically evaluate the discourse planning component.

We used DBPedia (Auer et al., 2007) - an RDF ontology extracted from Wikipedia - to generate content. Each document generated was a *comparison story* of two entities in a single category. The messages in the stories were selected from the set of triples where one of the entities was the subject. In order to experiment with different domains, we used four different categories:

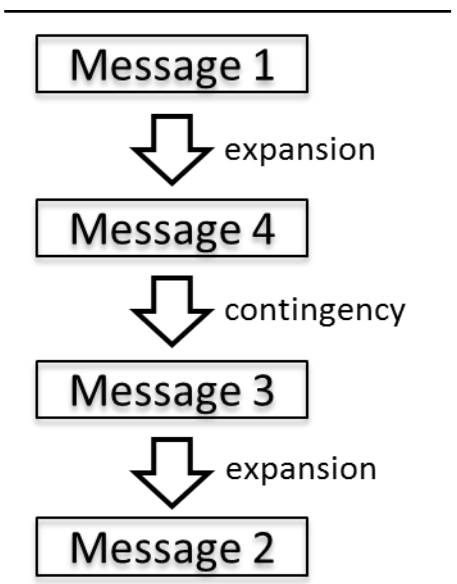
1. **Office Holder** (i.e., a person holding office such as a President or a Judge)
2. **River**
3. **Television Show**
4. **Military Unit**

The country of *Y los declaro marido y mujer* is Venezuela, so that the language of *Y los declaro marido y mujer* is Spanish. Additionally, the language of *Corazón Indomable* is Spanish. The country of *Corazón Indomable* is Mexico.



(a)

The country of *Y los declaro marido y mujer* is Venezuela. In turn, the country of *Corazón Indomable* is Mexico, so the language of *Corazón Indomable* is Spanish. In addition, the language of *Y los declaro marido y mujer* is Spanish.



(b)

Figure 4.4: Sample pair of comparison stories, for the TV shows *Y los declaro marido y mujer* and *Corazón Indomable*

The entity pairs from each category were chosen at random but were required to have at least 8 predicates and 3 objects in common, so that they were somewhat semantically related.

To ensure that human judges can easily tell the differences between the stories on a sentential level, we limited the size of each story to 4 messages. For each pair of stories, everything but the discourse plan (i.e. the content selection, the realization of messages and the lexical choice of connectives) was identical. Figure 4.4 shows an example pair of stories, and their discourse plans, from the TV Show category.

While it is not always clear on first glance which of these comparison stories is better, they exhibit differences in the order of the messages and in the sequence of relations. Based on our association ratio study and the n-gram models we create (as described below), we expect story (b) to be preferable to human annotators, which turns out to be true in this case (and as we show in our results, the model-generated plans are preferable to human annotators in most cases).

We conducted two crowd sourced experiments on the CrowdFlower platform. Each question consisted of two short stories that are completely identical in content, but each generated with a different discourse planner. The human judge was asked to decide which of the stories has a better flow (or whether they are equally good), and then to give each of the stories a score from 1 to 5, paying specific attention to the ordering of the prepositions and the relations between them. The stories were presented in a random order and were not given labels, beyond *Text 1* and *Text 2*, to avoid bias. We generated 125 pairs of stories from each category - a total of 500 - for each experiment.

Each question was presented to three judges. In each experiment, there was complete disagreement among the three annotators in approximately 15% of the questions, and those were discarded. In approximately 20% there was complete agreement, and in the rest of the questions there were two judges who agreed and one who disagreed. We also computed inter-annotator agreement using Cohen's Kappa for 217 pairs of judges who both answered at least 10 of the same questions. The average kappa value was 0.5, suggesting reasonable agreement.

In the first experiment, we compared stories generated by a planner using an n-gram model extracted from the PDTB with stories generated by a baseline planner, where all edges have identical probabilities. The results are shown in Table 4.5.

In the second experiment, we used a PDTB shallow discourse parser we developed, and which is described in the next section, to create a discourse-annotated version of the English Wikipedia. We then compared stories generated by a planner using an n-gram model extracted from the parsed Wikipedia corpus with those generated by a planner using the PDTB model. The results are shown in Table 4.6. The differences in total results in both tables are statistically significant ($p < 0.05$).

	Quality comparison			Avg. score	
	Base	Equal	PDTB	Base	PDTB
Of. Holder	27.4%	30.2%	**42.5%**	3.67	**3.76**
TV Show	34.3%	25.7%	**40%**	3.79	**3.8**
Mil. Unit	32.3%	23.2%	**44.4%**	3.69	**3.84**
River	**39.2%**	23.5%	37.3%	3.71	**3.72**
Total	34%	25%	**41%**	3.72	**3.78**

Table 4.5: Results for the comparison between the PDTB n-gram model and the baseline

	Quality comparison			Avg. score	
	PDTB	Equal	Wiki	PDTB	Wiki
Of. Holder	33.6%	14.5%	**51.8%**	3.51	**3.65**
TV Show	43.2%	8.1%	**48.6%**	3.62	**3.65**
Mil. Unit	40.4%	14.4%	**45.2%**	3.65	**3.67**
River	41.1%	11.2%	**47.7%**	3.68	**3.7**
Total	39.6%	12%	**48.4%**	3.61	**3.67**

Table 4.6: Results for the comparison between the Wikipedia model and the PDTB model

The results in Table 4.5 show that the judges preferred the stories created by the n-gram model-based planner to those created by the baseline planner, both in terms of the three-way decision and in terms of the numeric score. This is true for the total set as well as every specific topic, except for *River*. This may be because the predicates in the River category are much more cohesive than in other categories: virtually all predicates related to rivers describe an aspect of the location of the river. That fact may make it easier for a random planner to produce a story that seems coherent. Note, however, that while the judges preferred the baseline story more often in the *River* questions, the average score is higher for the model, which suggests that when the baseline was better it was only mildly so, while when the model was better is was significantly so.

The results in Table 4.6 show that the Wikipedia-based model produces better results than the PDTB-based model. We hypothesize that it is for two reasons. First, Wikipedia contains definitional texts and is closer in style and content to the stories we produce than the PDTB, which contains WSJ articles. Temporal relations constitute about 10% of both corpora, but contingency and comparison relations each make up almost 20% of the PDTB, while in Wikipedia they span only 10% and 12% of the corpus, respectively, making the share of expansion relations much larger. Second, since the PDTB is small, higher-order n-grams are sparsely found, which can add noise to the model. The Wikipedia corpus is significantly larger and does not suffer from this problem. On the other hand, of course, the Wikipedia corpus contains noise introduced by the automatic discourse parser.

The differences in average scores seen in the experiments are relatively small (but significant). That is expected, since we have eliminated the content coherence factor, which is known to be important. In addition, while judges were specifically asked to focus on the order of messages and relations between them, there is inevitably some noise due to accidental lexical or syntactic mismatches, ordering that is awkward content-wise, and other side-effects of the generation framework we employed.

4.5 Discourse Parsing

In order to extract the statistical discourse model described in the previous section from a plain text corpus, we must first automatically annotate the corpus with sequential discourse structure of the type we described in Section 4.2. In this section, we describe our shallow discourse parser, also described in (Biran and McKeown, 2015b). It utilizes a sequential model and produces a sequence of intra-sentence and adjacent sentence relations (in addition to other information, such as argument locations and connectives for explicit relation), in accordance with the structure shown in Figure 4.1, and is particularly useful for learning n-gram models of discourse of the kind we use for discourse planning in this chapter. The parser described in this section is publicly available at `www.cs.columbia.edu/~orb`.

Discourse structure is an important part of what makes a text coherent. Discourse parsing is the task of automatically determining the discourse structure of a text according to a particular theory of discourse (in our case, the PDTB). The ability to parse an entire document is crucial for understanding its linguistic structure and the intentions of its authors.

Discourse parsing is a difficult task. While some discourse relations have explicit lexical cues called discourse *connectives* or *markers*, such as "because" and "but", these are often ambiguous: they may apply to more than one relation category, or they may be used in a way that has nothing to do with discourse at all. In addition, many relations are not marked by connectives in text, and disambiguating these *implicit* relations is difficult even when it is known a relation exists. Adding to the difficulty is the fact that the arguments of the relation (there are usually two, although some frameworks allow more for certain relations) do not necessarily correspond to sentences or clauses, and may not even be contiguous under some theories.

Over the years, multiple theories of discourse have been proposed. Unlike RST and other frameworks, the discourse structure of PDTB is not fully hierarchical, so that documents in general do not have a tree-like structure. Instead, it has a predicate-argument structure, which in practice is very local (the arguments overwhelmingly appear near each other, as explained in Section 4.2). This is a crucial detail which allows our method to work on PDTB documents.

While there has been much work recently on disambiguating discourse relations in the PDTB, most have not been full parsing systems. That is, they operate in an experimental environment where some information is given (for example, some systems disambiguate only implicit relations, where it is assumed that the arguments of the relation have been identified and that the relation is known to be implicit (Pitler and Nenkova, 2009; Park and Cardie, 2012)). Full systems, in contrast, operate on unannotated text documents producing the full discourse structure of the text, including both implicit and explicit relations, and so can be realistically used in NLP applications. Although not strictly parsing in the case of PDTB, such systems perform what has been called the *end-to-end discourse parsing* task. Interest in full discourse parsing in the PDTB has been increasing, and it has been featured as a CoNLL shared task.

The only published work, to our knowledge, which provides end-to-end PDTB discourse parsing is (Lin et al., 2014); they use a four-stage architecture where each stage carries out one subtask in identifying discourse relations (e.g., explicit or implicit). The parser is evaluated in terms of *exact match* and *partial match.* Unlike exact match results, which are considered correct only if both the relation type and the exact span of its arguments are identified correctly, partial match results are correct as long as the relation type is correctly identified and each proposed argument shares at least one noun and verb with the true argument. We believe that partial match results are best to focus on at this point in time, since current performance on exact match results is too low to be useful. Many current NLP applications (such as summarization and question answering) focus on sentences or clauses anyway and would find this formulation natural.

In this section, we present a simple yet powerful sequential approach to PDTB discourse parsing, utilizing two CRFs and features that are designed to discriminate both explicit and implicit relations. We surpass state-of-the-art performance with a simpler structure, less hand-crafted rules for special scenarios and with an approach that makes adding new features extremely easy.

4.5.1 Related Work

Early data-driven work on discourse parsing has focused on frameworks such as RST, using the small RST Discourse Treebank (Carlson et al., 2001). Marcu (1997) and later Soricut and Marcu (2003) developed methods for parsing documents into the RST discourse representation. There has also been more recent work on end-to-end RST-style parsing (LeThanh et al., 2004; duVerle and Prendinger, 2009).

Recently, there has been more focus on the PDTB (Prasad et al., 2008), the largest annotated discourse corpus currently in existence. Most work so far has focused on solving specific subtasks of the overall parsing task. Pitler and Nenkova (2009) focused on explicit relations and found that they are relatively easy to disambiguate using syntactic features. Wellner (2009) used both lexical and syntactic features to identify the arguments of a relation. Identifying and disambiguating implicit relations has been the hardest task to achieve good performance at, and is an active area of research. Pitler et al. (2009) were the first to identify implicit relations in the PDTB in a realistic setting, and later work has improved on their methods as well as introducing new ideas (Lin et al., 2009; Zhou et al., 2010; Park and Cardie, 2012; Biran and McKeown, 2013a; Li and Nenkova, 2014a).

Most recently, Lin et al. (2014) have introduced and evaluated the first system which provides end-to-end discourse parsing over PDTB (the *Lin parser*). In their work, they have combined much of the earlier work on specific subtasks, utilizing a connective disambiguation component and an explicit relation disambiguation component inspired by Pitler and Nenkova (2009)'s method, as well as an implicit relation disambiguation component descending from their own previous work (Lin et al., 2009). Their approach is to decipher the document in a structured way, in four steps: first, identify explicit discourse connectives; second, identify the text spans of the arguments (in PDTB, there are always two arguments, arg1 and arg2) corresponding to the connective; third, identify the type of relation between the arguments (the third step completes the subtask of finding *explicit relations*); and fourth, for every adjacent pair of sentences, identify which type of *implicit relation* - relations where there is no connective - exists between them (or, if none does, identify the relation as EntRel - meaning the sentences share an entity but not a relation, or NoRel - meaning they share nothing at all).[2]

While the structured approach of the Lin parser has many advantages in that it attempts to solve the sub-tasks of discourse parsing in an organized, intuitive way, it has some disadvantages. One is that because of the pipeline structure, errors propagate from step to step. For example, if a (truly) implicit relation was incorrectly identified as an explicit relation because of a false connective, the features used by the implicit relation identifier that may correctly discriminate its type will not get a chance to be used. If argument spans are incorrectly identified, the explicit relation disambiguator will be handicapped since it mostly employs features based on the lexical and syntactic structure of the arguments.

Another disadvantage is the fact that in the structured approach, potential relations are considered individually, although adjacent relations can intuitively be indicators of the relation type.

Finally, building such a system requires significant design and engineering, and making changes that are not localized to a specific component can be difficult and time-consuming. At this point in time, when work on discourse parsing in PDTB is at its early stage, a more flexible and easily extensible approach would be beneficial to the community.

4.5.2 Method

As described in Section 4.2, PDTB discourse relations can be seen as a triple: relation type, argument 1 and argument 2. In principle, the two arguments of a discourse relation can be located anywhere, but in practice almost 93% of the relations annotated are either intra-sentence relations (both arguments completely contained within one sentence) or adjacent sentence relations (each argument completely contained in one of two adjacent sentences).

Given this information, and the understanding that the sequence of discourse relations can be useful for determining the type of a relation, we reformulate the task of parsing the PDTB discourse relations as the combination of two tagging tasks. For each document, we separately tag the sequence of sentences for intra-sentence relations, and the sequence

[2]There is also a fifth step, identifying spans that attribute a statement to a source, e.g. "B.P. explains that ...". Attribution span detection is a secondary task which is evaluated separately from the main discourse structure pipeline, and we are not concerned with it here.

of adjacent sentence pairs for cross-sentence relations. While intra-sentence relations are always explicit, adjacent sentence relations may be explicit, implicit, or fall into the PDTB's AltLex or EntRel categories. Unlike previous work, we use a single method to disambiguate all adjacent sentence relations. We call this approach to discourse parsing the *Two Taggers* approach.

As a result, we have a sequence of sentences, each tagged with the relation that exists within it and each adjacent pair tagged with the relation that exists between them (i.e., the structure in Figure 4.1). In order to transform this structure to a full discourse parse, we must also identify the arguments and their spans. Since our goal is a simpler system and our focus is on partial match results, we avoid using a complicated syntactic rule system for each possible scenario in favor of a few simple rules. For adjacent sentence relations, we mark arg1 as being the entire first sentence and arg2 as being the entire second sentence (under partial match, this turns out to be correct in all but 0.002% of relations in the training set). For single-sentence relations, we distinguish among two cases: if the first word of the sentence is an intra-sentence initial connective[3] then we identify arg2 from the beginning of the sentence until the end of the first VP, and arg1 from there to the end of the sentence. Otherwise we identify arg1 from the beginning of the sentence to the middle connective (if there are more than one) and arg2 from there to the end of the sentence. While this approach ignores many complexities of the true argument structure of PDTB (for example, arguments may be nested, and a sentence may include text that is not inside an argument), it works well for partial match. In fact, as we show in our evaluation, it is also not too far behind the state of the art on a slightly more lenient version of exact match. We use Pitler and Nenkova (2009)'s high performing connective classifier (F1 above 95) to distinguish discourse connectives from their non-discourse counterparts.

The PDTB relation categories are hierarchical, and we are interested in finding the *type*, or second-level categories, of which there are 16 (plus EntRel and NoRel, for a total of 18). The first level (the *class*, of which there are 4) is too coarse to be useful for many applications, and the third level (the *subtype*, of which there are 25) is too fine-grained and

[3] *After, although, as, because, before, except, if, since, though, unless, until, when, whereas,* and *while* (as well as variations such as *if and when*).

difficult to disambiguate. Table 4.1 shows the hierarchy of 4 classes and 16 types. The Lin parser also deals with type-level categories, but almost all other previous work has focused on the significantly easier class-level categories.

Treating discourse parsing as a tagging problem has many advantages. Tagging tasks have been widely explored in NLP and there are many off-the-shelf tools and methods for tackling them. Many generic taggers that can be applied to this task with minimal effort are available to researchers, while generic parsers do no exist. Tagging is a simpler and often more tractable task than parsing, and it can be done using sequential classifiers, which are both fast and powerful.

There are also some limitations to the tagging approach. As mentioned earlier, some rare relations span more than two sentences, or sentences that are not adjacent. In addition, there are (also rare) situations where there are multiple relations in a single sentence, and with our approach we can at most tag one correctly. Because of these two limitations, we have an upper bound on F-measure performance of 89.4 in the PDTB corpus. Since current state-of-the-art performance is far below this level, we do not view this as an urgent problem. At any rate, additional specialized approaches can be added to correctly handle those rare cases.

We use Conditional Random Fields (CRFs) to implement both taggers. CRFs were first introduced by Lafferty et al. (2001) and have been successfully used for many NLP tagging tasks such as named entity recognition (McCallum and Li, 2003) and shallow parsing (Sha and Pereira, 2003). We use simple linear-chain CRFs for both taggers. In the linear-chain CRF model, the posterior probabilities for an ordered sequence input $\mathbf{x} = \{x_1, \ldots, x_{|x|}\}$ of tag labels $\mathbf{y} = \{y_1, \ldots, y_{|x|}\}$ are defined as

$$P(\mathbf{y}|\mathbf{x}) \propto \prod_{i=1}^{|x|} exp\left(\sum_{k=1}^{K} \theta_k \Phi_k(y_{i-1}, \mathbf{x})\right)$$

where θ_k are weights corresponding to the features Φ_k. The feature values at index i of the sequence may be computed based on the previous tag in the sequence y_{i-1} and the entire sequence $\mathbf{x}$. The weights θ_k are estimated using gradient descent to maximize the likelihood of the input.

In our formulation, each $\mathbf{x}$ is a PDTB document, consisting of a sequence of sentences (for the intra-sentence relation tagger) or a sequence of sentence pairs (for the adjacent sentence relation tagger). $\mathbf{y}$ consists of all type-level discourse relation categories.

In our experiments, we used a maximum likelihood prior and limited the gradient descent to a maximum of 200 epochs instead of waiting for it to converge.

While CRFs have been used in the past for subtasks of RST discourse parsing (Feng and Hirst, 2014) and for finding the arguments of explicit relations in PDTB (Ghosh et al., 2011), no sequential approaches have ever been used in a way that models the sequential dependency between PDTB relations. Previous work (Pitler et al., 2009; Zhou et al., 2010) has utilized features that consider adjacent lexical information in relation type classification, but true sequential or joint classifications have not been attempted.

4.5.2.1 Features

The intra-sentence tagger deals only with explicit relations, and as such focuses on features related to discourse connectives. We use Pitler and Nenkova (2009)'s connective classifier to identify discourse connectives within the sentence, and for each connective generate the set of binary features shown in Table 4.7, all of which are features used in explicit relation detection by Pitler and Nenkova (2009) or by Lin et al. (2014).

Connective	Connective's syntactic category
Previous word + connective	Parent's category
Connective + next word	Left sibling's category
Path to root	Right sibling's category
Compressed path to root	

Table 4.7: Binary features used in the intra-sentence tagger.

The adjacent sentence tagger utilizes a larger variety of features, designed to disambiguate relations across sentences that may be explicit, implicit, AltLex or EntRel.

We divide the adjacent tagger's features into four thematic types: lexical, connective-related, syntactic and structural features. The full list of features is shown in Table 4.8, and we describe the non-obvious ones below.

Lexical	Connective	Syntactic	Structural
Unigrams	Connective	Production Angles	Paragraph Split
Word Pair Similarity	Prev. Word + Connective		Short Document
Word Pair Similarity Avg.			
Document Centrality			
Expanded Shared Words			

Table 4.8: Features used in the adjacent sentence tagger.

Connective features are created for any connective found in each sentence separately using Pitler and Nenkova (2009)'s connective classifier.

The structural *short document* binary feature encodes whether or not the document has 3 sentences or less. The intuition here is that short documents are much less likely to have certain relation types (e.g., argumentative ones).

Syntactic features are derived from the parse tree of the sentence. We use the Stanford Parser (Klein and Manning, 2003) to derive the trees. Lin et al. (2009) introduced the *production rule* features, which are some of the strongest for implicit relation disambiguation. Production rules are all parent-children relations in the constituent parse of a sentence, e.g. [VP $\rightarrow$ NP PP NP]. The binary feature formulation includes the existence of each rule in arg1, in arg2, and in both. Li and Nenkova (2014b) hypothesized that production rules are too sparse, and found that using their *production stick* features achieved higher performance. Unlike a production rule, which relates to all children of a parent, a production stick is a parent-single child relation. We experimented with both feature sets, and found that we achieve the best performance with a novel middle-ground formulation. **Production angles** are a family of features indicating the appearance of syntactic triples: a parent and two adjacent children. In cases where a parent has only one child, as in the lexical leaf nodes of the tree, we produce a stick-like feature (e.g. [NP $\rightarrow$ resources]. The triples are formed using the label of each node and the descendant directionality. For example, VP $\leftarrow$ VP $\rightarrow$ NP is a parent VP with adjacent children, VP and NP. We use features for angles in each sentence separately, as well as for angles that are shared by both.

Centrality in document is a simplistic form of topic similarity: the cosine similarity of the sentence pair to the document as a whole. The intuition is that certain relations (e.g., argumentative relations such as causality and concession) would tend to be more common around the main topic of the document.

We include features for words that are shared by both sentences called **expanded shared words** - expanded because we use WordNet (Fellbaum, 1998) to expand the usual list of words in each sentence with all synonyms and immediate hypernyms of each word's most frequent sense.

The **word pair similarity features** is the set of aggregated features described in Biran and McKeown (2013a), which utilize sets of word pairs that were mined from unannotated corpora around each discourse connective. The word pair scores within the set are given by TF*IDF and treated as a vector. The feature value is the cosine similarity of the connective's vector to the vector of word pairs extracted from the pair of adjacent sentences, where each pair contains one word from each sentence. It models the similarity of the sentence pair to a sentence where the connective is used directly, and is intended to help in identifying implicit relations. In contrast to previous formulations of word pair features (Marcu and Echihabi, 2002; Blair-Goldensohn et al., 2007; Pitler and Nenkova, 2009), ours is a dense set of features which is less prone to the lexical sparsity of the relatively small PDTB. The **word pair similarity average for connective pair** is a variant where we get the similarities of the adjacent sentence pair to the word pair sets of a couple of connectives (we use every possible combination of two connectives) and use the average as the feature value. The idea is that if two connectives are related to the same relation type, a high average similarity to both may be a stronger indicator for that relation.

In addition to the features described above, the CRFs utilize sequential features in both the intra-sentence tagger and the adjacent sentence tagger. Sequential features are the *transitional* features that consider the previous tag in the sequence. The same sequential features are used in both taggers.

We use two basic pieces of information from the previous tag: the **previous tag type** is the type (second-level relation category) of the previous tag, while the **previous tag class** is the class (first-level relation category) of the previous tag.

4.5.3 Evaluation

Following Lin et al. (2014) and other previous work, we use sections 2-21 of the PDTB as the training set, section 22 as the development set, and section 23 as the test set. Since we use an automatic parser for our syntactic features, our results are equivalent to Lin et al.'s "Partial, Auto + EP" overall results for partial match, and to their "Exact, Auto + EP" results for exact match. We consider the results using gold standard parses to be less important for an end-to-end system, the main function of which is an out of the box document parsing tool. The evaluation metric in all experiments, following Lin et al., is the micro-averaged F1 score.

We show our final partial match results on the test set in Table 4.9, compared with the Lin Parser performance. We also compare our approach with the results achieved by using the exact same formulation and features (other than the sequential features, of course) in two Logistic Regression classifiers, to show that the sequential approach is in fact helpful. To illustrate the effect of our simplistic argument span identification rules, we also show results without span matching, where argument spans are presumed to always partially match if the sentence/sentences and relation type are correctly identified.

	Prec.	Recall	F1
Two classifiers	46.12	31.68	37.56
Lin Parser			38.18
Two Taggers	48.52	33.06	**39.33**
No span matching	48.72	33.32	39.57
Upper bound	100	80.82	89.40

Table 4.9: Partial match results on all relations in the PDTB. The Lin parser paper does not report precision and recall

The results of each tagger individually are shown in Table 4.10. Note that the overall results are compared against all true relations in the document, including those that our method inherently cannot identify (hence the upper bound), while the individual tagger results are only in the context of the individual tagging task. This is why the recall of the end-to-end results is smaller than the recall of either of the individual taggers.

	Prec.	Recall	F1
Intra-sent. tagger	66.36	49.82	**56.91**
Intra-sent. classifier	66.19	48.77	56.16
Adj. sent. tagger	40.31	36.53	**38.33**
Adj. sent. classifier	37.13	34.21	35.61

Table 4.10: Results for each of the two taggers separately

While we are focused on partial match results, we also show exact match results in Table 4.11. In error analysis we noticed that many of our errors on exact match arise because we include in the span another discourse connective, or an initial word like "Eventually" or "Admittedly" in a non-discourse usage. We therefore include another set of results we call "almost-exact match" which allows a match if there is at most one word at the beginning or the end of the span that does not match. Using this less strict definition, we reach a performance that comes close to the Lin parser exact match results.

	Prec.	Recall	F1
2T exact match	14.47	5.93	8.41
2T almost-exact match	29.61	14.75	19.69
Lin Parser			20.64

Table 4.11: Exact match results on all relations in the PDTB. The Lin parser paper does not report precision and recall

To emphasize how much harder it is to identify the more fine-grained level 2 relation types than it is to identify the coarser level 1 classes, we also provide results on the class-level discourse parsing task in Table 4.12.

	Prec.	Recall	F1
Two Taggers	62.56	44.3	**51.87**
Upper bound	100	80.82	89.40

Table 4.12: Results for the same task when using the level 1 classes instead of the level 2 type relation categories

4.5.4 Discussion

As seen in Table 4.9, we achieve higher performance than the Lin parser on partial match results. This is despite the fact that we use fewer manually-crafted rules and do not rely on a complex argument span identification component. Moreover, the two taggers are clearly stronger than two classifiers with identical features, especially for the adjacent sentence task, which shows that there is value to the sequential approach.

It is clear from Table 4.10 that identifying relations in adjacent sentence pairs is a more difficult task than identifying them inside a single sentence. This makes sense because single sentence relations are always explicit in the PDTB while most adjacent sentence relations are implicit. It is well established that implicit relations are much harder to disambiguate than explicit ones. While we cannot provide an evaluation for implicit relations only - it is not clear how to fairly define false positives since we tag the entire document without differentiating between explicit and implicit relations - we can provide a lower bound for our performance by using only implicit relations to collect the true positives and false negatives, and all tagged relations to collect false positives.

Our lower bound F-measure for implicit relations is 28.32.[4] In the Lin parser, the F-measure performance of the implicit relation classifier is 25.46, while the explicit relation classifier has an F-measure over 80. These numbers imply that our method is especially advantageous for implicit relations, while explicit relations may be harder to disambiguate without the specialized argument location/span identification step taken by the Lin parser. In addition, the relations that our approach inherently cannot handle are all explicit.

It is interesting to note that the difference between the taggers and the classifiers is

[4]Precision is 28.02 and recall is 28.63.

much larger for the adjacent sentence pairs, meaning that the sequential features are very strong in the adjacent sentences tagger. This may indicate that intra-sentence relations are more "stand-alone" in nature while inter-sentence relations are more connected with the rest of the document. This result, and the fact that our performance on intra-sentence relations are not as high as previous results on explicit relations, suggest that one promising path for future work is the combination of a more structured intra-sentence explicit relation approach (one that would, among other advantages, allow finding multiple relations within the same sentence) with a sequential adjacent-sentence approach. Our performance suggests that this separation (intra-sentence and adjacent sentence) in methodology, which allows a sequential view, may in some cases be more useful than the traditional explicit vs. implicit separation.

Our approach beats state-of-the-art performance using partial match, which is the natural evaluation to use at this point in time given exact match performance (this view has been expressed by Lin et al. (2014) as well). While we do not achieve the same results on exact match, which is to be expected given our very simple approach to argument span identification, Table 4.11 shows that we come very close if a slightly less restrictive evaluation is used. This reaffirms the conclusion that exact match is a very difficult task: even with complex hand-crafted syntactic rules, correctly identified spans are relatively simple cases which can also be identified (if a single word error is allowed) by a much simpler method.

Table 4.12 illustrates how much harder the type-level parsing task is than the class-level parsing task. In our discourse planning experiments earlier in this chapter, we used our parser in the class-level mode to create a more accurate n-gram model from a discourse parse of Wikipedia.

4.6 Conclusion and Future Work

In this chapter we introduced an approach to discourse planning that relies on a potential discourse multigraph, allowing for an n-gram model of relations to drive the discourse plan and efficiently determine both the ordering and the relations between messages. In contrast to previous work on content planning (which used manually-created global models) and coherence modeling (which use learned content-based local models), we utilize a learned *local discourse coherence model.* This model can be learned from a text corpus annotated with shallow (local) discourse relation, and we presented an end-to-end discourse parser that is specifically suited to creating such annotated documents.

To motivate our method, we conducted an association ratio study of discourse relation pairs and showed that there is useful coherence information in local discourse sequences. To evaluate our method, we conducted two experiments, comparing stories generated with different discourse planners. The first shows that an n-gram model-based planner significantly outperforms the random baseline. The second suggests that using an n-gram model derived from a corpus that is larger and closer in style and content, though less accurately annotated, can further improve results.

In the generation system we used for our evaluations, as well as for the generation framework described in Chapter 5, entity-based local coherence is partially encoded in the entity-based way we build the MPR. In Chapter 5 we also supplement it with a more direct, external measure of entity coherence. In future work, it would be interesting to explicitly model it and combine it with our discourse-based view of coherence to create a unified statistical discourse planner. It would also be interesting to explore additional stochastic models of discourse that look at other, non-sequential collocational information.

Part II

Generation

Chapter 5

Generation Framework

In this part of the thesis we are concerned with a family of generation applications we call Generation Endeavors with Modular Subjects, or GEMS. As its name suggests, this family consists of generation applications which can each be implemented with a range of subject domains. This chapter describes a general framework for setting up GEMS applications - i.e., creating generation systems capable of handling a GEMS application for multiple domains. The framework requires limited manual work to be done only once for each application; adapting the application to new subject domains can then be done automatically, requiring only a domain corpus.

For example, consider the GEMS application "product description": we have a database of products, each with a set of features and specifications, and we want to automatically describe them. The products belong to different types: books, cameras, cars, insurance packages, etc. In some sense, all product descriptions are the same. They'll mention the important features of the product, what differentiates it from other similar products, and perhaps some background to help the consumer decide what is important. They may also mention the price of the product in comparison to others, ratings that the product received from various rating agencies, and why these agencies are good. In another sense, however, descriptions of cameras will be very different from descriptions of insurance packages. In particular, when it comes to explaining the product features and giving some background about it - anything beyond the simple names and numbers that exist in the database - we will want to see very different text for each product type. For cameras, we should talk about lenses and shutters (not only what the particular camera has, but relevant background knowledge as well: why is one lens better than another? what are the advantages of a mechanical shutter?) while for insurance packages we should talk about coverage and benefits (why and where is flood insurance important?). “Product description” is the application in this example, while “cameras” and “insurance” are two of many possible subject domains. Table 5.1 lists several example GEMS applications, showing their global (domain-independent) themes and a few sample subject domains.

Application	Description	Global Themes	Sample Domains
Product description	Generate a description for a new product	Product feature; Seller; Price; Rating agency	Books; Cameras; Cars; Insurance Packages
Biography	Generate the biography of an accomplished person	Experience; Achievement; Award; Colleague	Jazz Musicians; Computer Scientists; Politicians; Military Officers
Scientific Paper Explanation	Generate an explanation (for a layperson) of a scientific paper	Problem; Approach; Related approach; Result	NLP; Molecular Biology; Solid-state Physics
Company description	Generate a description for a company	Executive; Founder; Product	Computer Hardware; Oil and Gas; Financial Investment; Retail
MMORPG Character Description	Generate a description of a character in an online multiplayer RPG	Race; Class; Profession; Relationship	Different game worlds
Machine Learning prediction justification	Generate a justification for the prediction of a Machine Learning model	Prediction; Model; Feature	Movie recommendation; Stock price prediction; Medical diagnosis

Table 5.1: Sample GEMS

The reason our framework is able to adapt to new subject domains automatically is that it relies on hybrid concept-to-text and text-to-text generation: part of the generated text consists of C2T messages (messages that are created from structured data according to a recipe or algorithm that is unique to the application), while another part comes from T2T messages (messages that are extracted from a corpus, although we use our work on paraphrase mining in Chapter 2 to modify and add variation to those extracted messages). The special structure of the GEMS family allows us to define the recipes required for C2T generation, and the abstract recipes for automatically extracting the information required for T2T generation from a new subject domain corpus, only once per application. In addition, we use our data-driven methods from Chapters 2-4 to automatically extract paraphrases, discourse models and a taxonomy from the domain corpus, which further refine the ways in which text is generated differently for each subject domain.

Figure 5.1 shows a high level view of the way the framework operates, and is also an overview of the structure of this chapter beyond the next two sections which deal with definitions and a more detailed overview:

- Part 1 (on top, in red) is the abstract definition of a generation system for a GEMS application, and is described in Section 5.3. This is the only part which requires manual work - specifically, the creation of a few algorithms - and occurs once per application (e.g. once for the product description application).

- Part 2 (in the middle, in blue) is the adaptation of the application to a particular subject domain, which results in a domain-specific generation system. It includes, for example, extracting paraphrasal templates, a discourse model and a language model. This happens once per domain (e.g., once for cameras, once for cars, etc) and is completely automatic, except that a subject domain corpus must be provided. This part is described in Section 5.4.

- Part 3 (on the bottom, in green) describes the process of generating text given instance data in a domain-specific generation system. This part is the one using the algorithms defined in part 1 and the models/data extracted in part 2. This part is described in Section 5.5.

The gray part of the figure (data-driven bottleneck solutions) are the methods described earlier in this thesis, in Chapters 2- 4.

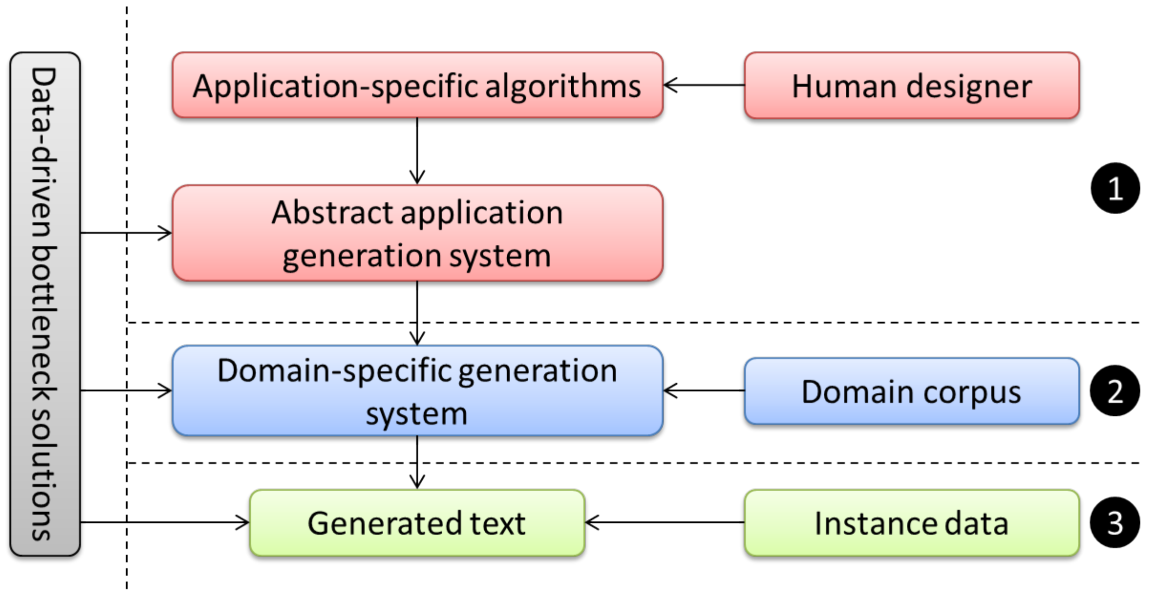

Data-driven bottleneck solutions
Application-specific algorithms
Human designer
Abstract application generation system
1
Domain-specific generation system
Domain corpus
2
Generated text
Instance data
3

Figure 5.1: High level overview on the framework.

The main contributions of this framework are the hybrid C2T-T2T generation approach - a new idea that has only recently started being explored, e.g. by Saldanha et al. (2016); and the elimination of manual work when adapting to new domains. The framework introduces a powerful tool for defining abstract generation systems that can be used in any domain for which a simple text corpus can be found.

In chapter 6, we demonstrate and evaluate our framework through two of the more straightforward applications on the list - biographies and company descriptions. In chapter 7, we demonstrate and evaluate it for the last application on this list - generating justifications for Machine Learning predictions - a more specialized application for which there are currently no other generation systems as far as we know.

5.1 Definitions

This section provides definitions for the central terms and concepts used in this chapter.

5.1.1 The Three Users

Throughout this chapter, we will refer to three types of users associated with this framework, with varying levels of required expertise.

The application designer is an NLG researcher or engineer who adopts the framework for a new GEMS application (without worrying about which subject domains it will be applied to). This person must be a programmer, have NLG knowledge, and be familiar with the framework described in this chapter. She should be familiar with the application in general, but does not need to be an expert in any of the possible subject domains to which the application may apply.

The subject domain designer is someone who adopts an existing GEMS application for a new subject domain. The technical requirements from this person may change from one GEMS application to the next (depending on the sort of information expected from an instance in the application), but he does not need to have any knowledge of NLG or NLP, and may not need to be a programmer. In the RDF applications presented in Chapter 6, for example, the only requirement of this person is to be able to execute command line programs and point them to a list of entities, so it can be anyone with a minimal technical ability (of course, a user interface can always be built by the application designer to alleviate even this requirement). In the prediction justification application of Chapter 7, in contrast, the subject domain designer (who adapts the application for justifying the predictions of a particular classifier) needs to be able to provide the feature values and other information about each prediction, either programmatically or via XML.

The end user is the person for whom the text is generated. This person only needs to be literate (in English) and be capable of understanding text about the particular subject domain. For some subject domains (e.g., a medical condition prediction domain in the prediction justification application presented in Chapter 7) this user should be an expert in the subject domain (e.g., a medical doctor).

A major theme of the framework we describe in this chapter is making the job of the subject domain designer, in particular, extremely easy. Once a GEMS application is defined by the application designer, adopting it to any new subject domain should be a fully automated process which requires no more than providing a corpus (or a proxy to automatically obtaining a corpus, such as a list of entities in the applications described in Chapter 6).

The next section (5.2) provides an overview of the framework, including the responsibilities of the different users. Section 5.3 describes the steps required for the definition of a GEMS application by the application designer. The following section (5.4) describes the models built and information extracted by the automated subject domain preparation step, initiated by the subject domain designer. Finally, Section 5.5 describes the generation pipeline used to generate text about an instance within a subject domain in a GEMS application.

5.1.2 The GEMS Application

GEMS is a family of generation applications, distinguished from other generation applications by the following:

1. A GEMS application is associated with a (closed or open) set of possible subject domains
2. The core structure (structure of central messages, types of central entity etc.) of the application can be defined abstractly without referring to the possible subject domains
3. Without knowing the subject domain, a GEMS application is a fully functional generation system. Knowledge of the subject domain can enhance it as described in this chapter

More formally, a GEMS application $\mathcal{A}$ defines a set of possible subject domains $\Delta^{\mathcal{A}}$, [1] which could be an open set, and each $\delta \in \Delta^{\mathcal{A}}$ contains a set of possible instances I^{δ}; and a set of functions $\Phi^{\mathcal{A}}$ which contains the following mapping functions (in this chapter, we use $\mathbb{E}$ to denote the set of all entities, $\mathbb{D}$ to denote the set of all discourse relations, and $\mathbb{M}$ to denote the set of all messages; we formally define a message later in this section):

[1] A note on notation: throughout this chapter, we will use superscripts on sets, variables and functions to denote membership - so, e.g, $\Delta^{\mathcal{A}}$ is the Δ of $\mathcal{A}$ as opposed to the Δ of another application.

1. A mapping from any subject domain δ to a set of *core entities* for the subject domain:
   $\mathit{CoreEnt}^{\mathcal{A}} : \{\delta | \delta \in \Delta^{\mathcal{A}}\} \rightarrow \mathbb{E}^{\mathbb{N}}$

2. A mapping from an instance $\odot$ (we use $\odot$ to denote an instance and reserve the use of i for counters) to a set of *core messages* for the instance:
   $\mathit{CoreMes}^{\mathcal{A}} : \{\odot | \exists \delta \in \Delta^{\mathcal{A}} : \odot \in I^{\delta}\} \rightarrow \mathbb{M}^{\mathbb{N}}$
   Note that $\mathit{CoreMes}^{\mathcal{A}}$ essentially defines what an instance in the domain includes (i.e., what sort of information is needed to create core messages for an instance).

While the previous two functions are both necessary and sufficient to define a GEMS application, in our framework we add another pair of optional functions. In contrast to the first two, which are dependent on the application, the next two should in principle be independent (i.e. they should be defined for language in general, regardless of the application). However, they are difficult general NLP problems which become significantly easier when the application is known, and can positively affect the quality of generated text:

3. An (optional) mapping from any message to an intrinsic preference score:
   $\mathit{BasePref}^{\mathcal{A}} : \mathbb{M} \rightarrow \mathbb{R}$

4. An (optional) mapping from any pair of messages to a set of possible discourse relations between them:
   $\mathit{DiscRel}^{\mathcal{A}} : \mathbb{M}^2 \rightarrow \mathbb{D}^{\mathbb{N}}$

Together, these four members of $\Phi^{\mathcal{A}}$ are the functions which must be provided by the application designer for a working GEMS application system. They constitute the whole of the manual work that needs to be done for a new application, which can then be applied to any subject domain (given a domain corpus).

5.1.3 Semantic Data Structures

Our main semantic structure is the *Semantic Typed Template (STT)*. An STT τ is a tuple $\langle V^{\tau}, R^{\tau}, L^{\tau} \rangle$ consisting of a set of vertices labeled with entity types $V^{\tau} = \{v_1^{\tau}, \ldots, v_n^{\tau}\}$, a set of edges labeled with relations among the vertices $R^{\tau} = \{r_1^{\tau}, \ldots, r_m^{\tau}\}$ and a set of lexical templates $L^{\tau} = \{l_1^{\tau}, \ldots, l_k^{\tau}\}$. The lexical templates L^{τ} are all assumed to be lexicalizations of the semantics of the STT and paraphrases of each other, and must be phrases or sentences (that is, multiple-sentence lexicalizations are not allowed). The STT represents both the meaning and some possible realizations of a sentence-level atomic unit of semantics, while not directly modeling the meaning in any way other than through the graph embodied in V^τ and R^τ. Instead, the meaning is grounded in the lexical template set.

A *message* μ, in turn, is a tuple $\langle \tau^\mu, E^\mu \rangle$ which consists of an STT τ^μ and a set of entities $E^\mu = \{e_1^\mu, \ldots, e_n^\mu\}$. The set of types V^{τ^μ} constrains the number and types of entities that are allowed to participate in E^μ, and the set of relations R^{τ^μ} further constrains the entities that are allowed to participate (the entities must have the proper relations among them).

Note that STTs, and by extension messages, are not true semantic structures in that they do not model the semantics explicitly. Instead, we rely on the paraphrasal nature of the lexicalizations to ensure that whenever a message (that is, an STT with specific entities) is realized, it always conveys the same meaning. There is some semantic representation in the form of entity types and the relations between them, but it does not generally cover the entire semantics of the STT.

5.2 Framework Overview

Figure 5.2 shows the entire pipeline for the GEMS generation framework. The arrows represent dependency.

The two columns at the right side of the figure show the levels of the pipeline. The left column is the description of each level, and the right column shows who is responsible for providing the content for the label. The algorithms in the Application Definition level are provided by the application designer; the domain corpus in the Subject Domain Definition level is provided by the subject domain designer; the Instance Data is provided as an input for each instance. Other than those, every component in the diagram is produced automatically.

The top two levels are the definition levels, where a human is involved. The first is described in Section 5.3. The second is trivial, since the domain designer need only provide a domain corpus. The third one is the preparation level which runs automatically for each

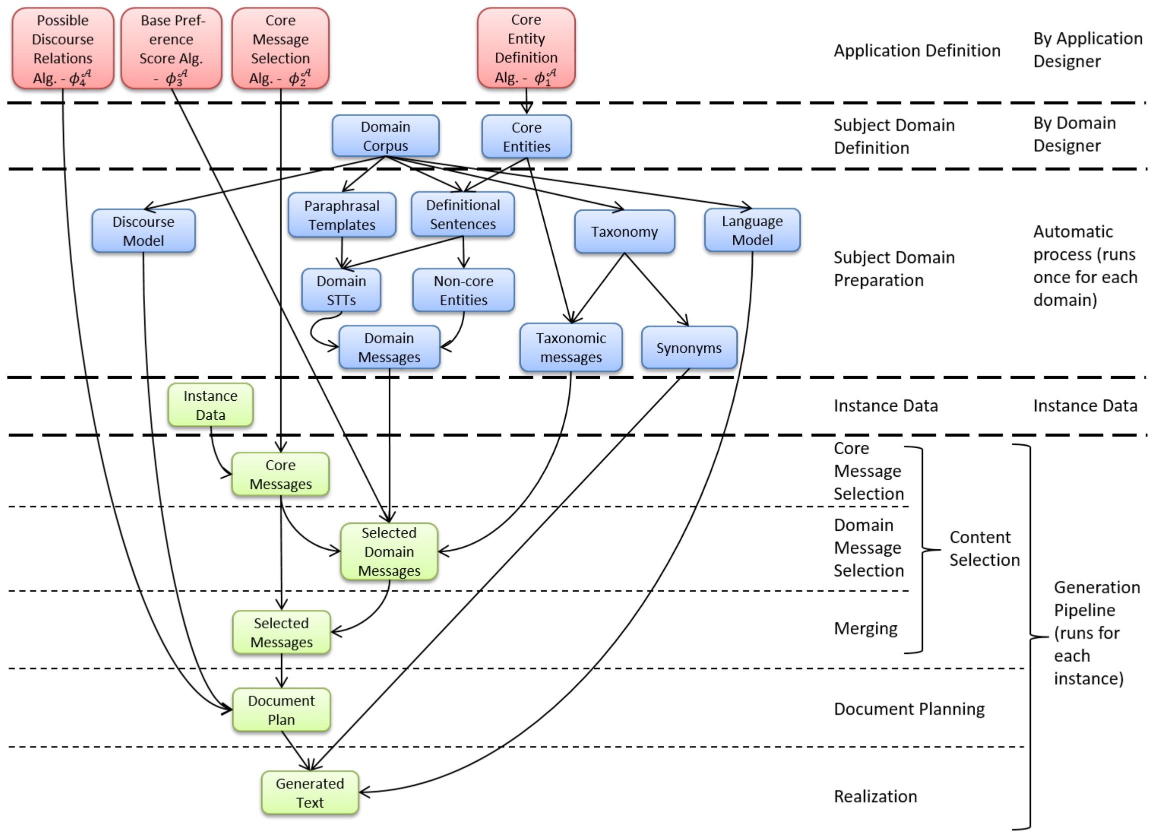


Figure 5.2: A framework for setting up and generating GEMS.

domain, given a domain corpus - this level is described in Section 5.4. Below that are the pipeline levels, which run once for each instance to be generated, and are described in detail in Section 5.5.

5.3 Defining an Application

Defining a new GEMS application is the responsibility of the application designer. It is a process that requires some thought regarding the scope and purpose of the application. We list a few of the relevant questions below.

- What is a subject domain in this application? the objective here is to maximize divergence in content and style across subject domains. For example, in a biography application, it makes more sense to think of different professions as different subject domains rather than, say, nationalities, because good biographies for different professions tend to diverge more (e.g., focus on different aspects of the person's traits and history; offer elaborations on different aspects; etc). In some cases the set of subject domains is a closed set, as in the set of industries for company descriptions. In other cases, as in the prediction justification domain (where a subject domain is any trained classifier), the set is much larger and open-ended.
- What is an instance in this application? i.e., what is a single occurrence of generation? note that this definition is completely independent of the subject domain. In the biography application, it is a single person; in the prediction justification application, it is a single prediction. A related question is what sort of information is provided with each instance: in the biography application, we expect RDF triples describing the person; in the prediction justification application, we require the prediction and the values of the features and other parameters of the model which made the prediction.
- What are the core frames or relations of the application? these will appear in every subject domain and much of the narrative will be structured around them. In the biography domain, for example, we think of birth, death, family, demographics etc as central in any biography, regardless of the subject domain (i.e., profession). In

the prediction justification domain, the relations of features to the prediction will be central to any justification, regardless of the classifier.

In more practical terms, the application designer needs to implement the four algorithms $CoreEnt^{\mathcal{A}}$, $CoreMes^{\mathcal{A}}$, $BasePref^{\mathcal{A}}$ and $DiscRel^{\mathcal{A}}$, described in Section 5.1 (although the last two are optional). We describe each in a subsection below, and provide concrete examples in the next two chapters.

5.3.1 Core Entities Definition Algorithm ($CoreEnt^{\mathcal{A}}$)

Each subject domain has a set of core entities: entities that are central to the domain and which serve as starting points for extracting domain-specific STTs, messages and additional entities. Section 5.4 describes how these entities are used for extracting domain messages, and Section 5.5 describes how they are given more weight in the content selection phase of the generation pipeline. Broadly, these are the entities which will take more prominence within a generated text, will be elaborated on and drive the narrative.

$CoreEnt^{\mathcal{A}}$ is simply an algorithm that, given a subject domain for the application (with its associated domain corpus) generates a list of core entities for the domain.

For example, in a domain of the biography application (i.e., a profession), these are all the people with that particular profession (note that in this particular case these core entities also happen to be the instances of the domain, but that is not the case in all applications). In a domain of the prediction justification application (i.e., a classifier), these are the features used by the classifier.

5.3.2 Core Message Selection Algorithm ($CoreMes^{\mathcal{A}}$)

As described earlier in this chapter, each instance (for any subject domain) has a set of associated core messages, which can then be expanded by the pipeline with domain-specific messages. The application designer is responsible for creating an algorithm for selecting the core messages of each instance. Generally, these will come from whatever information the instance contains, so an important part of defining $CoreMes^{\mathcal{A}}$ is defining what sort of information we can expect an instance to contain.

In some applications this is easy, as in Chapter 6 where we simply select the messages corresponding to the RDF triples of the instance entity. In other cases, as in the justification narratives of Chapter 7, it can be a complex research task: it is far from trivial to convert the values of features which we can expect each instance to contain to meaningful messages about the prediction.

Part of defining the core messages is defining the core STTs used by these messages, which represent the fundamental frames or relations expressed by this application. This can be a small manually-written set (as we have done in the justification application in Chapter 7), or it can be a template or algorithm used to derive them from the data of a subject domain (as we show in the RDF applications of Chapter 6). In principle, they can also be mined from an external source similarly to how we extract domain STTs in the next section.

5.3.3 Base Preference Score Algorithm ($\boldsymbol{BasePref^{A}}$)

An optional but powerful way to increase the quality of generated texts is to define an intrinsic base preference score for each message in the application. This score is used in the content selection phase of the pipeline, where it augments the generic entity coherence preference score we utilize. In the next chapter, we describe an example where the length of the message and the number of entities that match an RDF object are combined to approximate the likelihood that an extracted message is relevant and succinct.

If $BasePref^{A}$ is not defined, we use a default formulation where the base preference scores are 1 for core messages and -1 for domain messages. It is crucial that $BasePref^{A}$ has both positive and negative values, as explained in Section 5.5

5.3.4 Possible Discourse Relations Algorithm ($\boldsymbol{DiscRel^{A}}$)

While we have generic ways of inferring possible *expansion* and *comparison* between messages, *temporal* and *contingency* relations (and certain instances of the other relations as well) are often tied to the deeper semantics of the messages. An optional way to diversify the style and discourse structure of the generated text is to define possible discourse relations between types of messages in the application.

For example, in the RDF applications described in the next chapter, there are limited lists of RDF predicates within each application that we can manually define potential discourse relations over. In the justification application described in Chapter 7, certain types of STT combinations intrinsically entail discourse relations between them.

5.4 Preparing a New Subject Domain

Preparing a new subject domain requires the subject domain designer to provide a domain corpus. Paraphrasal templates (as described in Chapter 2), a taxonomy (Chapter 3) and a discourse model (Chapter 4) are extracted automatically from the corpus as described in previous chapters, as well as a language model for realization. Domain-specific STTs, entities and messages are automatically extracted as well.

In all three of our example GEMS applications in this thesis, we use Wikipedia as our source for domain corpora. While it is convenient for many reasons, and we take advantage of this convenience, there is nothing in the *framework* that requires the subject domain corpus to be a Wikipedia corpus. Note, however, that the application designer can make it a requirement that all subject domain corpora for a specific *application* come from a particular source or genre, as in our RDF applications (which require the corpus to be a Wikipedia corpus so that it matches the RDF data from DBPedia).

5.4.1 Extracting Domain STTs, Entities and Messages

The first step given a new subject domain and corpus is to extract *definitional sentences*. A definitional sentence is any sentence in the corpus which contains a core entity, based on the definition of core domain entities given by the application designer ($CoreEnt^{\mathcal{A}}$). For example, in the company descriptions application, in the hardware subject domain, definitional sentences for the core entity *Apple* may include “Apple is an American multinational technology company” and “In 1984, Apple launched the Macintosh, the first personal computer to be sold without a programming language at all”.

Each definitional sentence is templatized (as described in Chapter 2), resulting in a template and a set of entities. The entities in each definitional sentence are registered as additional domain entities. For the template itself, we use the paraphrasal template mining method described in Chapter 2 to compare it with existing STTs (both core STTs defined in advance by the application designer and STTs previously extracted by this step). If the template is found to be a paraphrasal template for an existing STT, it is added as a new lexicalization for that STT. Otherwise it is registered as a new STT.

In addition, a domain message is registered with the STT and the entities found in the definitional sentence (these messages may participate in a generated instance if the message expansion component chooses them). This gives us the set of domain messages, D^{δ}, which we will use in the generation pipeline.

5.4.2 Extracting the Domain Taxonomy

The taxonomy is extracted from an expanded Wikipedia corpus. We find entities in the domain corpus using the approach discussed in Chapter 2 (this is a side effect of extracting paraphrasal templates). Those entities are matched to Wikipedia articles, and any Wikipedia article mentioned more than once in the domain corpus becomes a part of the expanded corpus. We then apply our taxonomic classifier, described in Chapter 3, to all pairs of articles in this expanded corpus to build a domain taxonomy.

The taxonomy synonyms are used in the generation pipeline for lexical choice as described later in this chapter. In addition, the hypernyms can be used to generate taxonomic domain messages for core entities; those may or may not be interesting, depending on the application. In the prediction justification application we use these to define and provide more information on the features of a classifier. In the RDF applications we leave out these messages as they are unlikely to be useful (after all, no one needs to see a taxonomic definition of a person or a company). If the application designer decides to use taxonomic messages, they are created for each hyponym-hypernym pair in the extracted taxonomy where the hyponym is a core entity. These messages use a special, fixed STT that takes two entities of any type, and contains only a single template: "[$V_1^{\mathcal{T}}$] is a type of [$V_2^{\mathcal{T}}$]". The messages are then added to the pool of domain messages (and may or may not be selected for generation by the pipeline).

5.4.3 Extracting the Discourse Planning Model

A discourse planning model is extracted from the domain corpus as described in Chapter 4. We use our discourse parser to annotate the entire corpus with discourse relations and build the n-gram model of relations that is used by the generation pipeline in Section 5.5.

5.4.4 Extracting the Language Model

The language model used in the realization component of the pipeline is not a typical n-gram model. We are not trying to generate words within a sentence. Instead, we have a set of templates for each message to generate (which corresponds to a sentence or phrase in the final text) and we want to choose one that best fits the context. For this purpose, we define and extract three cross-sentence language models.

The first language model is a cross-sentence word pair model for pairs of words that appear in adjacent sentences. The probability that a word w appears in a sentence if word v appears in the previous sentence, independently of everything else, is

$$P(w|v) = \frac{Count(v,w)}{Count(v)}$$

For the probability of a particular template $\mathcal{T}$ given a selected previous sentence $\mathcal{S}$, we take the average over all word pairs:

$$P_{LM_1}(\mathcal{T}|\mathcal{S}) = \frac{\sum_{(w,v)\in\{\mathcal{T}\times\mathcal{S}\}} P(w|v)}{|\{\mathcal{T}\times\mathcal{S}\}|}$$

The second language model is a POS bigram pair model. It treats POS bigrams as individual words in the first model; in other words, $P_{LM_2}(\mathcal{T}|\mathcal{S})$ is defined in the same way as $P_{LM_1}(\mathcal{T}|\mathcal{S})$, except that w and v stand for POS bigrams (instead of words) in the candidate template and the selected previous sentence, respectively.

The third language model is a sentence length model. Here we compute the expected length of a sentence $\mathcal{T}$ given the length of the previous sentence $\mathcal{S}$ as

$$E[\#\mathcal{T}|\#\mathcal{S}] = \frac{\sum_{\{\sigma_i:\#\sigma_{i-1}=\#\mathcal{S}\}} \#\sigma_i}{|\{\sigma_i : \#\sigma_{i-1} = \#\mathcal{S}\}|}$$

where $\#\mathcal{S}$ is the length of sentence $\mathcal{S}$ in words. We then smooth this expectation estimate using the estimates of nearby lengths:

$$\tilde{E}[\#\mathcal{T}|\#\mathcal{S}] = \frac{\sum_{i=\#\mathcal{S}-3}^{\#\mathcal{S}+3} E[\#\mathcal{T}|i]}{7}$$

Based on this smoothed expectation, we define the probability of a template $\mathcal{T}$ given a selected previous sentence $\mathcal{S}$:

$$P_{LM_3}(\mathcal{T}|\mathcal{S}) \triangleq \frac{1}{(\#\mathcal{T} - \tilde{E}[\#\mathcal{T}|\#\mathcal{S}])^2}$$

This definition is not intended to have a true probabilistic interpretation, but it preserves an order of likelihood since it increases monotonically as the length of $\mathcal{T}$ gets closer to the expected values.

These three models are then used in Section 5.5 to provide rankings of all possible templates for a message to generate.

5.5 Generation Pipeline

In this section, we describe how an NLG system created by this framework (for a particular GEMS application and subject domain) operates. That is, we describe the process of generating text for an instance $\odot$ of subject domain δ in a GEMS application $\mathcal{A}$ (this section heavily relies on the definitions in Section 5.1).

The generation pipeline is shown in Figure 5.3, side by side with a traditional NLG pipeline showing which generation subtasks our components correspond to. The pipeline contains four components: *core message selection*, *subject domain message selection*, *discourse planning* and *realization*. Each one of the components is described individually below.

5.5.1 Core Message Selection

The core message selection component is application-dependent. Each GEMS application contains an algorithm $CoreMes^{\mathcal{A}}$ (as described in Section 5.1), written by the application

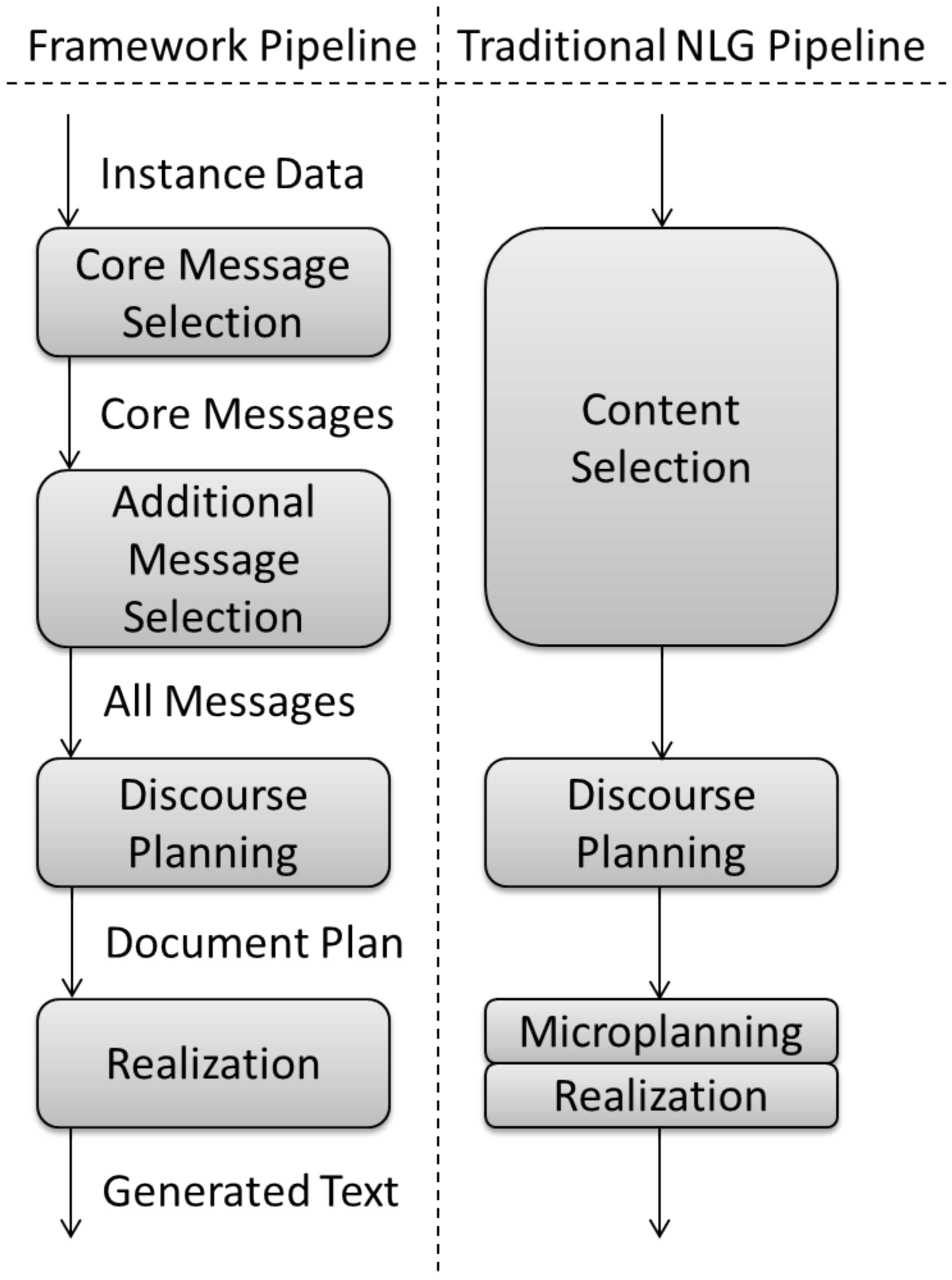


Figure 5.3: Our framework's generation pipeline, compared with the traditional NLG pipeline

designer, which maps the data of the instance into a set of core messages. Chapters 6 and 7 provide two examples of such an algorithm.

This component is conceptually similar to part of the content selection problem in general NLG applications. In a GEMS application, because of the unique structure of a well-defined task that can be applied to multiple subject domains, we can separate the content selection problem into two. The first (this component) is application-dependent and domain-agnostic (so it only needs to be defined once for each application), and handles the skeleton or core structure of the generated text; the second (the next component), which handles additional domain-specific content, is data-driven and does not need to be re-defined for new applications or domains.

5.5.2 Subject Domain Message Selection

The application designer also provides an algorithm $\mathit{CoreEnt}^{\mathcal{A}}$ which determines the set of *core entities* of the subject domain. From $\mathit{CoreEnt}^{\mathcal{A}}(\delta)$ and the set of core messages given by $\mathit{CoreMes}^{\mathcal{A}}(\odot)$ we have the set of core entities which participate in the core messages, $E^{\odot} = \{e | e \in \mathit{CoreEnt}^{\mathcal{A}}(\delta), \exists m \in \mathit{CoreMes}^{\mathcal{A}}(\odot) : e \in E^m\}$.

We also have the set of domain messages for the subject domain, M^{δ}, which are prepared (extracted from the domain corpus) ahead of time as described in Section 5.4. The set of *potential domain messages* is the subset of M^{δ} containing messages which have a core entity in common with the selected core messages: $P^{\odot} = \{p | p \in M^{\delta}, \exists e \in E^p : e \in E^{\odot}\}$. In this stage of the pipeline, we select a subset of $P^{\odot}$ to include in the generated text.

To select the subset of domain messages, we utilize the energy minimization framework described by Barzilay and Lapata (2005a). They describe a formulation that allows efficient optimization of what they call *independent scores* of content units and *link scores* among them through the energy minimization framework. The function to minimize is:

$$\sum_{p \in S} \mathit{ind}_N(p) + \sum_{p \in N} \mathit{ind}_S(p) + \sum_{\lambda \in L} \sum_{\substack{p_i \in S \\ p_j \in N}} \mathit{link}_{\lambda}(p_i, p_j)$$

where S is the subset of $P^{\odot}$ that is selected for generation, N is the subset not selected ($N = P^{\odot} \setminus S$), $\mathit{ind}_S(p)$ is p's intrinsic tendency to be selected, $\mathit{ind}_N(p)$ is p's intrinsic tendency to not be selected, L is the set of possible link types between messages and $link_\lambda(p_i, p_j)$ is the dependency score for the link of type λ between p_i and p_j. A globally optimal partition of $P^\odot$ to S and N can be found in polynomial time by constructing a particular kind of graph and finding a minimal cut partition (Greig et al., 1989).

We define the individual preference scores $ind(p)$ as an average of the similarity of p to each of the core messages in $CoreMes^{\mathcal{A}}(\odot)$ using the Jaccard coefficient as a similarity score:

$$ind(p) = \frac{\sum_{m \in CoreMes^{\mathcal{A}}(\odot)} J(p, m)}{|CoreMes^{\mathcal{A}}(\odot)|}$$

where

$$J(p, m) = \frac{E^p \cap E^m}{E^p \cup E^m}$$

Then, we use $BasePref^{\mathcal{A}}(p)$, the base intrinsic score provided by the application designer (see Section 5.1) to find $ind_S(p)$ and $ind_N(p)$:

$$ind_S(p) = \begin{cases} BasePref^{\mathcal{A}}(p) \times ind(p) & \text{if } BasePref^{\mathcal{A}}(p) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$ind_N(p) = \begin{cases} \frac{BasePref^{\mathcal{A}}(p)}{ind(p)} & \text{if } BasePref^{\mathcal{A}}(p) < 0 \\ 0 & \text{otherwise} \end{cases}$$

The intuition is that potential messages which have more in common with the core messages are more likely to be important. $BasePref^{\mathcal{A}}(p)$ provides the base preference, which is either positive or negative (by default, all domain messages have a negative base preference). If all base preferences were positive, the optimization above would always choose all available messages. With negative scores, only sets of messages which have high link scores and/or links to positive messages are selected.

The link scores $link(p_i, p_j)$ (we only use one type of link score, so the λ subscript is unnecessary) are defined using a type similarity score. In contrast to the individual preference scores, where we want to maximize the entity overlap with the core messages (after all, we would not want to include messages about completely new entities with no link to the core of the generated text), we should not encourage the domain messages to all share the same set of entities. Instead, we focus on a softer semantic similarity: shared entity types. This score enhances the coherence of the generated text (for example, by encouraging a focus on the executives of a company in a particular instance, and on its products in another) but allows a flexible range of messages to be selected. The link score definition is

$$link(p_i, p_j) = \frac{\sum_{(e_i,e_j)\in\{E^{p_i}\times E^{p_j}\}} typesim(e_i, e_j)}{|\{E^{p_i} \times E^{p_j}\}|}$$

where

$$typesim(e_i, e_j) = \begin{cases} 1 & \text{if } type(e_i) = type(e_j) \\ 0 & \text{otherwise} \end{cases}$$

and where the type of an entity $type(e)$ is defined by the external taxonomy described in Chapter 2 (as opposed to the domain taxonomy which we extract ourselves).

Denoting the subset of $P^{\odot}$ selected by this process as $selected(P^{\odot})$, at the end of this process, we have $M^{\odot} = CoreMes^{\mathcal{A}}(\odot) \cup selected(P^{\odot})$ - the full set of messages to be generated.

5.5.3 Discourse Planning

The discourse planning component transforms the unordered set of messages $M^{\odot}$ into an ordered sequence of paragraphs $\mathcal{P}^{\odot} = (p_1, \ldots, p_k)$ where each paragraph p_i is an ordered discourse sequence $p_i = (m_1, r_1, m_2, r_2, \ldots, r_{n-1}, m_n)$, where the alternating m_i and r_i are messages and discourse relations, respectively.

First, we calculate the semantic similarity of each pair of messages in $M^{\odot}$ as follows:

$$sim(m_i, m_j) = \cos(\mathcal{V}_{\psi_{m_i}}, \mathcal{V}_{\psi_{m_j}})link(m_i, m_j)$$

where ψ_{m_i} is the pseudo-sentence of message m_i, constructed by concatenating all of its templates; $\mathcal{V}_{\psi_{m_i}}$ is the vector representing ψ_{m_i}, constructed via Equation 2.1; and $link(m_i, m_j)$ is defined as in the previous component. Essentially, this is a combination of the entity type-based semantic similarity and the distributional similarity of the lexicalizations.

We use single-linkage agglomerative clustering (with a stopping criteria of $sim(m_i, m_j) \leq 0.05$) to group the messages into semantic groups of messages that are similar in topic. Then, for each semantic group, we build a Multigraph of Possible Relations (MPR) as described in Chapter 4. In this thesis, we use the four *class-level* discourse relations for the PDTB: *expansion*, *comparison*, *contingency* and *temporal*. To build the MPR, we use the application-specific $DiscRel^{\mathcal{A}}$ mapping function provided by the application designer, as well as the following global rules:

1. If $\tau^{m_i} = \tau^{m_j}$ and $E^{m_i} \cap E^{m_j} = \emptyset$ (that is, the STTs of m_i and m_j are the same but they have no entities in common) then there is a potential *comparison* relation between them
2. If $J(m_i, m_j) \geq 0.5$ then there is a potential *expansion* relation between them
3. All messages have a potential *norel* relation between them

Once the MPR is built, we use the discourse model extracted from the domain corpus in Section 5.4 to generate a discourse sequence as described in Chapter 4. However, in contrast to the evaluation in Chapter 4 where we focused only on discourse coherence, here we want to utilize entity coherence in addition to discourse sequence coherence. We augment the probabilities coming from the discourse n-gram model $P^{\delta}_{\mathcal{D}}(r_i|R_{i-1})$, where R_{i-1} is the sequence of relations chosen so far, with the entity coherence score $J(m_i, m_{i-1})$, so that the probability of each edge in the graph is given by

$$P(r_i|R_{i-1}, m_i, m_{i-1}) = P^{\delta}_{\mathcal{D}}(r_i|R_{i-1})J(m_i, m_{i-1})$$

The discourse sequence is created stochastically from the MPR using these probabilities. Then, we break the discourse sequence into paragraphs that do not contain *norel* relations. Concatenating all of the paragraphs built from the discourse sequences of all semantic groups, we have an unordered set of paragraphs $\mathcal{P}^{\odot}$, where each p_i is an ordered discourse sequence of messages and relations.

To order the paragraphs, we use the following importance score:

$$imp(p_i) = \frac{\sum_{m \in p_i} |\{e | e \in E^m, e \in \mathit{CoreEnt}^{\mathcal{A}}(\delta)\}| \mathit{BasePref}^{\mathcal{A}}(m)}{|p_i|}$$

which is the average number of core entities in a message of p_i, weighted by the optional base preference score $\mathit{BasePref}^{\mathcal{A}}(m)$. The paragraphs are then sorted in decreasing order using this score, so that the paragraphs containing the most important messages tend to appear earlier in the text.

5.5.4 Realization

At this stage, we have the ordered set of paragraphs $\mathcal{P}^{\odot}$ to be realized. Recall that each paragraph p_i is an ordered set of messages and discourse relations, $(m_1, r_1, m_2, r_2, \ldots, r_{n-1}, m_n)$. To generate a paragraph, we iterate through the messages and make three decisions for each:

1. Select a template to use for realizing the message
2. Make lexical choice changes
3. Select a discourse connective, or choose not to use one, for each discourse relation (this is done for the relation *preceding* the message, so it does not apply to the first message in the sequence)

5.5.4.1 Template Selection

Selecting a template is done using the three language models prepared ahead of time, as described in Section 5.4. We build a ranker from each one of the models, and choose the template (lexicalization) from $\{l \in L^{\tau^{m_i}}\}$ that maximizes the the sum of ranks given the previously realized sentence (in the paragraph) s:

$$\hat{l} = \operatorname*{argmax}_{l \in L^{\tau^{m_i}}} \Big(rank\big(P_{LM_1}(l|s)\big) + rank\big(P_{LM_2}(l|s)\big) + rank\big(P_{LM_3}(l|s)\big) \Big)$$

Once the template is chosen, we fill the slots with the entities E^{m_i} to transform it into a sentence.

5.5.4.2 Lexical Choice

At this stage, we have a sentence that was created from a template. The template has a fixed set of words, which we may want to change in some cases - specifically when they seem to fit the context more smoothly than the original choices.

We iterate through every n-gram (for $n \leq 4$) g in the sentence, in order, and find its synonyms in the domain taxonomy (extracted as described in Section 5.4). We now have the set $G = g \cup synonyms(g)$, which are the possible lexical choices for g. We choose the lexicalization as:

$$\hat{g} = \operatorname*{argmax}_{\tilde{g} \in G} \cos(\mathcal{V}_{\tilde{g}}, \mathcal{V}_{context(g)})\eta$$

where $\mathcal{V}_{\tilde{g}}$ is the vector of n-gram $\tilde{g}$ constructed using Equation 2.1, and $context(g)$ is an n-gram containing the 5 tokens to the left and right of g in the *original* sentence (that is, regardless of any lexical choices already made). η is a tuned parameter intended to make it harder for alternative lexical choices to be made. $\eta = 1$ for g, and we set it to 0.75 for all members of $synonyms(g)$.

Note that for the vast majority of n-grams in any sentence, $synonyms(g)$ is empty and this process is trivial since there are no choices other than the original g.

5.5.4.3 Discourse Connectives

At this point we have the final lexical form of the message, and the last task is to link it with the previous sentence. We have a set of discourse connective templates for each one of the 4 class-level PDTB relations (Table 5.2 shows the list of connective templates), and we know the relation between the message and the previous message. We randomly select a connective, with a 50% chance of having no connective and a uniform distribution among the connectives for the relation, with the following modifications:

- If the two sentences together are larger than 40 words, connective templates that do not include a sentence boundary have a 0% chance of being selected (to avoid long, cumbersome sentences)

Relation	Connective templates	Relation	Connective templates
Expansion	m_i and m_j	Contingency	Because m_i, m_j
	m_i. Additionally, m_j		Since m_i, m_j
	m_i. Also, m_j		m_i, so m_j
	m_i. Besides, m_j		m_i, so that m_j
	m_i. Furthermore, m_j		m_i, therefore m_j
	m_i. In addition, m_j		m_i, thus m_j
	m_i. Moreover, m_j		m_i, which means that m_j
Comparison	m_i. In comparison, m_j	Temporal	m_i. Afterwards, m_j
	m_i. In contrast, m_j		m_i. Eventually, m_j
	m_i. However, m_j		m_i. Finally, m_j
	m_i. In turn, m_j		m_i. Then, m_j
	Although m_i, m_j		m_i. Later, m_j
	While m_i, m_j		m_i. Next, m_j
	Whereas m_i, m_j		m_i. Subsequently, m_j

Table 5.2: Discourse connective templates for each discourse relation.

- A connective that has already been chosen in the paragraph gets half the chance of being chosen again, and this effect is cumulative (to avoid repetitive connective choices)
- The chance that no connective will be chosen is increased by an additional 10% if a connective was chosen for the relation immediately preceding this one, and this effect is cumulative (so long chains of sentences with connectives in between them is less likely)

At the end of this step, all paragraphs are generated with fully lexicalized sentences and discourse connectives.

5.6 Conclusion

In this chapter, we introduced and defined a family of generation applications called Generation Endeavors with Modular Subjects (GEMS). We described a framework that leverages the unique structure of these applications to allow automatic creation of domain-adapted hybrid C2T-T2T generation systems for new subject domains if a few abstract functions are defined by an application designer, once per application. This framework drastically reduces the amount of manual work required in building general generation systems.

Our framework consists of three very different life-cycle parts: the definition of the application by an application designer, the adaptation of the application to a new domain, and the generation of an instance by an adapted application. In virtually all previous approaches, the first two parts are unified into one: the building of the generation system. Because of this lack of separation, generation systems exist on a range between very general-purpose systems that are very repetitive (because they do not contain the nuances of different domains), and domain-specific systems that need to be rebuilt for each new domain. In our framework, the bottlenecks which exist in creating generation systems for new domains are confined to the second part, leaving the first part (which requires the work of a human application designer) relatively small and well-defined. We then use the approaches we described earlier in the thesis to solve the bottlenecks of the second part in an automated, data-driven way.

The framework relies on the data-driven methods we introduced in the first part of the thesis, and should be viewed as one example of how they can be used to solve bottlenecks in NLG. That said, those methods do not exist specifically for this framework and can be used in many other ways by other generation systems and frameworks.

In the following two chapters we will describe three examples of GEMS applications and evaluate their output as produced by this framework for multiple subject domains.

Chapter 6

Semantic Web Applications and Evaluation

In this chapter, we introduce two examples of GEMS applications that generate descriptions from semantic web data: *biography* and *company description*. We describe the algorithms that define these applications and adapt each to two subject domains. We then show examples of text generated for each subject domain in each applications, and use the four scenarios to evaluate our framework and its components in a human study.

The two GEMS applications described in this chapter can be said to belong to a sub-family of GEMS: applications which generate descriptions of entities based on RDF (Resource Description Framework) data. This special family allows even greater automation of the system creation process than that described in the previous chapter. Specifically, the mapping functions $\mathit{CoreEnt}^{\mathcal{A}}$, $\mathit{CoreMes}^{\mathcal{A}}$ and $\mathit{BasePref}^{\mathcal{A}}$ can be defined abstractly for this entire family.

6.1 General RDF Applications Definitions

Before we go into the definition of an RDF application, we shall introduce RDF itself. RDF is a framework for organizing data which revolves around the concept of *triples*. Each triple contains a subject, a predicate and an object. In this chapter, we will use the notation s^t, p^t and o^t to denote the subject, predicate and object of triple t, respectively. In this chapter, we use DBPedia (Auer et al., 2007) as our source of RDF data.

Each RDF application is defined by a core entity type $\eta^{\mathcal{A}}$, and a domain-differentiating predicate $\pi^{\mathcal{A}}$. For the biography application, $\eta^{\mathcal{B}} = \textit{person}$ and $\pi^{\mathcal{B}} = \textit{occupation}$, and for the company description application $\eta^{\mathcal{C}} = \textit{company}$ and $\pi^{\mathcal{C}} = \textit{industry}$. Intuitively, $\eta^{\mathcal{A}}$ represents the type of entities that can be instances of the application (and that we can use this application to generate a description about), and $\pi^{\mathcal{A}}$ is the predicate which governs the separation of entities of type $\eta^{\mathcal{A}}$ into different subject domains within the application. Each subject domain δ is defined by a type σ^{δ} which corresponds to a possible object of predicate $\pi^{\mathcal{A}}$. For example, in the biography application, possible σ^{δ}'s are occupations such as *politician*, *scientist* and *athlete*. Each instance in a domain of an RDF application (for example, "Barack Obama" in the *politician* domain of the *biography* application) is defined by an entity $e^{\odot}$ (recall that we use $\odot$ as the symbol for an instance), where $\textit{type}(e^{\odot}) = \eta^{\mathcal{A}}$ and there exists a triple t such that the predicate $p^t = \pi^{\mathcal{A}}$, the subject $s^t = e^{\odot}$ and the object $o^t = \sigma^{\delta}$.

The sets of RDF triples for an application $\mathcal{A}$, a subject domain δ and an instance $\odot$ are defined, respectively, as:

- RDF triples for application $\mathcal{A}$: $T^{\mathcal{A}} = \{t | \textit{type}(s^t) = \eta^{\mathcal{A}}\}$
- RDF triples for subject domain $\delta^{\mathcal{A}}$: $T^{\delta} = \{t | t \in T^{\mathcal{A}}, \exists \tilde{t} \in T^{\mathcal{A}} : s^{\tilde{t}} = s^t, p^{\tilde{t}} = \pi^{\mathcal{A}}, o^{\tilde{t}} = \sigma^{\delta}\}$
- RDF triples for instance $\odot$: $T^{\odot} = \{t | t \in T^{\delta}, s^t = e^{\odot}\}$

Now, using these definitions, we can abstractly define three of the four functions in $\Phi^{\mathcal{A}}$ for all RDF applications.

$\textit{CoreEnt}^{\mathcal{A}}$, the function which defines the core entities of each subject domain, is defined as $\textit{CoreEnt}^{\mathcal{A}}(\delta) = \{e | \exists t \in T^{\delta} : e = s^t\}$. In other words, all entities which are subjects in at least one triple of the subject domain triple set T^{δ}. Based on the definition of T^{δ}, the core entities in the subject domain are then all entities which satisfy the application type requirements (e.g., being a person in the biography application) as well as the domain predicate argument requirement (e.g., being a scientist in the scientists subject domain).

$CoreMes^{\mathcal{A}}$, the function which defines the core messages of each instance, is defined as $CoreMes^{\mathcal{A}}(\odot) = \{rdfmessage(t) | t \in T^{\odot}\}$. That is, for each instance, we produce one core message from each RDF triple that has the instance's entity as the subject.

To create a message from an RDF triple, we first match it to an STT based on the predicate. Each predicate that participates in T^{δ} becomes an STT τ with two entity types (the type of the core entity and the type of the object) in V^{τ}; a single relation between the two types (the predicate) in R^{τ}; and a set of simple initial templates in L^{τ}:

- The (PREDICATE) of $[v_1^{\tau}]$ is $[v_2^{\tau}]$
- $[v_1^{\tau}]$'s (PREDICATE) is $[v_2^{\tau}]$

where (PREDICATE) is replaced with the relevant predicate. Additional templates are then found using paraphrasal template mining as described in the previous chapter. We also create plural versions for cases where v_2^{τ} is a list of entities.

For example, in the biography domain, we create an STT τ for the *birthDate* predicate with $V^{\tau} = \{person, date\}$; $R^{\tau} = \{v_1^{\tau}$ birthDate $v_2^{\tau}\}$; and an initial template set $L^{\tau} = \{$"The birth date of $[v_1^{\tau}]$ is $[v_2^{\tau}]$", "$[v_1^{\tau}]$'s birth date is $[v_2^{\tau}]$"$\}$. In the preparation stage described in the previous chapter L^{τ} may be expanded with paraphrasal templates found in the corpus, for example "$[v_1^{\tau}]$ was born in $[v_2^{\tau}]$".

We then create a message that contains the relevant predicate STT and the entities in the triple. In case there are multiple triples with the same subject and predicate but different objects, we create a single message out of them with a plural version of the STT and define the second entity as the list of all objects.

$BasePref^{\mathcal{A}}$ defines the base preference score of each message. We define the preference score of any core message as 1, and the preference score of domain messages as follows. Let M^{μ} be the set of entities in message μ that are matched by at least one triple in T^{δ}. That is, if the definitional sentence the message was extracted from contains a subject-object pair that exists in any triple in T^{δ}, both the subject and the object will be members of M^{μ}. Another way of saying this is that M^{μ} consists of the subset of E^{μ} whose members participate in any relation in $R^{\tau^{\mu}}$. Then, we define the preference score as:

$$BasePref^{\mathcal{A}}(\mu) = \begin{cases} |R^{\tau^{\mu}}| & \text{if } M^{\mu} = E^{\mu} \\ -|E^{\mu} \setminus M^{\mu}| \frac{\#L^{\tau^{\mu}}}{10} & \text{otherwise} \end{cases}$$

where $\#L^{\tau^{\mu}}$ is the average length in words of the templates of the STT τ^{μ}. This definition results in a positive preference for any message where all entities participate in a relation, whose weight is the number of relations it covers (note that the score of 1 for core messages falls out of this equation, since they represent a single triple by definition); conversely, messages which have entities that do not participate in a relation (*unaccounted entities*) have a negative preference score which increases in magnitude with the number of unaccounted entities and with the length of the templates realizing them. The intuition is that a long message containing many entities that match no triples is unlikely to be relevant.

In our experiments in this chapter, we extract the domain corpus for each subject domain by collecting the Wikipedia articles of all core entities for that subject domain, which further automates the process. However, there is nothing about RDF applications that requires the corpus to be collected in this way.

6.2 Specific Application Definitions and Subject Domains

In this section we provide descriptions of the two RDF applications we are concerned with in this chapter, as well as the specific definitions not covered in the previous section. In addition, we introduce two subject domains for each application, which we will use to generate the texts used in the evaluation later in this chapter. We provide an example of generated text for each subject domain in the next section.

6.2.1 Biography

The biography application is aimed at generating descriptions of the life, accomplishments and significance of noted individuals. Because of the focus on accomplishments, the subject domains are separated by *occupation*. Generated biographies can (and should) look quite different for different occupations. The two subject domains we picked to illustrate this application are *politicians* and *models*.

We refer to the biography application as $\mathcal{B}$. The basic definitions of this RDF application are:

- $\eta^{\mathcal{B}} = person$
- $\pi^{\mathcal{B}} = occupation$ ($\Delta^{\mathcal{B}} = \{politician, model, scientist, athlete...\}$)

We also define $DiscRel^{\mathcal{B}}$, the function that determines possible discourse relations between messages, using some of the more common predicate pairs in the RDF triples of this application. Table 6.1 shows the possible discourse relations between the predicate pairs. $DiscRel^{\mathcal{B}}(m_i, m_j)$ contains all discourse relations shown in Table 6.1 between the predicate corresponding to τ^{m_i} and the predicate corresponding to τ^{m_j} if both m_i and m_j are core messages (which are always created from a single predicate), and $\{norel\}$ otherwise.

Predicate pair	Possible discourse relations
birthDate, deathDate	temporal
birthPlace, residence	temporal, comparison
birthPlace, deathPlace	temporal, comparison
deathPlace, restingPlace	temporal, comparison
birthPlace, nationality	contingency
birthPlace, citizenship	contingency
ethnicity, religion	contingency
predecessor, successor	temporal, comparison
influencer, influenced	temporal
doctoralAdvisor, doctoralStudent	temporal

Table 6.1: Possible discourse relations between common predicates of the biography application

6.2.2 Company Descriptions

The company descriptions application is aimed at generating descriptions of companies and their history, products, key executives, etc. The subject domains are separated by *industry*.

The two subject domains we picked to illustrate this application are the *automotive* and *video game* industries.

We refer to the company descriptions application as $\mathcal{C}$. The basic definitions of this RDF application are:

- $\eta^{\mathcal{C}} = company$
- $\pi^{\mathcal{C}} = industry$ $(\Delta^{\mathcal{C}} = \{automotive, videogame, bank, film...\})$

We also define $DiscRel^{\mathcal{C}}$, the function that determines possible discourse relations between messages, in the same way we did for the biography application - using some of the more common predicate pairs in the RDF triples of this application. Table 6.2 shows the possible discourse relations between the predicate pairs.

Predicate pair	Possible discourse relations
foundationPlace, location	temporal
division, owner	comparison
division, parentCompany	comparison
subsidiary, owner	comparison
subsidiary, parentCompany	comparison
founder, owner	comparison
revenue, netIncome	comparison
revenue, operatingIncome	comparison
netIncome, operatingIncome	comparison
foundingDate, extinctionDate	temporal, comparison

Table 6.2: Possible discourse relations between common predicates of the biography application

6.3 Examples

In this section, we present four examples of generated texts, one from each application and domain. The examples are:

1. Figure 6.1, showing a *biography* of a *model*, *Candice Bergen*.
2. Figure 6.3, showing a *biography* of a *politician*, *Vicente Fox*.
3. Figure 6.4, showing a *company description* of an *automotive* company, *Lexus*.
4. Figure 6.5, showing a *company description* of a *video games* company, *Sega*.

To show how different components contributed to the generated text, we mark sentences which were generated from extracted (T2T) domain messages in **bold**, and sentences which were generated from core messages but used an extracted paraphrasal template in *italics*. Sentences in unmarked typeface are those that were generated from core messages using one of the default templates. Underlined terms are those that were inserted by lexical choice (and replace some other, original term).

For ease of analysis, we put endlines between sentences of the same paragraphs (so each sentence starts on a new line) and an empty line between paragraphs. Each sentence is marked with a number to make discussion simpler.

The first example in Figure 6.1, a biography of the model *Candice Bergen*, is a relatively short one (and therefore easy to follow) and contains many of the phenomena we want to discuss. First, note the topical divisions of sentences into paragraphs: recall from Chapter 5 that paragraph boundaries are created by a combination of distributional and type dissimilarity, in addition to where no discourse relations are available. This particular text is too short for the discourse model to have a significant effect on intra-paragraph ordering (we will discuss it with the following examples), but it does create the separation of the first and second paragraph (because there is no possible discourse relation between sentence 2 and 3 in this case, although perhaps there should be).

Next, note the paraphrasal templates used for the core messages in sentences 1 and 2. Without the extracted templates, these sentences would have been realized by C2T as "the birth name of Candice Bergen is 'Candice Patricia Bergen' " and "the birth place of Candice Bergen is Beverly Hills, California", so clearly the extracted templates make the text better, and the realization language model accurately selects them in this case. Sentence 4 is based on a domain message, and extracted as-is from the domain corpus. This is an example of the hybrid (C2T - T2T) nature of our approach. The other sentences are default core

1. *Candice Bergen began life as "Candice Patricia Bergen".*
2. *Candice Bergen was born and raised in Beverly Hills, California.*

3. The birth date of Candice Bergen is 1946-05-09.
4. **Candice Bergen started her acting with advertisements in 1965.**

5. The parents of Candice Bergen are Frances Bergen and Edgar Bergen.

6. The alma mater of Candice Bergen is University of Pennsylvania.
7. Candice Bergen's occupation is Model.

Figure 6.1: Sample generated text for the *biography* application in the *model* domain.

message realizations. We think that the addition of the templates and extracted sentences make these appear more natural (imagine the alternative - 6 sentences all using the same two default templates in a sequence).

One problem that is immediately visible in this example (and will be seen again in the other examples), and that is clearly a weakness of our approach, is the repetition of proper nouns due to the lack of pronouns and aggregation (although our approach does have a form of aggregation through certain discourse connectives, it does not take care of the proper noun repetition problem). Almost every sentence in our generated texts contains the name of the main entity explicitly. This is something that we think should be handled in a post-processing step, and will be a relatively low-hanging fruit for improving the quality of the generated texts in future work. For example, Figure 6.2 shows the same Candice Bergen text with a rule for pronoun addition: the full name of the entity is used in the first sentence of each paragraph, and is replaced with a pronoun for all other sentences in the paragraph. While not perfect, even this simple heuristic enhances the readability of the text. The evaluations in this chapter, however, were done without any sort of post-processing.

The second example, the biography of *Vicente Fox* in Figure 6.3, is twice as long as the first one. It contains both more core messages and more extracted domain messages. Because of its length (and in particular, the length of its main paragraph), discourse plan-

1. *Candice Bergen began life as "Candice Patricia Bergen".*
2. *She was born and raised in Beverly Hills, California.*

3. The birth date of Candice Bergen is 1946-05-09.
4. **She started her acting with advertisements in 1965.**

5. The parents of Candice Bergen are Frances Bergen and Edgar Bergen.

6. The alma mater of Candice Bergen is University of Pennsylvania.
7. Her occupation is Model.

Figure 6.2: The same *biography*, *model* sample with pronouns added.

ning plays a more central part. The paragraphs are still topical: the first (short) one about his birth, the second single-sentence paragraph about his successors (and perhaps this one should not have been separated out from the next one). The third, main paragraph is about his background and political career, and the fourth is about his marriages. Looking more closely into the third paragraph, the ordering is not perfect but there is some reasonable order. For example, sentences 4 through 6 are about his background, while sentences 7 through 10 are about his political career. The two domain messages (sentences 9 and 10) complement the general information of the core messages (7-8) with more detailed information. The same thing happens in the last paragraph, with the domain messages 11-12 and the core message 13. Two of the discourse relations in this example are realized with connectives: the expansion between the messages of sentences 9 and 10 (realized with "moreover"), and the expansion between the two messages aggregated together as sentence 5 (realized with "and"). The other relations in the text (all expansions, in this case) were realized as implicit relations, without connectives.

One problem that can be seen in this example is the repetition of information in the first paragraph: the second sentence was extracted from the corpus, while the first sentence comes from the RDF triples. They contain (partially) the same information, but they were not matched as paraphrases because they do not contain the exact same entities. We leave

1. Vicente Fox's birth places are San Francisco del Rincón and Guanajuato.
2. **Vicente Fox was born in Guanajuato on July 2, 1942, the second of nine children.**

3. The successors of Vicente Fox are Ramón Martín Huerta and Felipe Calderón.

4. The occupations of Vicente Fox are Politician and Businessperson.
5. The birth date of Vicente Fox is 1942-07-02, and Vicente Fox's religion is Catholic Church.
6. Vicente Fox's alma maters are Harvard Business School and Universidad Iberoamericana.
7. The party of Vicente Fox is National Action Party.
8. Vicente Fox's offices are (alongside Pier Ferdinando Casini), Governor of Guanajuato and Co-President of Centrist Democrat International.
9. **On July 7, 1997, Vicente Fox decided to run for President of Mexico.**
10. Moreover, **during his campaign for president, Vicente Fox became well known for his unique cowboy style and popular charisma.**

11. **Vicente Fox married a receptionist at Coca-Cola, Lilian de la Concha.**
12. **Vicente Fox married for the second time while in office as President.**
13. The spouse of Vicente Fox is Marta Sahagún.

Figure 6.3: Sample generated text for the *biography* application in the *politician* domain.

solving issues like this to future work.

The next example, in Figure 6.4, is the first *company description* example, and much longer than the first two. It contains a description of the automotive company *Lexus*. Here, the number of extracted domain messages is vastly larger, and the large paragraphs are mainly composed of those (including two paragraphs that are completely made of domain messages). The paragraphs are visibly less well-composed here, as size increases complexity, but there is still reasonable order. The fourth paragraph (sentences 4-7) is mostly about the founding of the company; the one following that (8-9) is about the product naming scheme; the next one (10-17) mostly about background and history, while the last one (18-23) contains background as well but has local foci on operations and on hybrid vehicles. Obviously, some sentences could be better located: sentence 11, for example, should be part of the fourth paragraph (4-7); sentences 12 and 15 should possibly also be moved to the end of that paragraph. Sentence 16 should be moved to the last paragraph, next to the others dealing with hybrid vehicles. Also, note the lexical choice injections in sentence 7 (where *marque* was chosen to replace the original synonym *brand*) and 22 (where *cars* was chosen to replace *vehicles*) .

Finally, the last example is shown in Figure 6.5, with a description of the video games producer *Sega*. This is another long text with many domain messages. There are a few interesting things in this description. First, note the comparison relation, marked with the discourse connective *in comparison*, in sentence 4: while there are few non-expansion relations in our chosen examples (and as we discuss in Chapter 4, expansion relations generally dominate Wikipedia), they do exist and are not always marked with an explicit connective. It is important to remember that although discourse connectives are not always used, the discourse model relies on discourse relations to determine ordering - these are just not always visible in the text.

Second, note the paraphrasal template in sentence 6. It is used to generate the relation *foundingYear*, but is not exactly correct: Sega was not formed as a *mobile* game development company. If we could remove the adjective *mobile*, it would be perfectly correct. This is an interesting direction for future work. Of course, errors like this are inevitable when using a statistical method for paraphrasing - but this example highlights the importance of using a

1. The product of Lexus is Luxury vehicle.

2. Lexus' homepage is http://www.lexus-int.com/.

3. *Lexus is headquartered in Nagoya, Japan.*

4. The founding year of Lexus is 1989.

5. **That same year, Lexus also became one of the first marques to debut a certified pre-owned program, with the aim of improving trade-in model values.**

6. **Despite being an upstart, Lexus established instant customer loyalty and its debut was generally regarded as a major shock to the pedigree luxury marques.**

7. **They also represent their Lexus marque in other sports car racing categories.**

8. **Lexus production models are named alphanumerically using two-letter designations followed by three digits.**

9. **The first letter indicates relative status in the Lexus model range, and the second letter refers to car body style or type.**

10. Lexus' key persons are General manager, Chief executive officer and Vice president.

11. Lexus is founded by Eiji Toyoda.

12. **From its inception, Lexus has been advertised to luxury consumers using specific marketing strategies, with a consistent motif used for the brand's advertisements.**

13. **In industry ratings of build quality, owner satisfaction, and reliability, Lexus vehicles have outperformed other manufacturers in successive years.**

14. **Lexus produces its highest-performance models under its F brand division.**

15. **The launch of Lexus was heralded by a multimillion dollar advertising campaign in both television and print media.**

16. **Continuously variable transmissions, regenerative brakes, and electric motors have been used on all Lexus hybrid models.**

17. **Other officially sanctioned regional distributors have sold Lexus models prior to the launch of, or in absence of, a dedicated dealership network.**

18. **Financial data of Lexus operations are not disclosed publicly.**

19. **Lexus sales operations vary in structure by region.**

20. **Lexus models sold in Japan featured higher specifications and a price premium compared with their discontinued Toyota counterparts.**

21. **By the mid-1990s, Infiniti was lagging behind Lexus and Acura in sales.**

22. **Toyota is also pushing hybrid cars in the US such as the Prius, Camry Hybrid, Highlander Hybrid, and various Lexus products.**

23. **It labeled such technology in Toyota cars as "Hybrid Synergy Drive" and in Lexus versions as "Lexus Hybrid Drive".**

Figure 6.4: Sample generated text for the *company descriptions* application in the *automotive* domain.

1. Sega's key persons are Yu Suzuki and David Rosen.

2. Sega's homepages are http://www.sega.co.jp/, http://www.sega.co.uk/, http://www.sega.com/ and http://www.playsega.com/.

3. Sega's net income is 41,500,000,000 JPY.
4. In comparison, Sega's revenue is 396,700,000,000 JPY.
5. In addition, *Sega has around 2208 employees.*

6. *Sega was founded in 1940 as a mobile game development and publishing company.*
7. **But it surely is the best segment in Sega from an investors point of view as the profitability is still much higher than that of the general market.**
8. **Sega has had a long history of different slogans and ad campaigns.**
9. The location country of Sega is Japan.
10. **Their strategy was to make the hardware reject any cartridge that did not include a Sega trademark.**
11. **If an unlicensed company included this trademark in their game, Sega could sue the company for trademark infringement.**
12. **Sega is closing and reopening facilities on an ongoing basis using a scrap and rebuild strategy.**
13. The location city of Sega is Ōta, Tokyo.
14. **Sega's Consumer Business segment is producing and distributing games for consoles and pc.**
15. In addition, **Sega has also been publishing games from independent studios, and is currently considering turning them into franchises.**
16. **They were both very involved in the arcade business, and Sega's fame also comes from their console systems.**
17. **With the merger, Sega reabsorbed its second party studios and began to reorganize them.**
18. The foundation places of Sega are United States and Honolulu.
19. **The shift to software development affected Sega's Australian operations.**

20. **Someone yelling "SEGA!".**
21. **Do me a favor, plug me into a Sega.**
22. **Sega does what Nintendo won't!**
23. **The latter was canceled for undisclosed reasons by Sega.**
24. **To be this good takes AGES, To be this good takes SEGA.**
25. **These have been the cornerstone of an internal shift within Sega to appeal to a more Oriental and Western audience.**

Figure 6.5: Sample generated text for the *company descriptions* application in the *video games* domain.

domain-specific corpus. If we had extracted paraphrasal templates from a general company corpus, we would likely get many more such errors and they would be further from the truth ("Sega was founded in 1940 as a chain of pizza restaurants"). Similar problems would occur if we had trained our discourse model or taxonomy on a more general corpus.

Another interesting phenomenon in this text is the last paragraph. This consists mostly of extracted domain messages that are actually slogans used by Sega. It so happens that the Wikipedia article about Sega, which is part of the domain corpus, contains a long discussion about its advertising strategy and slogans. These slogans often contain the name *Sega*, which identifies them as being about the core entity and makes them candidate domain messages; they tend to not include other entities and to be very short, which increases their base preference score and makes them more likely to be selected; and they are similar in the vector space which helps clustering them in a single paragraph. If we could move sentence 8 to this paragraph, it could become a very interesting and relevant one. As it is, it likely makes little sense to most readers.

6.4 Evaluation

To evaluate our RDF applications we conducted a crowd-sourced human experiment. We picked 100 instances from each subject domain of each application, for a total of 400 (we picked the instances that had the most RDF triples in each subject domain). Then, we generated 5 versions for each instance:

1. A full-system version
2. A version that excludes the paraphrasal template mining component (so core messages only had the two manually-created templates, and domain messages only had a single template each)
3. A version that excludes the taxonomy (so there was no lexical choice, and we always used the original lexicalization of the template)
4. A version that excludes the discourse model (so discourse planning was done using only entity coherence scores)

5. A baseline version that is fully C2T instead of hybrid (i.e., only core messages were generated, without any extracted domain messages) and excludes the above three components

Using these 5 versions, we devised 9 questions for each instance. 5 questions where the annotator sees a single generated text (one from each version), and is asked to rate it along several criteria (see below); and 4 questions where the annotator sees two generated texts about the same instance (one is the full system version, and the other is one of the other four versions) and is asked which is better, again along several criteria. The questions were presented in random order, the systems were anonymized and each question was presented to three annotators. For the four comparison questions, we used the majority vote (and threw out results where there was total disagreement between the annotators, which happened 12% of the time for the baseline version, and $17-21\%$ of the time for the other variants). For the five rating questions, we used the average but ignored ratings which were more than 1 point away from both other ratings for the text (we would completely throw out results where that was true for all ratings, i.e. 1-3-5, but that never occurred in practice). This is in the spirit of the typical approach for binary crowd-sourced evaluations, where three annotators answer the same question and the majority answer is used where disagreement occurs, effectively throwing away the minority answer. This mechanism is designed to filter out noisy answers by annotators who are gaming the task or otherwise not earnestly answering the question, which is unfortunately always a risk with crowd-sourcing.

The purpose of this evaluation is to show that each of the three data-driven methods described in the first part of this thesis have a positive affect on text generated for an instance in a new subject domain, where adaptation to a new domain is completely automated (in part, by using those three methods). The baseline system is intended to show that beyond the contribution of our data-driven methods, the general approach of hybrid C2T-T2T generation embedded in the framework we proposed in the previous section is a useful way to generate text in previously unseen domains.

At each question, we asked the annotators to rate (or rank) the text (or pair of texts from two versions) along four criteria:

1. The *content* of the text, i.e. the information it contains and how relevant it is for understanding the topic

2. The *ordering* of the text, i.e. how well are sentences and paragraphs positioned with respect to one another (we specifically mentioned two sub-criteria here: more important sentences and paragraphs should in general appear earlier, and sentences that are related to each other should tend to appear together)

3. The *style* of the text, i.e. how close (or far) it is to a well-written description by a human

4. The overall satisfiability of the text, i.e. how happy are you in general with this text as a description of the person/company in question (we told the annotators that this does not necessarily have to be directly related to the other criteria)

We show the results of the comparison experiment (where we presented the annotators with two texts and asked them to tell us their preference along the criteria) in Table 6.3. The results in this table are for both applications and all four subject domains. Each comparison (e.g., "No Hybrid VS Full System" shows the breakdown of preference by annotators when they were shown texts generated by the two variants: how many (in percentage) preferred the baseline system (e.g. No Hybrid), how many preferred the full system, and how many thought they were equal. We also show the *winning difference* between the two systems, i.e. those who thought that the full system was better than the baseline minus those who thought the opposite, and we measure statistical significance on these differences. Statistically significant results are marked with a dagger.

Table 6.4 shows the breakdown of the *overall* criteria results into the different applications and domains. Note that it does not contain values for the *No Taxonomy* variant of the system for the biography domains. We did not use this system for evaluating biographies because in the biography domains, the generated taxonomies are very small and contain less than 10 synonyms each. The *No Taxonomy* variant, therefore, produces text that is virtually identical to that of the full system variant. We believe the reason for this drastic difference between the applications (the taxonomies of the company description application, in contrast, contain over a hundred synonyms each) is the way we create an expanded corpus

	Preference	Content	Ordering	Style	Overall
No Hybrid VS Full System	No Hybrid	20%	27%	24%	22%
	Equal	14%	11%	20%	14%
	Full System	66%	62%	56%	64%
	Full - baseline win diff.	**46%** †	**35%** †	**32%** †	**42%** †
No Paraphrases VS Full System	No Paraphrases	29%	33%	29%	30%
	Equal	31%	26%	28%	27%
	Full System	40%	41%	43%	43%
	Full - baseline win diff.	**11%** †	**8%** †	**14%** †	**13%** †
No Taxonomy VS Full System	No Taxonomy	29%	34%	36%	34%
	Equal	37%	29%	29%	27%
	Full System	34%	37%	35%	39%
	Full - baseline win diff.	**5%**	**3%**	-1%	**5%**
No Discourse Model VS Full System	No Discourse Model	33%	34%	32%	34%
	Equal	30%	22%	26%	23%
	Full System	37%	44%	42%	43%
	Full - baseline win diff.	**4%**	**10%** †	**10%**	**9%** †

Table 6.3: Preferences, with different criteria, given by the human annotators when presented with two versions - the full system VS each of the baseline versions. Statistically significant winning differences are marked with a dagger.

to extract the taxonomy from. Recall (from Chapter 5) that we extract the taxonomy from a corpus of Wikipedia articles that contains all articles which are mentioned at least twice in the domain corpus, which often means they are mentioned in two different documents of the corpus (in our experiments, these documents are themselves Wikipedia articles). From an informal review of the articles composing those corpora, we believe the difference is that what people (the core entities of the biography application) of the same profession have in common, at least when they are described in the domain corpus, are *entities*: specific places, institutions, and other people, while companies tend to have in common *types of entities* such as the type of their products (e.g. cars) and the titles of their executives (e.g. CEO), and *concepts* such as brand, headquarters and division. Specific entities tend to have less synonyms than types and concepts, and they are less likely to appear twice within a corpus, which is why we get a smaller taxonomy for biography domains. Obviously biographies also contain, at least conceptually, types (university, city, family member...) and concepts; it is just that they are not usually mentioned explicitly in the text. We think that is because those concepts tend to be more obvious to the reader. For example, text about a company will mention its product name but also the type of the product (not all companies make the same product); text about a person will mention the university he graduated from but not necessarily the fact that it is a university (because it is obvious).

The results of the ratings experiment (where we presented the annotators with a single text and asked them to rate it along the criteria) are shown in Table 6.5. As with the previous experiment, these results are for both applications and all four subject domains, and we show the breakdown of the *overall* results into the different applications and domains in Table 6.6. Ratings for baseline systems which are statistically significant with respect to the full system ratings are marked with a dagger.

6.4.1 Discussion

The most striking result of Table 6.3 is that the full system is overwhelmingly favored by annotators over the non-hybrid baseline, with a $32\% - 46\%$ lead in all categories. This result, more than anything, shows the value of our framework and the hybrid approach. The full system was particularly better than this baseline in *content*, which is generally

		Biography		Company Desc.		
				Auto-	Video	
	Preference	Politician	Model	motive	Games	Total
No Hybrid	No Hybrid	29%	27%	16%	18%	22%
VS	Equal	23%	15%	7%	11%	14%
Full System	Full	48%	58%	77%	71%	64%
	Full - baseline win diff.	**19%** †	**31%** †	**61%** †	**53%** †	**42%** †
No	No Paraphrases	31%	29%	29%	31%	30%
Paraphrases	Equal	29%	30%	25%	26%	27%
VS	Full	40%	41%	46%	43%	43%
Full System	Full - baseline win diff.	**9%**	**12%**	**17%** †	**12%** †	**13%** †
No	No Taxonomy	–	–	39%	29%	34%
Taxonomy	Equal	–	–	26%	29%	27%
VS	Full	–	–	35%	42%	39%
Full System	Full - baseline win diff.	–	–	-4%	**13%**	**5%**
No	No Discourse Model	36%	31%	37%	32%	34%
Discourse	Equal	21%	27%	22%	24%	23%
Model VS	Full	43%	42%	41%	44%	43%
Full System	Full - baseline win diff.	**7%**	**11%**	**4%**	**12%**	**9%** †

Table 6.4: Overall preferences, for each application and subject domain, given by the human annotators when presented with two versions - the full system VS each of the baseline versions. Statistically significant winning differences are marked with a dagger.

	Content	Ordering	Style	Overall
No Hybrid	3.46 †	3.19 †	3.29 †	3.3 †
No Paraphrases	3.59	3.19	3.33	3.38
No Taxonomy	3.6	3.29	3.46	3.46
No Discourse Model	3.58	3.14	3.34	3.38
Full	3.59	3.24	3.34	3.4

Table 6.5: Overall ratings, with different criteria, given by the human annotators to each version of the generation system. Ratings which are significantly different when compared to those of the full system are marked with a dagger.

	Biography		Company Description		
	Politician	Model	Automotive	Video Games	Total
No Hybrid	3.11	3.27	3.48	3.33	3.3 †
No Paraphrases	3.15	3.34	3.54	3.5	3.38
No Taxonomy	–	–	3.46	3.45	3.46
No Discourse Model	3.17	3.3	3.53	3.51	3.38
Full	3.24	3.36	3.5	3.51	3.4

Table 6.6: Overall ratings, for each application and subject domain, given by the human annotators to each version of the generation system. Ratings which are significantly different when compared to those of the full system are marked with a dagger.

expected since it by definition contains less content than the full system (it only generates the core messages); note, however, that this result suggests that the domain messages that are being extracted and selected are *relevant* and enhance the reader's satisfaction with the text. The baseline (which, in addition to not using extracted domain messages, also does not use the extracted paraphrasal templates, taxonomy, and discourse model) also loses heavily to the full system in *ordering* and *style*, as well as overall. In all criteria, the percentage of annotators who thought the texts were equally good was very low ($11\% - 20\%$), which means the difference was very visible.

While the effect of removing a single component is not as dramatic as removing all three in addition to the domain messages, it is clearly visible in the preferences of Table 6.3. All three reduced versions (*No Paraphrases*, *No Taxonomy* and *No Discourse Model*) lose to the full system in every criteria, often in double digits, with the sole exception of *style* for the *No Taxonomy* version, which was very slightly preferred to the full system. Note however that the results of the comparison with the *No Taxonomy* system are not statistically significant. The most meaningful component overall is the paraphrasal templates: the *No Paraphrases* version loses to the full system more heavily than either of the other two in *content*, *style* and *overall*. This result is not surprising since this component has the most dramatic effect on the text itself (as it changes the templates that are used to convey the information, enhances the diversity of the text and potentially merges together domain messages that are duplicates of each other), and it suggests that the paraphrasal templates we find are generally more satisfying than the default. Also not surprising is the fact that the *No Discourse Model* variant is the one that loses most on ordering among the three. While the difference is not as dramatic here, it is statistically significant and shows that our automatically extracted domain-specific discourse model helps in producing a more satisfying ordering of the text. Finally, the *No Taxonomy* variant performs most similarly to the full system, with more annotators marking it as equal than any other system, with the smallest differences (all at 5% or less) in all criteria, and with none of the results being statistically significant. This result is also somewhat expected, since the taxonomy is used only for lexical choice for a limited set of terms, which does not affect all texts across the board like the paraphrases or the discourse model. Instead, it applies relatively rare, surgical changes to specific terms where another synonymous term fits the context better. While the effect is predictably smaller than that of the other components, and the lack of significance makes it risky to draw conclusions, it seems to be mostly positive. We also take it to be a good sign that the two system variants that are most alike (that is, produce the most similar texts) are those with the smallest differences between them in terms of preference: it serves as a sort of experimental control group, and provides further evidence that the differences (for all variants) represent true differences in preference rather than arbitrary differences coming from the stochastic nature of the generation process.

Section 6.4.2 contains examples of actual output from this experiment, which materialize the differences between the variants of the system. It also includes some additional discussion and error analysis.

As seen in Table 6.4, there are significant differences between the (overall) preferences across different applications and domains. The difference in preference between the full system and the *No Hybrid* baseline are much more pronounced in company description domains than in biography domains (although the full system wins decisively across all domains). In both company description domains, more than 70% of the annotators preferred the output of the full system, while less than 20% preferred the baseline. We think this happens because of the difference in the magnitude of triples relating to people and to companies in DBPedia: people have on average 1.65 times the number of triples associated with them than companies (a company has on average 11.4 triples, while a person has on average 18.8 triples). What this means is that the core messages, which come from the triples, are more numerous and more satisfying for biographies than they are for company descriptions, which is why the baseline does better in biographies. If our hypothesis is correct, it is evidence that the hybrid approach is particularly suited to applications where the C2T component is handicapped by relatively impoverished data.

The differences for the three removed-component variants of the full system are more subtle, but still visible. Most strikingly, *video games* seems to be a better domain than *automotive* from our full system's point of view: overall satisfaction with *No Discourse Model* is almost as good as with the full system for *automotive*, and *No Taxonomy* actually does a little better than the full system (although again, these results are not significant and we should be careful not to draw conclusions from these numbers). Note, however, that this effect is reversed with *No Paraphrases*: the full system wins more often in *automotive* than it does in *video games* (although it clearly wins in both, with a double-digit lead), and unlike the other two variants, these are significant results. We are not sure why that is the case, but it is clear that different domains behave differently in subtle ways.

The differences in ratings given by the annotators to the different variants, shown in Table 6.5, are small in comparison with the direct comparison results. We expected this trend and view the ratings as a secondary evaluation since it is a more difficult and ambiguous task. Note, also, that while the differences in ratings are statistically significant between the full system and the *No Hybrid* baseline, they are not significant for any other pair. Nevertheless, the differences in ratings between the baseline and the full system are further evidence to the claim that user satisfaction is generally increased when the hybrid approach and the three components are used, even when texts are not shown side-by-side and annotators have to provide a score without any context. In fact, the *No Hybrid* baseline receives the lowest overall scores in all criteria, except in the ordering criteria where it receives the second-lowest score (and the *No Discourse Model* variant, which has exactly the same ordering algorithm but more moving elements because of the added domain messages, gets the lowest score). We will not discuss the results of the three variants, since they are very far from being statistically significant, and for the same reason we will not discuss the breakdown of ratings results across applications and domains in Table 6.6 (none of the results in that table are significant, except for the total ratings of the baseline). Overall, it seems that the rating task is much harder, in the sense that human ratings (without the context of direct comparison) have a large variance. Compare it with the similar study we performed for the discourse planning model in Chapter 4: there, we asked the annotators to provide a rating for two texts *together* with their comparison, and the results were both significant and more interesting. Note, however, that out-of-context human ratings are not completely arbitrary: for the two most different variants, namely the full system and the baseline, the annotators consistently and significantly gave the full system higher ratings.

6.4.2 Examples and Error Analysis

In this section we show a few examples of output that was scored by annotators in the first experiment of the evaluation of Chapter 6 (that is, the experiment where the annotators were shown two variants of the same output side by side, and were asked to choose which was better in four criteria). The first two examples are typical, and are intended to make concrete the differences among the versions that contributed to the results of Table 6.3. In both of those examples, the full system is scored higher than the variant. The following examples are non-typical, where the variant was seen as equal to or better than the full system: these serve as informal error analysis of some cases we have repeatedly in the results.

Figure 6.6 shows the output of the biography for politician *Marine Le Pen* of the full system and the non-hybrid baseline. As in the examples shown in Chapter 6, we mark sentences which were generated from extracted (T2T) domain messages in **bold**, and sentences which were generated from core messages but used an extracted paraphrasal template in *italics*. Sentences in unmarked typeface are those that were generated from core messages using one of the default templates. Underlined terms are those that were inserted by lexical choice using the taxonomy.

The main advantage of the full system is clear when looking at these two variants: it simply has much more content. The full system output contains six sentences (messages) more than the baseline output, and they are clearly relevant to the biography. The entire last paragraph, concerned with Le Pen's policies and positions - clearly an important part of a politician's biography - is missing from the baseline. These messages were extracted from the domain corpus, and show the power of the hybrid approach. In addition to the final paragraph, two extracted messages are included which are concerned with Le Pen's controversial history, and together with the RDF-derived message about her offices, they comprise a paragraph generally about her political background (with the addition of a sentence about her birth name). This is typical of the way that extracted messages contribute to the organization of the text in addition to the content: in the baseline version, the offices message is lumped together with messages about her background in general (alma mater, birth date, religion, partner etc). This demonstrates how the full system consistently

Full system output:

Marine Le Pen's birth places are Neuilly-sur-Seine and France. Marine Le Pen's residences are Millas, Hénin-Beaumont and Saint-Cloud.

The birth name of Marine Le Pen is Marion Anne Perrine Le Pen. Marine Le Pen's offices are Leader of the National Front, Municipal Councillor, Member of the European Parliament and Regional Councillor. **Marine Le Pen's ups and downs in the political arena follow those of the National Front at the time. Marine Le Pen stirred up controversy during the internal campaign.**

The homepage of Marine Le Pen is http://www.marinelepen.fr/.

The alma mater of Marine Le Pen is Panthéon-Assas University. Marine Le Pen's birth date was 1968-08-05. Marine Le Pen's religion is Catholic Church. Marine Le Pen's occupation is Politician. Marine Le Pen's partner is Louis Aliot.

Marine Le Pen regularly denounces sharp rises in energy prices which has "harmful consequences on the purchasing power of the working and middle-class families". Marine Le Pen denounces the current corporate tax as "a crying injustice". Marine Le Pen advocates to "vote for the abolition of the law enabling the regularization of the illegal immigrants". Marine Le Pen seeks to establish a moratorium on legal immigration.

Baseline output:

Marine Le Pen's party is National Front. Marine Le Pen's occupation is Politician. Marine Le Pen's homepage is http://www.marinelepen.fr/. Marine Le Pen's offices are Leader of the National Front, Municipal Councillor, Member of the European Parliament and Regional Councillor. Marine Le Pen's birth name is Marion Anne Perrine Le Pen. Marine Le Pen's religion is Catholic Church. Marine Le Pen's alma mater is Panthéon-Assas University. Marine Le Pen's birth date was 1968-08-05. Marine Le Pen's partner is Louis Aliot.

The birth places of Marine Le Pen are Neuilly-sur-Seine and France. Marine Le Pen's residences are Millas, H'enin-Beaumont and Saint-Cloud.

Figure 6.6: Output for *Marine Le Pen*.

outperforms the baseline in the *ordering* and *style* criteria, in addition to *content* and *overall*.

Figure 6.7 shows the output of the company descriptions for video game developer *Taito Corporation* of the full system and the no-paraphrases variant. Unlike the non-hybrid baseline, in this case the two outputs contain exactly the same information and have almost the same organization of the text (in the last paragraph, the ordering of messages is slightly different). The way in which the text is realized, however, is very different in the last paragraph. The full system realizes four of the six messages in that paragraph using extracted templates, including merging two messages into a single template in one case ("Taito Corporation was founded in 1953 by Michael Kogan", instead of the two separate sentences in the no-paraphrases baseline). The single-sentence messages also look better with their paraphrased versions: "Taito Corporation has around 662 employees" instead of the awkward-sounding "Taito Corporation's number of employees is 662".

Full system output:	No-paraphrases output:
The homepage of Taito Corporation is http://www.taito.com.	Taito Corporation's homepage is http://www.taito.com.
The products of Taito Corporation are Lufia, Bubble Bobble, Cooking Mama, Space Invaders, Chase H.Q., Gun Fight and Puzzle Bobble.	The products of Taito Corporation are Lufia, Bubble Bobble, Cooking Mama, Space Invaders, Chase H.Q., Gun Fight and Puzzle Bobble.
Taito Corporation was founded in 1953 by Michael Kogan. Taito Corporation has around 662 employees. Taito Corporation's location is Shibuya, Tokyo, Japan. **Taito Corporation currently has a subsidiary in Beijing, China.** *Taito Corporation was merged with "Square Enix".*	Taito Corporation's founding year is 1953. The founder of Taito Corporation is Michael Kogan. Taito Corporation's owner is Square Enix. **Taito Corporation currently has a subsidiary in Beijing, China.** Taito Corporation's location is Shibuya, Tokyo, Japan. Taito Corporation's number of employees is 662.

Figure 6.7: Output for *Taito Corporation*.

While the previous two examples illustrate the typical strengths of the full system, they are not seen in all cases. Figure 6.8 shows the output of the company descriptions for automotive producer *Škoda Auto* of the full system and the non-hybrid baseline. Although

Full system output:	Baseline output:
Škoda Auto is well known within the Czech Republic for its adverts that make use of repetitive beats. The location city of Škoda Auto is Mladá Boleslav. The founders of Škoda Auto are Václav Laurin and Václav Klement. *Škoda Auto is a part of Volkswagen Group. Škoda Auto was established in 1895.* The key persons of Škoda Auto are Winfried Vahland and Martin Winterkorn.	The key persons of Škoda Auto are Winfried Vahland and Martin Winterkorn. The founding year of Škoda Auto is 1895. The founders of Škoda Auto are Václav Laurin and Václav Klement. The parent company of Škoda Auto is Volkswagen Group. The location city of Škoda Auto is Mladá Boleslav.
Škoda Auto's product is Automobile. Škoda Auto's net income is 712,000,000 EUR. The revenue of Škoda Auto is 10,400,000,000 EUR.	Škoda Auto's homepage is http://skoda-auto.com.
Škoda Auto's homepage is http://skoda-auto.com.	The product of Škoda Auto is Automobile. Škoda Auto's revenue is 10,400,000,000 EUR. The net income of Škoda Auto is 712,000,000 EUR.

Figure 6.8: Output for *Škoda Auto.*

it is not a typical output in general (for most outputs, the full system clearly produces a better text than the baseline, as in Figure 6.6), it is typical of what we see in error analysis for the case where the full system does not beat the baseline. Essentially, for this particular entity, there was little in the way of domain messages to find in the corpus - the full system output in this case contains only one message which is not included in the baseline output (the first sentence of the first paragraph), and this message was not found by annotators to be particularly important to the text, so that the baseline version actually won in the *content* criterion. In addition, although the full system uses extracted paraphrases for two messages ("Škoda Auto is a part of Volkswagen Group" instead of "The parent company of Škoda Auto is Volkswagen Group", and "Škoda Auto was established in 1895" instead of "The founding year of Škoda Auto is 1895"), the annotators did not find them to be stylistically better than the default versions, deeming the two texts equal in the *style* criteria. Finally, the opinions on the *ordering* criteria were mixed. ordering decisions (which are determined by the discourse plan), unlike other decisions in the framework, are made stochastically, which makes it harder to analyze the results of a single output. One thing to keep in mind is that the choice of templates for different messages can have an effect on the ordering. For example, the choice of template in the sentence "Škoda Auto was established in 1895" (instead of the one using the term "founding year") probably contributed to having that message separated from the message about the founders of the company, because of a lower vector similarity. A majority of annotators (two out of three) agreed that the baseline wins overall in this case.

Full system output:	No-discourse output:
Holly Madison's occupations are Television presenter and Model. Holly Madison's alma mater is Portland State University. Holly Madison's partner is Hugh Hefner. The birth date of Holly Madison was 1979-12-23. *Holly Madison started modeling in 1998.* em Holly Madison began life as "Holly Sue Cullen". The homepage of Holly Madison is http://www.hollymadison.com. The birth place of Holly Madison is Astoria, Oregon. Later, the residence of Holly Madison is Las Vegas Valley. Holly Madison's height is 1.702.	Holly Madison's birth date was 1979-12-23. *Holly Madison started modeling in 1998. Holly Madison began life as "Holly Sue Cullen".* Holly Madison's homepage is http://www.hollymadison.com. The occupations of Holly Madison are Television presenter and Model. The alma mater of Holly Madison is Portland State University. Holly Madison's partner is Hugh Hefner. The residence of Holly Madison is Las Vegas Valley. The birth place of Holly Madison is Astoria, Oregon. Holly Madison's height is 1.702.

Figure 6.9: Output for *Holly Madison*.

Finally, Figure 6.9 shows a typical negative example for the discourse model component. The text shown is the output of the biography for model *Holly Madison* of the full system and the no-discourse variant. Two of the three annotators who saw these outputs thought they were equal in terms of ordering (and everything else, for that matter). It is not difficult to see why: since the paragraphs in this example are very short, the internal ordering (which is determined by the discourse model) makes little difference. In terms of discourse relations, only one is made explicit with a connective: the temporal relation in the second-to-last paragraph. The use of this connective ("later") is fairly awkward in this context, but this in itself did not seem to bother the annotators (but it also did not invoke a positive reaction). In all cases we have looked at where the no-discourse variant beat the full system, or was deemed equal, the paragraphs tend to be short and few (if any) discourse relations are made explicit.

6.5 Conclusion

In this chapter, we introduced and defined two GEMS applications - biography and company descriptions - belonging to a sub-family of GEMS which we call RDF applications. For each of these applications, we (automatically) adapted two subject domains using a domain corpus as described in the previous chapter, and used our framework to generate texts (biographies and company descriptions) in each of the four domains.

In a crowd-sourced human evaluation, we showed that our full framework, using hybrid generation and the three data-driven bottleneck solutions we presented in this thesis, vastly outperforms a C2T baseline with access to the same input data. In addition, we showed that at least two of the three solutions we introduced in the first part of this thesis significantly increase overall user satisfaction with the generated text, and specifically increase it in the criteria we expect it to (i.e., the paraphrasal templates help with content and style while the discourse model helps with ordering). While results were less conclusive for the taxonomy as a solution for lexical choice, it seems to have a positive effect.

Within this thesis, the main purpose of this chapter is to provide examples of implementing traditional generation applications using the GEMS framework and to evaluate the contribution of the three data-driven solutions we proposed earlier in the thesis and of the framework itself. In general, however, we believe that this special sub-family, namely RDF applications, in conjunction with our GEMS framework, can prove a useful tool for almost fully automated generation of the description of virtually any RDF entity (or, at the very least, a powerful starting point). The automated part of the definitions we provide in this chapter can be used as-is for creating applications for describing products, institutions, sports teams, animals, video games, and any number of other entity types. It is a step towards truly generic generation systems that can work for any application and in any domain.

While the RDF applications family is large and diverse, it has its limits. An obvious one is that for many applications, there is no RDF or similar source from which the core data can be retrieved in an easy-to-digest format; another is that many applications do not fit into the theme of describing a well defined entity. The GEMS framework, however, can handle such applications. The next chapter describes an important application which does not conform to these constraints and shows how it fits into the GEMS framework.

Chapter 7

Machine Learning Prediction Justification

Machine learning systems are increasingly used by humans to assist them in decision making. The systems produce predictions or recommendations which are then considered by a human decision-maker, and it is important that the prediction can be *justified*: the user will want to understand why the system produced its recommendation before making a decision.

For the rule-based expert systems that were prevalent in past decades, it is often enough to *explain* how the system reached its conclusion. The human user will be able to understand the set of rules governing the prediction, and given the proper information about the particular situation leading to a specific prediction (the relevant states of the data and the chain of rules that led to the final decision), will be able to make up her mind about the prediction's validity. This is called the "glass box" model,[1] in contrast to the "black box" model where explanation is not given.

Recently, machine learning techniques have all but replaced rule-based methods, often resulting in increased accuracy and an ability to handle more complex problems. In contrast to rule-based expert systems, justifying the predictions of machine learning models is not a straightforward task: it is no longer the case that *explaining* how a prediction was reached automatically *justifies* it to the user. Due to the complex, quantitative and unintuitive

[1]It is also sometimes called the "white box" model

nature of many machine learning models,[2] it is unreasonable to expect that users who are not machine learning experts, even if they are experts in the domain of the prediction, will understand how the model works, regardless of how transparently it is presented. In other words, the glass box model is no longer useful for most users.

A black box with no justification at all, however, is even worse. We propose what might be called a "self-explaining box" model, where Natural Language Generation (NLG) is used to produce simple, short, qualitative and intuitive justifications for machine learning predictions, relying on the domain knowledge embodied in the features.

This chapter describes our approach to Machine Learning Prediction Justification generation, relying on the GEMS generation framework described in Chapter 5. We describe the core message selection algorithm and the core types and templates, and present an evaluation of this system in multiple subject domains. We have made this work available as a library called PreJu, which is described in Section 7.5. PreJu allows engineers to easily adapt this GEMS application to new prediction subject domains and features both a stand-alone mode where configuration and input are given externally in XML files and an API which allows programmatic usage with existing ML libraries.

7.1 Related Work

Related work for producing justifications (or more commonly *explanations*, which are presumed to be justifying) come from multiple fields. Historically, explanations first appeared in the context of rule-based expert systems, and were mostly treated as a systems design task (i.e., the task of designing a system capable of producing explanations and drill-down into its decisions). In some fields - especially the medical - probabilistic decision-making systems are still called expert systems and are treated as a continuation of that line of research, and explanation of these systems is treated somewhat similarly. In the machine learning literature, explanation is often understood in the sense of visualizing the state of a model to conveniently show how a decision was reached, or in the sense of explaining the internals of complex models in simpler terms. In the 2000's justification has also been of

[2]One notable exception is a shallow decision tree

particular importance in the field of recommender systems, where a ML or other probabilistic system needs to justify its recommendation to many users. It is there that we find a concept of justification that most resembles our work. We will discuss related work in each one of these fields in a separate subsection, following a survey of more general work below. Finally, in recent years, explanation and justification have also been explored in the ubiquitous computing community with the rise of context-aware applications (Tullio et al., 2007; Lim and Dey, 2010; Lim, 2012). We will not explore that literature beyond this mention, as it is concerned with a constantly changing interactive environment and is thus quite different from what we do.

Explanation has been shown to be important for user acceptance and satisfaction in a number of studies. In one early study, physicians rated the ability to explain decisions as the most highly desirable feature of a decision-assisting system (Teach and Shortliffe, 1981). (Ye and Johnson, 1995) experimented with three types of explanations for an expert system - trace, justification and strategy - and found that explanations in general and justifications in particular make the generated advice more acceptable to users.

There has also been some theoretical work on explanation (note that the terms *explanation* and *justification* are sometimes used interchangeably in the literature, and theoretical work is often in fact concerned with justification). In the context of expert systems, Johnson and Johnson (1993) presented a short survey of accounts of explanation in philosophy, psychology and cognitive science and found that they fall into three categories: associations between antecedent and consequent; contrasts and differences; and causal mechanisms. Based on that insight and a study of contemporary systems with explanation facilities, they developed a theory of task-based explanation. In the context of recommender systems, Yetim (2008) proposed a framework of justification-type explanation, which defines the components of a justification based on Toulmin (1958)'s model of argument and a classification of justification types based on Habermas (1984)'s discourse theory.

Corfield (2010) aims to philosophically formalize justifications for the accuracy of machine learning models by classifying them into four types of reasonings. While out of scope for our work (as it deals with the justification of *models*, and not of individual predictions), it nonetheless provides an interesting view on the sort of reasoning that underlies the trust that machine learning experts have in the predictions of their classifiers.

7.1.1 Expert Systems

The need for explaining the decisions of expert systems has been discussed as early as the 1970's (Shortliffe and Buchanan, 1975) in the context of MYCIN, a medical diagnosis system. XPLAIN (Swartout, 1983) was an early framework for creating expert systems with explanation capabilities, and was one of the first to stress the importance of explanations that are not merely traces, but also contain justifications. EES (Swartout et al., 1991) is a later example of such a framework, again with a focus on justifications. Both were exclusively for rule-based systems and relied on a domain-specific taxonomic knowledge base and a separate logic/strategic knowledge base. Barzilay et al. (1998) further separated the knowledge into three layers, adding the *communication* layer to the previously described *domain* and *strategic* layers. Separating the communication layer from the rest of the system was intended to allow a communication expert (such as a NLG expert) to create solutions that were independent of the specific system and domain. Another relatively late example of a framework for rule-based expert system explanation is (El-Beltagy et al., 1999). Their framework is agent-based and focused on systems which employ dialog explanations.

The expert systems literature does not end with the rule-based systems of the 80's and 90's. In some domains probabilistic decision-making systems, often based on Bayesian networks (BN), are still referred to as expert systems and regarded as successors of earlier rule-based systems. The (scarce) work on explanation for these BN systems self-describes as expert systems explanation. Lacave and Díez (2002) present a survey of methods of explanation for Bayesian networks and an excellent analysis of the methods in terms of several properties of explanation. Of particular interest is their classification of the focus of explanation into an explanation of the *reasoning*, the *model*, and the *evidence* for the decision. The first (explaining the reasoning) is particularly suited to rule-based systems, including those that use rules probabilistically. An explanation of the model is static and useful in an early introduction-to-the-system stage. Our work would fall into the third class, also called *abductive reasoning*, as we produce a justification by explaining the evidence.

Druzdzel (1996) notes that probability theory is not a good model for human reasoning, and that probabilistic justifications are therefore not desirable. Instead, he proposes a mapping of probability ranges to verbal qualifiers ("unlikely", "very likely", "certain" etc) for expressing predictions along with a simple description of dependencies among nodes in the network, and a description of evidence causality ("decreased likelihood of excessive oil consumption is evidence against worn piston rings"). This work, while using only canned text, failing to define the roles of different types of evidence, relying exclusively on a manually-defined knowledge base, and providing no evaluation of its proposed solution, attempts to solve problems similar to those we focus on. Most work on explanation in Bayesian networks has been within the narrow context of a particular system, and relies on producing canned text showing the actual posterior probabilities of each node and providing no explanation for what the nodes themselves symbolize, assuming that their names are enough (individual nodes are often symptoms, in the medical domain, or simple physical evidence, e.g. "valve open", in other domains) (Druzdzel and Henrion, 1990; Haddawy et al., 1997; Yap et al., 2008; Helldin and Riveiro, 2009).

7.1.2 Machine Learning

In the machine learning literature, work on explanation has often focused on producing visualizations of the prediction in order to assist machine learning experts in evaluating the correctness of the model. One very common visualization technique is *nomograms*. It was first applied to logistic regression models by Lubsen et al. (1978), and later to Naive Bayes (Možina et al., 2004), SVM (Jakulin et al., 2005) and other models. An example of a nomogram can be seen in Figure 7.1.[3] Szafron et al. (2003) proposed a more detailed visualization-based explanation framework for Naive Bayes classifiers.

Other work has focused on interpreting the predictions of specific complex models, often by proposing to isolate the contributions of individual features to the prediction. Such proposals were made for Bayesian networks (Suermondt, 1992), multi-layer Perceptrons (Feraud and Clerot, 2002), SVMs (Carrizosa et al., 2006), RBF networks (Robnik-Šikonja et al., 2011) and general hierarchical networks (Landecker et al., 2013). Martens et al. (2008)

[3]Image taken from a public forum post by Aleks Jakulin at `http://www.stat.columbia.edu/~cook/movabletype/archives/2006/05/`

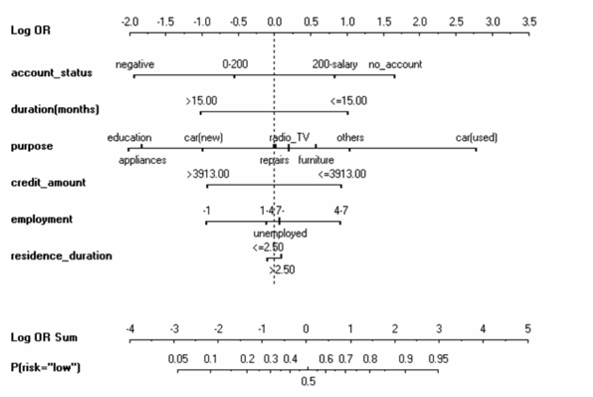


Figure 7.1: Example nomogram for a logistic regression model of credit risk.

proposed to interpret the predictions of an SVM classifier by extracting logical rules of the form $\{feature_1 = value_1 \wedge \cdots \wedge feature_k = value_k\} \rightarrow class$ for a small maximum number of features k. This method was later used for CRFs as well (Seth and Bhattacharyya, 2011).

In addition to model-specific methods, there have been a few suggestions for model-agnostic frameworks. Robnik-Sikonja and Kononenko (2008) propose measuring the effect of an individual feature on an unknown classifier's prediction by checking what the prediction would have been if that feature value was absent and comparing the two using various distance measures. The effects are then displayed visually to explain the main contributors towards a prediction or to compare the effect of the feature in various models. This method was extended to include regression models in (Kononenko et al., 2013). Baehrens et al. (2010) describe an alternative approach using *explanation vectors* (class probability gradients) which highlight the effect of the most important features.

The work presented in (Robnik-Sikonja and Kononenko, 2008) is also interesting because of its classification of explanation types. The authors differentiate between two levels of explanation: the *domain* level and the *model* level. They define domain level explanation as an explanation of the true causality between the independent variables and the dependent variables, which is not knowable in realistic settings, while a model level explanation is an explanation of the way that the model arrives at a prediction. Given this definition, the correctness of the prediction and the correctness of the model-level explanation are orthogonal. In better models (those which model the domain better, approximated as those that have higher prediction accuracies), correct model-level explanations will tend towards being correct domain-level explanations as well. This is the only work we know of that explicitly acknowledges that good explanations (correct model-level explanations) are not necessarily correct explanations (correct domain-level explanations). The term *domain level* is somewhat unfortunate, because their work is still geared towards machine learning experts and focuses on the model alone, containing no links to actual domain entities. In our language, we would rather talk about domain explanation as an explanation of the features in domain terms, tying feature values to real-world entities.

7.1.3 Recommender Systems

Recommender systems are online services that serve a large number of users and provide individualized recommendations for media or products. It is usually desirable to produce a short and intuitive justification to help the users decide whether to follow the recommendation or not.

Herlocker et al. (2000) conducted an experiment measuring user satisfaction with a variety of justification types for a collaborative filtering (neighbor-based) movie recommendation system. They found that the most satisfying were simple and conclusive methods, such as simply stating the neighbors' ratings or showing a histogram of them, focusing on single strong features like a favorite actor, and even simply stating the past performance of the system. Surprisingly, justification types incorporating ML concepts such as model confidence and types showing complex justifications such as a full neighbor graph scored *lower than the baseline black box.* In another experiment presented by the authors, 86% of users stated they wanted the form of justification they were shown added to the system. Other studies from the early 2000's have also shown that users are overwhelmingly more satisfied with systems that contain some form of justification (Sinha and Swearingen, 2002).

Tintarev and Masthoff (2007) presented a survey of explanation in recommender systems and identify seven distinct benefits of explanation: transparency, scrutability (enables the user to point out errors), trust, persuasiveness (user more likely to follow system decisions), effectiveness (helps the user make better decisions), efficiency (helps the user make decisions faster), and satisfaction. They note that existing work at the time of their survey have overwhelmingly focused on evaluating trust, persuasiveness, effectiveness and efficiency. This highlights the inconsistency in the literature when using the terms explanation and justification: the four benefits that have been studied most in the context of explanation are in fact benefits of any kind of justification, while the first two - transparency and scrutability - are benefits only of explanation.

Symeonidis et al. (2009) presented a style of justification that focused on a single most important feature with some additional context (the user's past history with regards to that feature). A user study showed that this justification style was significantly more satisfying to users than previous methods. Papadimitriou et al. (2012) defined a classification of recommender system explanations (which they call *justifiable recommendations*) into three types: those based on previous items chosen by the user, those based on choices of similar users, and those based on features. They also defined a hybrid type which combines two or more of the above, and following a user study concluded that feature-based explanations were the best of the three core types, and that hybrid explanations were best overall. Bilgic and Mooney (2005) noted that previous studies have often evaluated the persuasiveness of the justification and not its justifiability (which they tied with post-consumption satisfaction). Their experiments showed that for justifiability, feature-based justifications were superior to neighbor-based and user-history-based ones.

7.2 Core Message Selection: Justification Narratives

In this section we motivate and describe the core message selection algorithm we use for the ML Prediction Justification implementation of the GEMS generation framework. Parts of this work are described in Biran and McKeown (2014).

In machine learning, unlike rule-based or knowledge-based expert systems, it is not reasonable to expect non-experts to understand the details of how a prediction was made. It is still important, however, that they understand the variables affecting the current prediction enough to satisfy the question of justification. It has been shown that evidence-based causal models of justification are often more satisfactory to users than full glass box models (Herlocker et al., 2000), and that replacing numeric values with qualifying linguistic expressions (high, strong, etc) also enhances satisfaction (Druzdzel, 1996; Herlocker et al., 2000; Lacave and Díez, 2002). The independent variables used in machine learning models often correspond to real-world evidence that non-experts understand well, and a justification for a prediction can rely on these variables, their importance to the model, their effect on the prediction, and their interactions.

A robust method of automatically generating prediction justification for non-experts, then, should focus on selecting the most important pieces of evidence for a particular prediction and on analyzing and presenting their roles in the prediction. The selected evidence should be presented to the user in a way that is invariant across different models and readily understandable by a non-expert.

We have created a framework for producing *justification narratives* from machine learning predictions (Biran and McKeown, 2014). Each narrative is essentially composed of a subset of the relevant features, where each feature has a discrete *role*. The framework enables us to more easily decide what an appropriate feature subset looks like and to present it in a way that is more qualitative than quantitative. This plays a somewhat similar role as the Signal Analysis and Data Interpretation modules described in Reiter (2007) for data-to-text generation systems.

7.2.1 Narrative Roles

The first step in producing a narrative is determining the role of each feature. Following some of the examples found in the machine learning explanation literature (Robnik-Sikonja and Kononenko, 2008; Carrizosa et al., 2006; Yap et al., 2008), we identify two central concepts that can be defined for many different types of classifiers, namely the *effect* of a feature on the prediction (its actual contribution towards or against predicting the predicted class) and the feature's *importance* in the model (the expected effect of the feature for a prediction of the particular class, which is not dependent on its value in the current prediction).

In our work, we focus on linear classifiers which utilize *linear discriminant functions*. This family contains many of the most commonly-used machine learning classifiers, including Logistic Regression, Perceptrons and Linear SVMs, and log-linear models such as Naive

Bayes can also be formulated in a way that fits this framework. We leave the task of fitting more complex models (such as Kernel SVMs and Neural Networks) to future work, making note of the fact that there has been some work on determining the effect of features in various complex models (Carrizosa et al., 2006; Yap et al., 2008) as well as suggestions for model-agnostic definitions (Robnik-Sikonja and Kononenko, 2008; Kononenko et al., 2013).

A linear discriminant function for data instance x in the general multi-class linear classifier is

$$f(y, x) = \sum_i \theta_{yi} x_i$$

Where each x_i is a feature value and the coefficients θ_{yi} have been learned from the training data for each class y using some learning algorithm, and may include an intercept.[4] [5] The classifier predicts the class of the instance as the one that maximizes the predictor function, possibly through a monotonic non-linear distortion function φ:

$$\begin{aligned}\hat{y} &= \underset{y}{\operatorname{argmax}}\, \varphi\left(f(y, x)\right) \\ &= \underset{y}{\operatorname{argmax}}\, f(y, x)\end{aligned}$$

In some models, the function φ used to determine the class has a probabilistic interpretation (e.g., the logistic function in Logistic Regression). This is not a requirement for our approach, which is valid as long as the argmax formulation holds. We are also indifferent to regularization methods that may be used in training the models, since we rely only on the final coefficients and feature values in defining the role of each feature.

The linear *effect* of a feature i towards or against predicting class y for a data instance is the product of the feature's coefficient and its value in the instance:

[4]This is a generalization of the binary classifier case, where there is only one coefficient vector θ for the positive class, and we can think of the negative class implicitly having a coefficient vector which is the negation of the positive class coefficients

[5]There is also an alternative formulation where multi-class predictions are obtained though voting among a set of binary classifiers. We do not attempt to handle such voted constructions

Importance \ Effect	High positive	Low	High negative
High positive	Normal evidence	Missing evidence	Contrarian counter-evidence
Low	Exceptional evidence	Negligible	Exceptional counter-evidence
High negative	Contrarian evidence	Missing counter-evidence	Normal counter-evidence

Table 7.1: Narrative roles assignment for the range of feature effect and importance

$$\text{Effect}_{yi} = \theta_{yi} x_i$$

While its *importance*, the expected effect for predictions of the class, can be estimated using the mean feature value for the class (X^y is the set of all instances in the training set with class y):

$$\text{Importance}_{yi} = \theta_{yi} \frac{\sum\limits_{x \in X^y} x_i}{|X^y|}$$

Using these two concepts, we assign a *narrative role* for each feature. Narrative roles are assigned based on the sign and magnitude of the importance and effect of a feature towards the predicted class. They represent semantically clear concepts that non-experts readily understand, and are rooted in the true details of the prediction. Table 7.1 shows the roles for all possible combinations. The role explanations and a few alternatives for defining "high" and "low" in practice are described below.

7.2.1.1 Narrative Role Descriptions

Narrative roles can be broadly divided into three groups: evidence roles, missing evidence roles, and counter-evidence roles.

Features having evidence roles are those that had a positive contribution toward making the prediction. **Normal evidence** is evidence that is expected to be present in many instances predicted to be in the class: with high importance, high effect is not surprising. **Exceptional evidence** is evidence that is not usually expected to be present. With low importance, high effect means that the feature value is surprisingly high. **Contrarian evidence** is strongly unexpected, since the effect has the opposite sign than expected. Note that contrarian evidence (and contrarian counter-evidence) is only possible for features that may have negative values, and may not appear in many real-world applications.

Features that have **Missing evidence** as a role are important features that were expected to contribute positively, but were weak for the particular instance. If a normally high-effect feature had a low effect in the prediction, that is something we want to include in the narrative because it means that the prediction was uncharacteristically made without the important effect of that feature (depending on the prediction domain, this may tell the human that this prediction is likely to be wrong. It may also signal the opposite - that the evidence for the prediction is exceptionally sound). Similarly, **missing counter-evidence** is given to features that were expected to contribute negatively but did not.

Finally, there are counter-evidence roles. Features having these roles contributed against the prediction, although they were not strong enough to prevent it. **Normal counter-evidence** is expected: it is normal for the feature to have a high negative effect, even if the positive effects from other features ultimately overpower it. **Exceptional counter-evidence** is not expected. Finally, **contrarian counter-evidence** means that a feature we normally expect to contribute positively contributes negatively instead.

7.2.1.2 Quantizing Effect and Importance

Table 7.1 assumes that we have quantized the importance and effect values of the features into *high positive*, *high negative* and *low*. We use the following method to quantize the values.

To quantize importance, we first separate the features with positive and negative importances. For each one of these groups, we rank the features in order of their (absolute) importance values, and starting with the highest, we iterate through them and assign high importance to each feature we encounter until the ratio of the total importance value of the high importance features and the total importance value of all features is equal to or greater than a tunable parameter τ. In our experiments in this chapter, we set $\tau = 75\%$. In other words, the smallest set of features that together have an importance equal to 75%

of the total importance of the features (on the positive side, and separately on the negative side) are determined to have high importance, while all others have a low importance.

Once importance is quantized, we find a threshold (one for the positives, one for the negatives) midway between the highest importance of a low importance feature and the lowest importance of a high importance feature (essentially maximizing the margin between the two groups). This threshold is used to determine the effect of features for each instance: the feature effect is high if it is on the high side of the threshold, and low otherwise.

7.2.2 Key Feature Selection

Once the roles are determined, we use them to select the set of *key features* for the prediction. These are the features that represent the most important evidence which a user will need to see in order to make up his mind about the prediction.

The appropriate way to select the key features depends on the application. As a general rule, we would want to keep the expected number of selected features to a reasonable size that allows generating a justification which a human can reasonably consume. If the feature space is small and/or sparse, it may make sense to select all features that have any role other than negligible. For example, an e-mail spam classifier which relies on ngram features might take this approach and present the important ngrams which contributed to the decision ("the text of the email contains the terms 'Nigerian Prince' and 'bank account' but not the terms 'joke' or 'Facebook post', which makes it spam").

Some consumer-facing applications (e.g., recommender systems) may want to select a fixed number of features to show as evidence to the user, and may want to constrain their roles (only showing the *evidence* roles in the first column of table 7.1, for example). A music recommendation stream, for example, will want to show quick and simple justifications for each suggestion and may choose only the top two evidence features ("You should listen to this band because it plays Heavy Metal and is based in New York").

In other cases, we may want to choose the top features from each role group (where the top is determined by a ranking of effect for evidence and counter evidence, and by importance for missing evidence). For example, a classifier used by medical professionals to assess the condition of a patient given symptoms might choose this approach to provide a more complete view of the classification ("The patient's condition appears to be influenza. The main evidence symptoms are a mild fever, sore throat and nausea. Missing evidence symptoms include cough and upset stomach. The strongest counter-evidence symptom is a mild skin rash which is more associated with other conditions");

The above are all examples of key feature selection *strategies*. In PreJu, the justification library we describe in section 7.5, we provide a range of strategies, and it is up to the user to select (in configuration) which is the appropriate one to use. A user with programming knowledge can also write and use her own strategies. The default strategy, and the one we use in the evaluation at the end of this chapter, is the one that selects all non-negligible features as key features.

It is important to note that key feature selection is completely unrelated to *feature selection* in the Machine Learning sense. Feature selection techniques operate at the model level, selecting the most predictive and least repetitive features overall so that other features can be excluded from model training. Key feature selection operates at the prediction level on a model that has already been trained, and is concerned with features that are key for the particular prediction only.

7.3 GEMS Definition

In this section we describe the definition of the Justification Prediction GEMS application in terms of the framework of Chapter 5. We shall refer to the this application as $\mathcal{J}$.

$CoreEnt^{\mathcal{J}}$, the function which defines the core entities of the domain, is simple in this application: for each domain (i.e. a classifier), the core entities are the features participating in the classifier.

$CoreMes^{\mathcal{J}}$, the function which defines the core messages of each instance, implements the justification narrative approach described in the previous section. The justification narrative of the instance determines the key features $K^{\odot}$ (as in the previous two chapters, we use $\odot$ as the symbol for an instance) and their roles in the prediction $R^{\odot}$. For each key feature k_i in $K^{\odot}$ we build a message m_i with STT $\tau^{m_i} = rolestt(r_i)$ and a single-entity set $E^{m_i} = \{k_i\}$. The $rolestt(r_i)$ mapping chooses the relevant STT from Table 7.3 based on the role of the key feature. In other words, we create one message for each key feature, describing its role in the prediction. In addition, we create a single message describing the result of the prediction and a single message describing the main evidence from the key feature with the highest effect, using the STTs in Table 7.2.

STT Description	Entities+Relations	Templates
Prediction	a (Model)	a predicts that c is b
	b (Value)	according to a , c is b
	c (Thing)	c is b according to a
	a madePrediction b a predictsAbout c	c , given by a , is b
Main Evidence	a (Feature)	the main evidence for the prediction is in a
		the prediction was made mostly because of the evidence in a
		the evidence in a constitutes the main reason for the prediction
		a constitutes the main evidence for the prediction

Table 7.2: STTs for other core messages in the prediction justification application

BasePref$^{\mathcal{J}}$, the function which defines the base preference scores of the messages, is also simple in this application: all domain messages have a base preference of -0.5 while core messages have a preference of 1.

DiscRel$^{\mathcal{J}}$ is the function which defines the possible discourse relations between messages. Possible relations are defined for pairs of core messages based on their STTs, as shown in Table 7.4.

7.4 Evaluation

In contrast to the previous chapter, where we evaluated the benefits of the GEMS framework in general and of the various methods we proposed in this thesis, here we are concerned with evaluating the justification application. In particular, we want to see if our generated justifications really help the end-user in correctly deciding whether or not the prediction is correct.

STT Description	Entities+Relations	Templates
Expected Evidence	a (Feature)	there is strong evidence in a, which is normal
		normal strong evidence exists in a
		a exhibits normal strong evidence
Expected Counter-Evidence	a (Feature)	there is strong counter-evidence in a, which is normal
		normal strong counter-evidence exists in a
		a exhibits normal strong counter-evidence
Missing Evidence	a (Feature)	normally , there would be strong evidence in a, but it is missing in this case
		although normally there would be strongevidence in a, in this case it is missing
Missing Counter-Evidence	a (Feature)	normally , there would be strong counter-evidence in a, but it is missing in this case
		although normally there would be strong counter-evidence in a, in this case it is missing
Exceptional Evidence	a (Feature)	normally, a does not provide strong evidence, but in this case it does
		a provides unusually strong evidence
		unusually strong evidence exists in a
		there is strong evidence in a, which is unusual
Exceptional Counter-Evidence	a (Feature)	normally, a does not provide strong counter-evidence, but in this case it does
		a provides unusually strong counter-evidence
		unusually strong counter-evidence exists in a
		there is strong counter-evidence in a, which is unusual
Contrarian Evidence	a (Feature)	normally a provides counter-evidence, but in this case it does the opposite
		a exhibits strong evidence, although usually it provides counter-evidence
Contrarian Counter-Evidence	a (Feature)	normally a provides evidence, but in this case it does the opposite
		a exhibits strong counter-evidence, although usually it provides evidence

Table 7.3: STTs for feature-role core messages in the prediction justification application

First Message STT	Second Message STT	Possible Discourse Relations
Main Evidence	Prediction	contingency $\rightarrow$
Expected Evidence	Expected Counter-Evidence	comparison $\leftrightarrow$
Expected Evidence	Missing Evidence	comparison $\leftrightarrow$
Expected Evidence	Exceptional Counter-Evidence	comparison $\leftrightarrow$
Expected Evidence	Contrarian Counter-Evidence	comparison $\leftrightarrow$
Missing Counter-Evidence	Expected Counter-Evidence	comparison $\leftrightarrow$
Missing Counter-Evidence	Missing Evidence	comparison $\leftrightarrow$
Missing Counter-Evidence	Exceptional Counter-Evidence	comparison $\leftrightarrow$
Missing Counter-Evidence	Contrarian Counter-Evidence	comparison $\leftrightarrow$
Exceptional Evidence	Expected Counter-Evidence	comparison $\leftrightarrow$
Exceptional Evidence	Missing Evidence	comparison $\leftrightarrow$
Exceptional Evidence	Exceptional Counter-Evidence	comparison $\leftrightarrow$
Exceptional Evidence	Contrarian Counter-Evidence	comparison $\leftrightarrow$
Contrarian Evidence	Expected Counter-Evidence	comparison $\leftrightarrow$
Contrarian Evidence	Missing Evidence	comparison $\leftrightarrow$
Contrarian Evidence	Exceptional Counter-Evidence	comparison $\leftrightarrow$
Contrarian Evidence	Contrarian Counter-Evidence	comparison $\leftrightarrow$
Expected Evidence	Missing Counter-Evidence	expansion $\leftrightarrow$
Expected Evidence	Exceptional Evidence	expansion $\leftrightarrow$
Expected Evidence	Contrarian Evidence	expansion $\leftrightarrow$
Missing Counter-Evidence	Exceptional Evidence	expansion $\leftrightarrow$
Missing Counter-Evidence	Contrarian Evidence	expansion $\leftrightarrow$
Exceptional Evidence	Contrarian Evidence	expansion $\leftrightarrow$
Expected Counter-Evidence	Missing Evidence	expansion $\leftrightarrow$
Expected Counter-Evidence	Exceptional Counter-Evidence	expansion $\leftrightarrow$
Expected Counter-Evidence	Contrarian Counter-Evidence	expansion $\leftrightarrow$
Missing Evidence	Exceptional Counter-Evidence	expansion $\leftrightarrow$
Missing Evidence	Contrarian Counter-Evidence	expansion $\leftrightarrow$
Exceptional Counter-Evidence	Contrarian Counter-Evidence	expansion $\leftrightarrow$

Table 7.4: Possible discourse relations between pairs of messages given their STTs. The arrows show the possible directions of the relations.

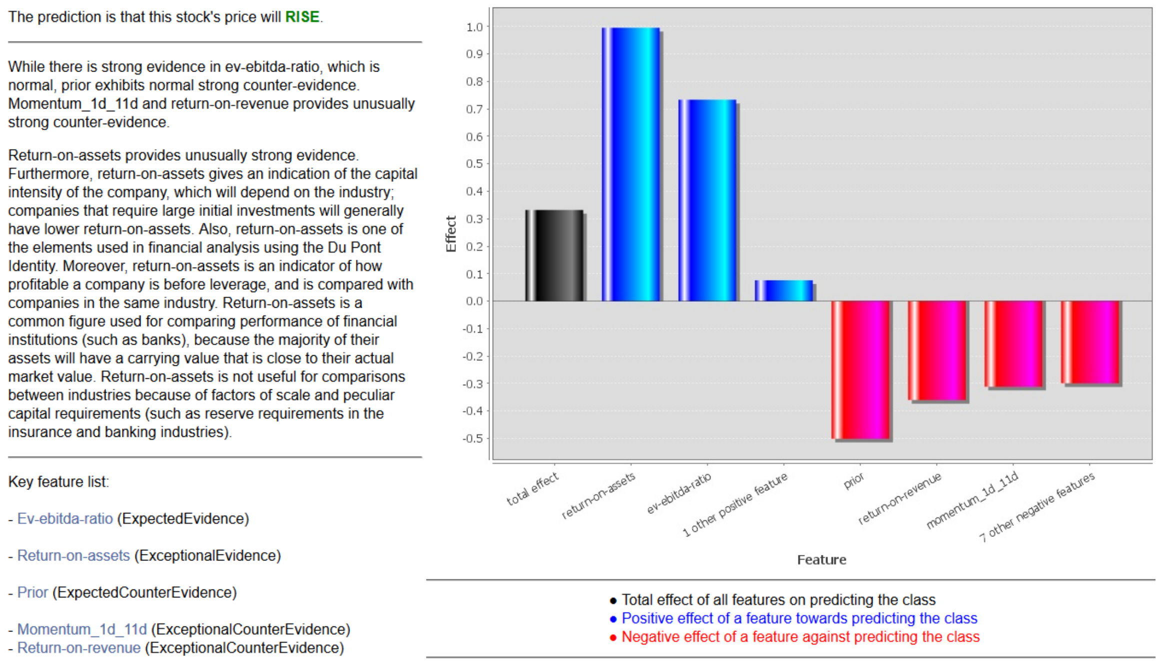
For this purpose, we devised a task based evaluation. Our subject domain in this evaluation is a *stock price prediction* classifier which predicts whether the price of a stock is going to rise or fall in the following 10 business days (that is, whether the close price after 10 business days will be higher or lower than the current close price). The classifier uses the 23 features shown in Table 7.5 and is trained on two years of daily pricing data for the S&P500 companies, available on Yahoo! Finance. The accuracy of the classifier (for the same S&P500 companies, for predictions made on the day following the training period) is 58.5% (F1 is 68.7). In our evaluation, we present a prediction about the price of an anonymized stock to a human, along with a justification or a baseline and ask whether she would buy or avoid this stock. The task in this evaluation is to make as much money as possible betting on the stocks; we hypothesize that humans who saw our generated justification will make more accurate decisions, and therefore make more money, than those who saw the baseline.

Technical Analysis Signals	2 day high to current price ratio 5 day high to current price ratio 10 day high to current price ratio 30 day high to current price ratio	2 day momentum 5 day momentum 10 day momentum 30 day momentum
Prior Returns	2 day returns 5 day returns 10 day returns	neutralized 2 day returns neutralized 5 day returns neutralized 10 day returns

Valuation Multiples	price/earnings ratio
	price/book ratio
	ev/ebitda ratio
	return on equity
	debt to equity ratio
		price/sales ratio
		dividend yield
		return on assets
		return on revenue

Table 7.5: Features used by the stock price prediction classifier.

Foe each prediction in our evaluation, we create four justification versions. One is the baseline, where we give only the prediction and no justification. In the second, we give the prediction along with a graphical representation of the effect of key features that the user can look at. In the third, we give the prediction along with a textual justification, and in the fourth, we provide both the graphical representation and the textual justification. In that way, we evaluate the relative importance of the textual justification we propose in this chapter and a more traditional (in the Machine Learning literature) graphical visualization - although the graphical visualization still relies on our justification narratives, since it differentiates between key features and others (using the narrative roles). An example of the full justification, with the textual and graphical components, is shown in Figure 7.2. The text-only and graphical-only versions look the same, except they consist of only one of the parts.


The prediction is that this stock's price will **RISE**.

While there is strong evidence in ev-ebitda-ratio, which is normal, prior exhibits normal strong counter-evidence. Momentum_1d_11d and return-on-revenue provides unusually strong counter-evidence.

Return-on-assets provides unusually strong evidence. Furthermore, return-on-assets gives an indication of the capital intensity of the company, which will depend on the industry; companies that require large initial investments will generally have lower return-on-assets. Also, return-on-assets is one of the elements used in financial analysis using the Du Pont Identity. Moreover, return-on-assets is an indicator of how profitable a company is before leverage, and is compared with companies in the same industry. Return-on-assets is a common figure used for comparing performance of financial institutions (such as banks), because the majority of their assets will have a carrying value that is close to their actual market value. Return-on-assets is not useful for comparisons between industries because of factors of scale and peculiar capital requirements (such as reserve requirements in the insurance and banking industries).

Key feature list:

- Ev-ebitda-ratio (ExpectedEvidence)
- Return-on-assets (ExceptionalEvidence)
- Prior (ExpectedCounterEvidence)
- Momentum_1d_11d (ExceptionalCounterEvidence)
- Return-on-revenue (ExceptionalCounterEvidence)

● Total effect of all features on predicting the class
● Positive effect of a feature towards predicting the class
● Negative effect of a feature against predicting the class

Figure 7.2: Example of a full justification produced for a prediction in the evaluation.

We used predictions made by the classifier for 487 members of the S&P500 (we were missing pricing data for the other 13) on July 7th, 2016 predicting the difference in price on July 21st, 2016. We then conducted a crowd-sourced experiment where annotators were shown the prediction for one of the stocks (they were anonymized to avoid bias from any real-world knowledge the annotators may have had) along with one of the four justification options: prediction only; graphical only; text only; or text + graphical. The annotators were asked three questions: whether they would buy or avoid the stock, given the prediction and justification they saw; whether or not the information provided helped them in making their choice; and to what extent (on a scale of 1-5) they were satisfied with the information they were presented. To keep it interesting and encourage the annotators to behave like investors, we offered (relatively) large bonuses to the two annotators who made the most virtual money in the evaluation.

The results are shown in Table 7.6. We show the average accuracy, precision, recall, F1 and financial returns achieved by the annotators for each justification category (these were measured based on the agreement of the annotator's choice with the true returns of the stock, regardless of the classifier's prediction), as well as the ratio of annotators' agreement with the prediction for each justification category. In addition, we show the ratio of annotators who said the information helped them in making the decision, and the average satisfaction rating for each category.

	Prediction only	Graphical only	Text only	Text + Graphical
Accuracy	23.85%	28.67%	29.1%	32.17%
Precision	56.57	63.16	64.57	68.09
Recall	15.51	23.27	22.71	26.59
F1	24.35	34.01	33.61	38.25
Returns	1.71%	2.1%	2.79%	2.42%
Agreement w/ Prediction	85.22%	81.93%	83.98%	81.31%
Help	45.79%	78.44%	75.36%	84.39%
Satisfaction	2.54	3.17	3.21	3.38

Table 7.6: Results of the task-based evaluation. Precision, Recall and F1 are measured with respect to the positive ("buy") class.

For the accuracy metrics, all differences between categories are statistically significant with the exception of the difference between *graphical only* and *text only*, and the difference between *text + graphical* and *text only*. For "help" and satisfaction, all differences are statistically significant except for the difference between *graphical only* and *text only*.

7.4.1 Discussion

The first thing to notice in Table 7.6 is that having any kind of justification significantly increases all accuracy measures as well as the financial returns achieved by the annotators. It also reduces the agreement with the prediction of the classifier, which suggests that the justification is doing what it is intended to: give annotators enough information to decide (in some cases) that the classifier is wrong.

It is interesting to see how low the annotators' recall is with respect to precision. This suggests that the annotators' behaved like fearful investors: they chose to "buy" a stock more rarely. The classifier itself predicted that stocks will rise and fall approximately equally for the test period (in fact, with a slight advantage to rising) - this, combined with the relatively high agreement annotators had with the prediction, suggests that most deviations from the classifier's prediction were to *not* buy a stock even when it predicted it will rise, presumably because the justification was not convincing enough.

Looking at the three types of justification (graphical, textual, and both), it is clear that having a combined textual and graphical justification achieves the best results on all accuracy measures. While the returns achieved with text-only are higher than those achieved by the combined justification, keep in mind that the returns are heavily affected by the *amount* by which each stock increases or decreases, which is not modeled in any way by the classifier or the justification, and so are more volatile than the accuracy metrics. However, the accuracy differences between the textual and combined justification are not statistically significant.

Unfortunately, we were not able to show statistical significance for any of the differences between the graphical-only and the textual-only variations. It seems that different annotators have different preferences. It is clear, however, that having *both* is significantly better than having just the graphical justification, on all counts - accuracy metrics (including actual financial returns), helpfulness, and satisfaction - which shows the value of using NLG for justifying predictions.

7.5 PreJu

PreJu is a Java package that generates justifications for classifier predictions using the method and definitions described in this chapter and the framework described in Chapter 5. We have made it publicly available[6] for researchers who are interested in justifying their classifiers' predictions.

PreJu works in one of two ways: as a stand-alone configurable tool, it accepts input in the form of XML providing the effect and importance scores of the features, leaving the implementation details to the user, and allows the configuration of key feature selection, output types and other parameters via XML. As an API, it provides simple interfaces for producing justifications programmatically and currently contains implementations for Weka's Logistic Regression and SMO classifiers as integrated input sources.

Adapting PreJu to a new classifier is highly automated and simple. The user provides a training set in a standard XML format and a domain corpus; the former is used to extract the feature list and assign importance scores to the features as described in this chapter, while the latter is used to extract domain-specific information and models as described in Chapter 5.

7.6 Conclusion and Future Work

In this chapter, we tackled the relatively unexplored task of ML prediction justification via NLG. We leveraged our GEMS framework for this purpose and proposed a novel method of determining the core information (and consequently, core messages) of the prediction justification task using *narrative roles*. We then adapted our application to a stock price prediction classifier and conducted a human study which showed our generated justification significantly enhances users' ability to determine whether or not the classifier prediction is accurate, as well as their satisfaction and inclination to say the justification is helpful. These effects hold when comparing a text-only justification to no justification, and also when comparing a text+graphical justification to a graphical-only justification, suggesting that the textual part is key to all three evaluation metrics. We made our justification

[6]At `http://www.cs.columbia.edu/~orb/preju/`

method publicly available as a Java package.

Unlike the RDF applications described in the previous chapter, the data for each instance in this application comes directly from the classifier, and it does not come in a form that is ready to be converted into messages. Our justification narratives method was created especially in order to close that gap. Also unlike RDF applications (which are essentially applications for creating an entity description), this application is not something that has been widely considered in the NLG literature. We consider the work in this chapter to be a contribution in its own right (that is, independently of the GEMS framework and the NLG bottleneck solutions), as a novel method for creating NLG justifications of classifier predictions.

While the work in this chapter provides a solid proof of concept, there are many obvious ways to improve our method in future work. We list the ones we view as most important below:

- One phenomenon we neglect in our handling of feature contributions is the case of highly-correlated features. In such a case, we may miss some very important evidence because it has been separated into multiple features, all of which have relatively low weights in the classifier. A solution to this would be to use a feature aggregation method which combines highly correlated features into groups. A more optimal (from the classifier's point of view) solution, though more problematic from an interpretability point of view, would be a transformation such as Principal Components Analysis.
- PreJu is ready to use, but it requires a bit of work on the user's part to convert prediction values into XML, unless they happen to be using certain Weka classifiers. Adapting additional classifiers (and additional ML packages) is just a matter of time, but can really enhance its usability.
- Our method operates in a straightforward way with linear classifiers, but adapting it to other types of models is a very interesting and important problem. Regression models, Kernel SVMs and Neural Nets are just a few examples.

Chapter 8

Conclusions, Limitations and Future Work

In this chapter, we summarize the contributions presented in this thesis and the conclusions we draw from our experiments. We also discuss the limitations of our work and go over future work directions in more detail.

This thesis contains five major contributions. The first three - our approaches to paraphrasal template extraction, taxonomy induction and statistical discourse planning - are methods for solving various bottlenecks in concept-to-text generation, but they also have relevance independently of generation. The fourth is a framework for creating domain-adaptable, hybrid (C2T-T2T) NLG systems for a particular family of applications. The fifth is a novel approach to justifying predictions made by machine learning models using NLG.

Our approach to **mining paraphrasal templates from a simple text corpus** was presented in Chapter 2. To our knowledge, this is the first approach to paraphrase mining that does not rely on an aligned corpus, which vastly increases the amount of data and the range of domains it can be applied to. It is also unique in that it finds paraphrasal templates from sentences that were not originally paraphrases, and produces templates that utilize a rich type-system which is both much larger than those used previously, and features a hierarchical structure. In addition, our experiments in Chapter 6 show that using this approach to solve the bottleneck of finding domain-specific templates to use in generation significantly enhances user satisfaction with the generated text.

Our approach to **automatically extracting a taxonomic thesaurus** from Wikipedia, presented in Chapter 3, provides significantly higher performance for extracting taxonomic structure from Wikipedia than two leading extracted ontologies. In particular, it provides drastically higher recall while still keeping precision relatively high (and extremely high for synonyms). While the experiments in Chapter 6 show inconclusive results about the effect of using it to solve the bottleneck of domain-specific lexical choice in generation, it constitutes a contribution to the literature of taxonomy and ontology induction.

Our work on **statistical discourse planning** and **sequential discourse parsing** was discussed in Chapter 4. The main contribution in that chapter is a statistical model of discourse planning that jointly determines the order of and discourse relations between messages, and that can be extracted from a text corpus annotated with discourse relations. We show that this model significantly enhances user satisfaction with generated text, both in independent experiments in Chapter 4 and in the experiments of Chapter 6. This effect was stable both when only the discourse model was used to determine the document plan (in Chapter 4) and when it was used in conjunction with distributional and entity-based coherence (in Chapter 6). In order to automatically annotate text documents with discourse relations, we presented an efficient approach to discourse parsing which reformulates the task as a tagging task and achieves state of the art performance with a simpler model that does not include multiple hand-tuned components. We thus make a contribution to discourse parsing in addition to our work on discourse planning.

The three *data-driven solutions to generation bottlenecks* above constitute the first part of the thesis. In the second part, we focused on a generation framework that leverages those solutions, as well as hybrid C2T and T2T generation (which tackles another bottleneck, of obtaining domain-specific content), to allow fully automatic domain adaptation. We showed the power of this approach by implementing and evaluating two applications that generate descriptions of semantic web entities - illustrating how easy and quick it is to create such applications and how they can be automatically adapted to new domains - and by leveraging it to tackle the novel and difficult generation application of prediction justification.

We introduced our **framework for automatically domain-adaptable hybrid generation systems** and defined the GEMS family of generation applications it applies to in Chapter 5. Chapter 6 shows how this framework can be used for two previously-explored generation applications and describes our experiments which show the value of the hybrid approach, as well as the value of the bottleneck solutions we proposed in the first part of the thesis. In addition to the contribution of the hybrid approach, a main contribution of these two chapters is a framework for creating what might be called meta-systems: a base system for a generation application that can then be automatically adapted to any subject domain. Another contribution of Chapter 6 is the definition of a family of applications which generate descriptions of semantic web entities (relying on RDF data which is publicly available for many domains), and building on the generation framework of Chapter 5 to make the creation of new meta-systems of this type almost completely automated.

Finally, we presented a novel approach to **generating justifications for machine learning predictions** in Chapter 7, using the framework described in the previous chapters. This is an important yet under-researched topic, particularly in the NLG community. We showed that using our justification significantly enhances the ability of users to correctly identify correct and incorrect classifier predictions, as well as their satisfaction, even if they are also provided a graphical representation. We also created, and made publicly available, a library that allows the use of this justification framework by other researchers.

8.1 Limitations and Future Work

In this section we discuss the limitations of our work - in particular, of the generation framework described in the second part of the thesis - and some directions for future work.

The framework has two major *intended* limitations. First, it is limited to handing GEMS applications as defined in Chapter 5, and not other types of applications. In particular, it would not provide a major value over a manually created generation system with applications that have only one pre-set subject domain (e.g., weather reports), and it would not be able to handle situations where the application itself changes for different subject domains; for example, we cannot use it for an "essay writing" application with different academic fields as subject domains, because writing an essay on a literature topic is not the same task as writing an essay on a math topic (literature essays are interpretive and critical, while math essays rely on formal proofs). In contrast, in a GEMS application, the core task is the same for each domain while secondary content, discourse and lexical style and other aspects of the text are specific to the domain.

Second, the framework is focused on informative text, and would not in its present form be suited for applications that have to do, for example, with storytelling or dialog. We think it may be possible to create frameworks such as this for other application types, although it would require a deep rethinking of the way we have created the framework.

There is a generation bottleneck for which we did not propose a data-driven approach: content selection. In the framework, content selection occurs in two places. First, the selection of core content messages is delegated to an algorithm provided by the application designer ($\mathit{CoreMes}^{\mathcal{A}}$). Second, the selection of additional domain messages relies on two types of scores: individual preference scores, which are again requested from the application designer ($\mathit{BasePref}^{\mathcal{A}}$) and then modified in a generic way based on their relevance to core messages; and link scores, which are derived in a generic way based on their similarity. A data-driven solution that could extract the preference and link scores from the domain corpus would both make the generated text better (since content selection will be more suited to the specific domain) and reduce the amount of work required from the application designer ($\mathit{BasePref}^{\mathcal{A}}$ would no longer be needed, and perhaps $\mathit{CoreMes}^{\mathcal{A}}$ can be made simpler - it can simply provide all possible core messages, which can then be selected by a domain-specific extracted model).

Another function that the framework relies on the application designer for is $\mathit{DiscRel}^{\mathcal{A}}$, which gives possible discourse relations between a pair of messages, beyond those that are discovered generically. We think this can also be extracted from the domain corpus (by mining features of sentences that have discourse relations between them - for example, prominent predicate or verbs), and doing so would both further reduce the work required from the application designer and result in more interesting texts.

There are other obvious paths for improving and enhancing the generation framework. One candidate that we think can make a big difference with relatively little effort is an increased focus on post-processing. Replacing repetitive proper nouns with pronouns (as discussed in Chapter 6), aggregating sentences and shortening very long sentences (compression) or paragraphs (splitting) are all examples of post-processing steps that could enhance the readability of the text.

While the paraphrasal template mining approach provides variation in style and prevents repetition, we think it could be made even more powerful if it can be improved to produce "deep templates", or templates with more semantic structure. If we can identify not only the slots of the template, but also some of the hidden relations between the participating entities (a structure similar to a frame), we could use this structure to create more informed models of ordering and content selection, and end up with a full representation of the generated text (that is, a representation that contains both semantic and discourse relations). In addition, it may help in making the paraphrasal template mining method itself more precise.

8.2 Final Conclusions

We view our work in this thesis as a step towards fully generic, fully automated frameworks of generation. While generation systems that are truly generic (all-in-one systems that can generate any text given some semantic representation) are not likely to be possible in the near future, generic data-driven frameworks that automatically create non-generic generation systems (instead of relying on a human programmer to create them) can be a satisfying approximation. As we demonstrated in the second part of this thesis, it is possible (with some constraints!) to automatically adapt generation meta-systems to a new domain, and as we show with the even more constrained set of generation applications we considered in Chapter 6, it is sometimes possible to automate even the creation of the meta-systems.

Frameworks like ours make tractable certain generation applications that previously were not. The justification application we discuss in Chapter 7 is one such example, as it would not be practical to manually adapt a generation system to each classifier that we want to be able to justify.

Such frameworks have to rely on data-driven solutions for generation bottlenecks. As we show in this thesis, hybrid C2T-T2T generation provides a powerful solution for the domain content bottleneck. Our proposed methods in the first part of the thesis successfully tackle other difficult bottlenecks: obtaining domain-specific paraphrasal templates; obtaining a domain taxonomy and thesaurus; and obtaining a domain-specific discourse model. Other bottlenecks exist, and our methods are not perfect, so there is still more work to be done in this arena.

We have made available much of the code, results and resources described in this thesis. It is currently compiled and organized at `http://www.cs.columbia.edu/~orb`. For other resources, as well as for questions regarding any of them, please contact the author.

Part III

Bibliography

Bibliography

Einat Amitay and Cécile Paris. 2000. Automatically summarising web sites: Is there a way around it? In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, CIKM '00, pages 173–179, New York, NY, USA. ACM.

Ion Androutsopoulos, Gerasimos Lampouras, and Dimitrios Galanis. 2013. Generating natural language descriptions from owl ontologies: The natural owl system. *J. Artif. Int. Res.*, 48(1):671–715, October.

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 502–512, Stroudsburg, PA, USA. Association for Computational Linguistics.

Douglas E. Appelt. 1985. *Planning English Sentences.* Cambridge University Press, New York, NY, USA.

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation.* Studies in Natural Language Processing Series. Cambridge University Press.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: a nucleus for a web of open data. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg. Springer-Verlag.

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11, August.

Breck Baldwin and Thomas S. Morton. 1998. Dynamic coreference-based summarization. In *EMNLP*.

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 42–48. Association for Computational Linguistics.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 597–604, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Comput. Linguist.*, 36(4):673–721, December.

Regina Barzilay and Michael Elhadad. 1999. Using lexical chains for text summarization. *Advances in automatic text summarization*, pages 111–121.

Regina Barzilay and Mirella Lapata. 2005a. Collective content selection for concept-to-text generation. In *Proceedings of the HLT/EMNLP*, pages 331–338, Vancouver.

Regina Barzilay and Mirella Lapata. 2005b. Modeling local coherence: An entity-based approach. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 141–148, Stroudsburg, PA, USA. Association for Computational Linguistics.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 16–23, Stroudsburg, PA, USA. Association for Computational Linguistics.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL*, pages 113–120.

Regina Barzilay and Kathleen R. McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 50–57, Stroudsburg, PA, USA. Association for Computational Linguistics.

Regina Barzilay, Daryl Mccullough, Owen Rambow, Jonathan Decristofaro, Tanya Korelsky, Benoit Lavoie, and Cogentex Inc. 1998. A new approach to expert system explanations. In *In 9thInternational Workshop on Natural Language Generation*, pages 78–87.

Regina Barzilay, Noemie Elhadad, and Kathleen R. McKeown. 2001. Sentence ordering in multidocument summarization. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Regina Barzilay, Noemie Elhadad, and Kathleen R. McKeown. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *J. Artif. Int. Res.*, 17(1):35–55, August.

John A. Bateman. 1997. Enabling technology for multilingual natural language generation: The kpml development environment. *Nat. Lang. Eng.*, 3(1):15–55, March.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Nat. Lang. Eng.*, 14(4):431–455, October.

Adam L. Berger and Vibhu O. Mittal. 2000. Ocelot: A system for summarizing web pages. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 144–151, New York, NY, USA. ACM.

Rahul Bhagat and Deepak Ravichandran. 2008. Large Scale Acquisition of Paraphrases for Learning Surface Patterns. In *Proceedings of ACL-08: HLT*, pages 674–682, Columbus, Ohio, June. Association for Computational Linguistics.

Fadi Biadsy, Julia Hirschberg, and Elena Filatova. 2008. An unsupervised approach to biography production using wikipedia. In Kathleen McKeown, Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, editors, *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 807–815. The Association for Computer Linguistics.

Mustafa Bilgic and Raymond J. Mooney. 2005. Explaining recommendations: Satisfaction vs. promotion. In *Proceedings of Beyond Personalization 2005: A Workshop on the Next Stage of Recommender Systems Research at the 2005 International Conference on Intelligent User Interfaces*, San Diego, CA, January.

Or Biran and Kathleen McKeown. 2013a. Aggregated word pair features for implicit discourse relation disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 69–73.

Or Biran and Kathleen McKeown. 2013b. Classifying taxonomic relations between pairs of wikipedia articles. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 788–794.

Or Biran and Kathleen McKeown. 2014. Justification narratives for individual classifications. In *Proceedings of the AutoML Workshop at the International Conference of Machine Learning (ICML)*.

Or Biran and Kathleen McKeown. 2015a. Discourse planning with an n-gram model of relations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1973–1977, Lisbon, Portugal, September. Association for Computational Linguistics.

Or Biran and Kathleen McKeown. 2015b. Pdtb discourse parsing as a tagging task: The two taggers approach. In *Proceedings of the 16th Annual SIGdial Meeting on Discourse and Dialogue*, SIGDIAL 2015, Prague, Czech Republic.

Or Biran, Terra Blevins, and Kathleen McKeown. 2016. Mining paraphrasal typed templates from a plain text corpus. In *Proceedings of the Association for Computational Linguistics (ACL)*, Stroudsburg, PA, USA. Association for Computational Linguistics.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 546–556, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sasha Blair-Goldensohn, Kathleen R. McKeown, and Andrew Hazen Schlaikjer. 2003. Defscriber: A hybrid system for definitional qa. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 462–462, New York, NY, USA. ACM.

Sasha Blair-Goldensohn, Kathleen McKeown, and Owen Rambow. 2007. Building and refining rhetorical-semantic relation models. In *HLT-NAACL*, pages 428–435. The Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

Danushka Bollegala, Naoaki Okazaki, and Mitsuru Ishizuka. 2005. A machine learning approach to sentence ordering for multidocument summarization and its evaluation. In *Natural Language Processing–IJCNLP 2005*, pages 624–635. Springer.

Nadjet Bouayad-Agha, Richard Power, and Donia Scott. 2000. Can text structure be incompatible with rhetorical structure? In *Proceedings of the International Conference on Natural Language Generation*, pages 194–200. Association for Computational Linguistics.

Nadjet Bouayad-Agha, Gerard Casamayor, and Leo Wanner. 2011. Content selection from an ontology-based knowledge base for the generation of football summaries. In *Proceedings of the 13th European Workshop on Natural Language Generation*, ENLG '11, pages 72–81, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nadjet Bouayad-Agha, Gerard Casamayor, Leo Wanner, and Chris Mellish. 2012. Content selection from semantic web data. In *Proceedings of the Seventh International Natural Language Generation Conference*, INLG '12, pages 146–149, Stroudsburg, PA, USA. Association for Computational Linguistics.

L. Bourbeau, D. Carcagno, E. Goldberg, R. Kittredge, and A. Polguere. 1990. Bilingual generation of weather forecasts in an operations environment. In *COLNG 1990 Volume 3: Papers presented to the 13th International Conference on Computational Linguistics*.

Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini. 2005. *Ontology learning from text: methods, evaluation and applications*, volume 123. IOS press.

Jill Burstein and Daniel Marcu. 2000. Towards using text summarization for essay-based feedback. In *La 7e Conference Annuelle sur Le Traitement Automatique des Langues Naturelles (TALN)*.

Sharon A. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 120–126, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jaime R. Carbonell and Allan M. Collina. 1973. Natural semantics in artificial intelligence. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, IJCAI'73, pages 344–351, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue - Volume 16*, SIGDIAL '01, pages 1–10, Stroudsburg, PA, USA. Association for Computational Linguistics.

Emilio Carrizosa, Belen Martin-Barragan, and Dolores Romero Morales. 2006. A column generation approach for support vector machines. Technical report.

A. Cawsey. 1992. *Explanation and Interaction: The Computer Generation of Explanatory Dialogues*. A Bradford book. Bradford Book.

Yllias Chali and Shafiq R. Joty. 2008. Improving the performance of the random walk model for answering complex questions. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 9–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 976–986, Stroudsburg, PA, USA. Association for Computational Linguistics.

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th International Conference on Machine Learning*, pages 128–135. ACM.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, March.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 377–384, Stroudsburg, PA, USA. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression an integer linear programming approach. *J. Artif. Int. Res.*, 31(1):399–429, March.

L. Stephen Coles. 1969. Talking with a robot in english. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, IJCAI'69, pages 587–596, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

John M. Conroy, Judith D. Schlesinger, and Dianne P. O'Leary. 2006. Topic-focused multi-document summarization using an approximate oracle score. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 152–159, Stroudsburg, PA, USA. Association for Computational Linguistics.

D Corfield. 2010. Varieties of justification in machine learning. *Minds and Machines*, 20(2):291–301, 7.

Hercules Dalianis and Eduard H. Hovy. 1996. Aggregation in natural language generation. In *Selected papers from the Fourth European Workshop on Trends in Natural Language Generation, An Artificial Intelligence Perspective*, EWNLG '93, pages 88–105, London, UK, UK. Springer-Verlag.

Dana Dannélls, Mariana Damova, Ramona Enache, and Milen Chechev. 2012. Multilingual online generation from semantic web ontologies. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12 Companion, pages 239–242, New York, NY, USA. ACM.

Dipanjan Das and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proc. of ACL-IJCNLP*.

Hal Daumé, III and Daniel Marcu. 2006. Bayesian query-focused summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 305–312, Stroudsburg, PA, USA. Association for Computational Linguistics.

Aggeliki Dimitromanolaki and Ion Androutsopoulos. 2003. Learning to order facts for discourse planning in natural language generation. *arXiv preprint cs/0306062*.

Quang Xuan Do and Dan Roth. 2010. Constraints based taxonomic relation classification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1099–1109, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of Coling 2004*, pages 350–356, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Ji Donghong and Nie Yu. 2008. Sentence ordering based on cluster adjacency in multi-document summarization. In *Proceedings of the International Joint Conference on Natural Language Processing*.

Marek J Druzdzel and Max Henrion. 1990. Using scenarios to explain probabilistic inference.

Marek J Druzdzel. 1996. Qualitiative verbal explanations in bayesian belief networks. *AISB QUARTERLY*, pages 43–54.

Pablo A. Duboue and Kathleen R. McKeown. 2001. Empirically estimating order constraints for content planning in generation. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 172–179, Toulouse, France, July. Association for Computational Linguistics.

Pablo A. Duboue and Kathleen McKeown. 2002. Content planner construction via evolutionary algorithms and a corpus-based fitness function. In *In Proceedings of the 2 nd International Natural Language Generation Conference (INLG'02*, pages 89–96.

Pablo A. Duboue and Kathleen R. McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, pages 121–128, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pablo A. Duboue, Kathleen McKeown, and Vasileios Hatzivassiloglou. 2003. Progenie: Biographical descriptions for intelligence analysis. In *ISI*.

Daniel Duma and Ewan Klein. 2013. Generating natural language from linked data: Unsupervised template extraction. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 83–94. ASSOC COMPUTATIONAL LINGUISTICS-ACL.

Ondrej Dušek and Filip Jurcicek. 2015. Training a natural language generator from unaligned data.

David A. duVerle and Helmut Prendinger. 2009. A novel discourse parser based on support vector machine classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 665–673, Stroudsburg, PA, USA. Association for Computational Linguistics.

H.P. Edmundson and V.A. Oswald. 1959. *Automatic Indexing and Abstracting of the Contents of Documents*. Planning Research Corporation.

Samhaa R. El-Beltagy, Ahmed A. Rafea, and Ahmed H. Sameh. 1999. An agent based approach to expert system explanation. In Amruth N. Kumar and Ingrid Russell, editors, *FLAIRS Conference*, pages 153–159. AAAI Press.

Michael Elhadad and Kathleen R. McKeown. 1990. Generating connectives. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 3*, COLING '90, pages 97–101, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Elhadad and Jacques Robin. 1996. An overview of surge: a reusable comprehensive syntactic realization component. Technical report.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, December.

Christiane Fellbaum, editor. 1998. *WordNet An Electronic Lexical Database*. The MIT Press.

Vanessa Wei Feng and Graeme Hirst. 2014. A Linear-Time Bottom-Up Discourse Parser with Constraints and Post-Editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 511–521, Baltimore, Maryland, June. Association for Computational Linguistics.

Vanessa Wei Feng, Ziheng Lin, and Graeme Hirst. 2014. The impact of deep hierarchical discourse structures in the evaluation of text coherence. In *Proceedings of the 25th International Conference on Computational Linguistics*.

Raphael Feraud and Fabrice Clerot. 2002. A methodology to explain neural network classification. *Neural Networks*, 15(2):237 – 246.

Katja Filippova and Michael Strube. 2008a. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, INLG '08, pages 25–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

Katja Filippova and Michael Strube. 2008b. Sentence fusion via dependency graph compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 177–185, Stroudsburg, PA, USA. Association for Computational Linguistics.

Katja Filippova and Michael Strube. 2009. Tree linearization in english: Improving language model based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '09, pages 225–228, Stroudsburg, PA, USA. Association for Computational Linguistics.

Atsushi Fujita, Pierre Isabelle, and Roland Kuhn. 2012. Enlarging paraphrase collections through generalization and instantiation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 631–642, Stroudsburg, PA, USA. Association for Computational Linguistics.

Albert Gatt, François Portet, Ehud Reiter, Jim Hunter, Saad Mahamood, Wendy Moncur, and Somayajulu Sripada. 2009. From data to text in the neonatal intensive care unit: Using nlg technology for decision support and information management. *AI Commun.*, 22(3):153–186, August.

Sucheta Ghosh, Richard Johansson, Giuseppe Riccardi, and Sara Tonelli. 2011. Shallow discourse parsing with conditional random fields. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1071–1079.

Roxana Girju, Adriana Badulescu, and Dan Moldovan. 2003. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dimitra Gkatzia, Helen Hastie, and Oliver Lemon. 2014. Multi-adaptive natural language generation using principal component regression. In *Proceedings of the International Natural Language Generation (INLG)*.

Dimitra Gkatzia, Verena Rieser, Phil Bartie, and William Mackaness. 2015. From the virtual to the real world: Referring to objects in real-world spatial scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

D. M. Greig, B. T. Porteous, and A. H. Seheult. 1989. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, 51(2):271–279.

J.E. Grimes. 1975. *The Thread of Discourse.* Janua Linguarum. Mouton.

Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Comput. Linguist.*, 21(2):203–225, June.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 864–872, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yuqing Guo, Josef van Genabith, and Haifeng Wang. 2008. Dependency-based n-gram models for general purpose sentence realisation. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 297–304, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jurgen Habermas. 1984. *The theory of communicative action.* Beacon Press, Boston.

P. Haddawy, J. Jacobson, and C. E. Kahn. 1997. BANTER: a Bayesian network tutoring shell. *Artificial Intelligence in Medicine*, 10(2):177–200, June.

Vasileios Hatzivassiloglou, Judith L. Klavans, Melissa L. Holcombe, Regina Barzilay, Min yen Kan, and Kathleen R. McKeown. 2001. Simfinder: A flexible clustering tool for summarization. In *In Proceedings of the NAACL Workshop on Automatic Summarization*, pages 41–49.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

Tove Helldin and Maria Riveiro. 2009. Explanation methods for bayesian networks : review and application to a maritime scenario. In *Proceedings of the 3rd Skvde Workshop on Information Fusion Topics (SWIFT 2009) :*, number 2009:3 in Skvde Universisty Studies in Informatics, pages 28–32. University of Skvde.

Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00.

Jerry R Hobbs. 1979. Coherence and coreference. *Cognitive science*, 3(1):67–90.

Eduard Hovy. 1987. Generating natural language under pragmatic constraints. *Journal of Pragmatics*, 11(6):689 – 719.

Eduard H. Hovy, 1991. *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, chapter Approaches to the Planning of Coherent Text, pages 83–102. Springer US, Boston, MA.

Eduard H. Hovy. 1993. Automated discourse generation using discourse structure relations. *Artif. Intell.*, 63(1-2):341–385, October.

Ali Ibrahim, Boris Katz, and Jimmy Lin. 2003. Extracting structural paraphrases from aligned monolingual corpora. In *Proceedings of the Second International Workshop on Paraphrasing - Volume 16*, PARAPHRASE '03, pages 57–64, Stroudsburg, PA, USA. Association for Computational Linguistics.

Aleks Jakulin, Martin Možina, Janez Demšar, Ivan Bratko, and Blaž Zupan. 2005. Nomograms for visualizing support vector machines. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 108–117, New York, NY, USA. ACM.

Srinivasan Janarthanam and Oliver Lemon. 2010. Adaptive referring expression generation in spoken dialogue systems: Evaluation with real users. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '10, pages 124–131, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 891–896. ACL.

Paul D. Ji and Stephen Pulman. 2006. Sentence ordering with manifold-based classification in multi-document summarization. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 526–533, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hongyan Jing. 1998. Usage of wordnet in natural language generation. In *Proceedings of COLING-ACL'98 workshop on Usage of WordNet in Natural Language Processing Systems*, Montreal, Canada, August.

Hilary Johnson and Peter Johnson. 1993. Explanation facilities and interactive systems. In *Proceedings of the 1st International Conference on Intelligent User Interfaces*, IUI '93, pages 159–166, New York, NY, USA. ACM.

Karen Sparck Jones. 1993. What might be in a summary? *Information Retrieval*, pages 9–26.

Subbarao Kambhampati, editor. 2016. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*. IJCAI/AAAI Press.

Min-Yen Kan and Kathleen McKeown. 2002. Corpus-trained text generation for summarization.

Nikiforos Karamanis, Chris Mellish, Massimo Poesio, and Jon Oberlander. 2009. Evaluating centering for information ordering using corpora. *Comput. Linguist.*, 35(1):29–46, March.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L Berg. 2014. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Colin Kelly, Ann Copestake, and Nikiforos Karamanis. 2009. Investigating content selection for language generation using machine learning. In *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, pages 130–137, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joohyun Kim and Raymond J Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 543–551. Association for Computational Linguistics.

Richard Kittredge, Tanya Korelsky, and Owen Rambow. 1991. On the need for domain communication knowledge. *Computational Intelligence*, 7(4):305–314.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sheldon Klein, Robert F. Simmons, and System Development Corporation. 1963. Syntactic dependence and the computer generation of coherent discourse. *Mechanical Translation*, pages 50–61.

Sheldon Klein. 1965. Automatic paraphrasing in essay format. *Mechanical Translation and Computational Linguistics*, 8(3-4).

S. Klein. 1973. *Automatic Novel Writing: A Status Report.* Technical report.

Kevin Knight and Vasileios Hatzivassiloglou. 1995. Two-level, many-paths generation. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, pages 252–260, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ravi Kondadadi, Blake Howald, and Frank Schilder. 2013. A statistical nlg framework for aggregated planning and realization. In *ACL (1)*, pages 1406–1415. The Association for Computer Linguistics.

Igor Kononenko, Erik Strumbelj, Zoran Bosnic, Darko Pevec, Matjaz Kukar, and Marko Robnik-Sikonja. 2013. Explanation and reliability of individual predictions. *Informatica (Slovenia)*, 37(1):41–48.

Ioannis Konstas and Mirella Lapata. 2012. Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 752–761, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1503–1514, Seattle, Washington, USA, October. Association for Computational Linguistics.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *Proceedings of the 21st Annual Meeting on Association for Computational Linguistics*, ACL '83, pages 145–150, Stroudsburg, PA, USA. Association for Computational Linguistics.

C. Lacave and F. J. Díez. 2002. A review of explanation methods for Bayesian networks. *Knowledge Engineering Review*, 17:107–127.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Gerasimos Lampouras and Ion Androutsopoulos. 2013. Using integer linear programming in concept-to-text generation to produce more compact texts. In *ACL*, pages 561–566.

W. Landecker, M.D. Thomure, L.M.A. Bettencourt, M. Mitchell, G.T. Kenyon, and S.P. Brumby. 2013. Interpreting individual classifications of hierarchical networks. In *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on*, pages 32–38, April.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*, COLING '98, pages 704–710, Stroudsburg, PA, USA. Association for Computational Linguistics.

Irene Langkilde-Geary. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 545–552, Stroudsburg, PA, USA. Association for Computational Linguistics.

Benoit Lavoie and Owen Rambow. 1997. A fast and portable realizer for text generation systems. In *Proceedings of the fifth conference on Applied natural language processing*, pages 265–268. Association for Computational Linguistics.

Benoit Lavoie, Owen Rambow, and Ehud Reiter. 1997. Customizable descriptions of object-oriented models. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, ANLC '97, pages 253–256, Stroudsburg, PA, USA. Association for Computational Linguistics.

Huong LeThanh, Geetha Abeysinghe, and Christian Huyck. 2004. Generating discourse structures for written texts. In *Proceedings of the 20th International Conference on*

Computational Linguistics, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jessy Junyi Li and Ani Nenkova. 2014a. Addressing class imbalance for improved recognition of implicit discourse relations. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 142–150. Association for Computational Linguistics.

Jessy Junyi Li and Ani Nenkova. 2014b. Reducing sparsity improves the recognition of implicit discourse relations. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 199–207. Association for Computational Linguistics.

Brian Y. Lim and Anind K. Dey. 2010. Toolkit to support intelligibility in context-aware applications. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, Ubicomp '10.

Brian Y. Lim. 2012. *Improving Understanding and Trust with Intelligibility in Context-aware Applications*. Ph.D. thesis, Pittsburgh, PA, USA. AAI3524680.

Dekang Lin and Patrick Pantel. 2001. Dirt @sbt@discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 323–328, New York, NY, USA. ACM.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 343–351.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 997–1006. Association for Computational Linguistics.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2):151–184.

E. Lloret and E. Boldrini. 2015. Multi-genre summarization: Approach, potentials and challenges. In *eChallenges e-2015 Conference*, pages 1–9, Nov.

J. Lubsen, J. Pool, and E. van der Does. 1978. A practical device for the application of a diagnostic or prognostic function. *Methods of information in medicine*, 17(2):127–129, April.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM J. Res. Dev.*, 2(2):159–165, April.

Stephanie Lukin, Lena Reed, and Marilyn Walker. 2015. Generating sentence planning variations for story telling. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 188–197, Prague, Czech Republic, September. Association for Computational Linguistics.

Nitin Madnani, Rebecca Passonneau, Necip Fazil Ayan, John M. Conroy, Bonnie J. Dorr, Judith L. Klavans, Dianne P. O'Leary, and Judith D. Schlesinger. 2007. Measuring variability in sentence ordering for news summarization. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, ENLG '07, pages 81–88, Stroudsburg, PA, USA. Association for Computational Linguistics.

N. Madnani, Philip Resnik, Bonnie J Dorr, and R. Schwartz. 2008. Applying automatically generated semantic knowledge: A case study in machine translation. *NSF Symposium on Semantic Knowledge Discovery, Organization and Use.*

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 182–190, Stroudsburg, PA, USA. Association for Computational Linguistics.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1552–1561, Stroudsburg, PA, USA. Association for Computational Linguistics.

William C Mann and James A Moore. 1981. Computer generation of multiparagraph english text. *Computational Linguistics*, 7(1):17–29.

William C. Mann and Sandra A. Thompson. 1987. Rhetorical Structure Theory: A theory of text organization. Technical Report ISI/RS-87-190, ISI.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *ACL*, pages 368–375. ACL.

Daniel Marcu. 1997. The rhetorical parsing, summarization, and generation of natural language texts. Technical report.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June.

David Martens, Johan Huysmans, Rudy Setiono, Jan Vanthienen, and Bart Baesens. 2008. Rule extraction from support vector machines: An overview of issues and application in credit scoring. In Joachim Diederich, editor, *Rule Extraction from Support Vector Machines*, volume 80 of *Studies in Computational Intelligence*, pages 33–63. Springer Berlin Heidelberg.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 188–191, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kathleen F. McCoy. 1989. Generating context-sensitive responses to object-related misconceptions. *Artificial Intelligence*, 41(2):157 – 195.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *In Proceedings of EACL*.

Kathleen R. McKeown, Shimei Pan, James Shaw, Desmond A. Jordan, and Barry A. Allen. 1997. Language generation for multimedia healthcare briefings. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, ANLC '97, pages 277–282, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kathleen R. McKeown. 1985. Discourse strategies for generating natural-language text. *Artif. Intell.*, 27(1):1–41, September.

James R. Meehan. 1977. Tale-spin, an interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'77, pages 91–98, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Chris Mellish, Alistair Knott, and Jon Oberlander. 1998. Experiments using stochastic search for text planning. In *Proceedings of International Conference on Natural Language Generation*.

Chris Mellish, Donia Scott, Lynne Cahill, Daniel Paiva, Roger Evans, and Mike Reape. 2006. A reference architecture for natural language generation systems. *Nat. Lang. Eng.*, 12(1):1–34, March.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 775–780. AAAI Press.

Rada Mihalcea. 2007. Using wikipedia for automatic word sense disambiguation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Stroudsburg, PA, USA. Association for Computational Linguistics.

D. Milajevs, D. Kartsaklis, M. Sadrzadeh, and M. Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. In *Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar. Association for Computational Linguistics, Association for Computational Linguistics.

David Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 509–518, New York, NY, USA. ACM.

Johanna D. Moore and Cécile L. Paris. 1993. Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Comput. Linguist.*, 19(4):651–694, December.

Martin Možina, Janez Demšar, Michael Kattan, and Blaž Zupan. 2004. Nomograms for visualization of naive bayesian classifier. In *Proceedings of the 8th European Conference* *on Principles and Practice of Knowledge Discovery in Databases*, PKDD '04, pages 337–348, New York, NY, USA. Springer-Verlag New York, Inc.

Courtney Napoles, Chris Callison-Burch, Juri Ganitkevitch, and Benjamin Van Durme. 2011. Paraphrastic sentence compression with a character-based metric: Tightening without deletion. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, MTTG '11, pages 84–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hitoshi Nishikawa, Takaaki Hasegawa, Yoshihiro Matsuo, and Genichiro Kikui. 2010. Opinion summarization with integer linear programming formulation for sentence extraction and ordering. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 910–918, Stroudsburg, PA, USA. Association for Computational Linguistics.

Naoaki Okazaki, Yutaka Matsuo, and Mitsuru Ishizuka. 2004. Improving chronological sentence ordering by precedence relation. In *Proceedings of the 20th international conference on Computational Linguistics*, page 750. Association for Computational Linguistics.

Jessica Ouyang and Kathleen McKeown. 2015. Modeling reportable events as turning points in narrative. In *In Proceedings of EMNLP*. EMNLP.

Marius Paşca and Péter Dienes. 2005. Aligning needles in a haystack: Paraphrase acquisition across the web. In *Proceedings of the Second International Joint Conference on Natural Language Processing*, IJCNLP'05, pages 119–130, Berlin, Heidelberg. Springer-Verlag.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 102–109, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alexis Papadimitriou, Panagiotis Symeonidis, and Yannis Manolopoulos. 2012. A generalized taxonomy of explanations styles for traditional and social recommender systems. *Data Min. Knowl. Discov.*, 24(3):555–583, May.

Cécile L. Paris. 1988. Tailoring object descriptions to a user's level of expertise. *Comput. Linguist.*, 14(3):64–78, September.

Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 108–112.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *ACL/IJCNLP (Short Papers)*, pages 13–16. The Association for Computer Linguistics.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind K Joshi. 2008. Easily identifiable discourse relations.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *ACL/IJCNLP*, pages 683–691. The Association for Computer Linguistics.

Simone Paolo Ponzetto and Roberto Navigli. 2009. Large-scale taxonomy mapping for restructuring and integrating wikipedia. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 2083–2088, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2015. Ghostwriter: Using an LSTM for automatic rap lyric generation. In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1919–1924. The Association for Computational Linguistics.

Richard Power and Allan Third. 2010. Expressing owl axioms by english sentences: Dubious in theory, feasible in practice. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1006–1013, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Power. 2000. Planning texts by constraint satisfaction. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, pages 642–648, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. In *In Proceedings of LREC*.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 142–149.

Dragomir R. Radev and Kathleen R. McKeown. 1998. Generating natural language summaries from multiple on-line sources. *Comput. Linguist.*, 24(3):470–500, September.

Owen Rambow and Tanya Korelsky. 1992. Applied text generation. In *Proceedings of the Third Conference on Applied Natural Language Processing*, ANLC '92, pages 40–47, Stroudsburg, PA, USA. Association for Computational Linguistics.

L. F. Rau. 1988. Conceptual information extraction and information retrieval from natural language input. In *Proceedings of the Conference on User-Oriented, Content-Based, Text and Image Handling*.

Michaela Regneri and Rui Wang. 2012. Using discourse information for paraphrase extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural*

Language Processing and Computational Natural Language Learning, EMNLP-CoNLL '12, pages 916–927, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Nat. Lang. Eng.*, 3(1):57–87, March.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*, volume 33. Cambridge university press.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artif. Intell.*, 167(1-2):137–169, September.

Ehud Reiter. 2007. An architecture for data-to-text systems. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, ENLG '07, pages 97–104, Stroudsburg, PA, USA. Association for Computational Linguistics.

Verena Rieser and Oliver Lemon. 2011. Learning and evaluation of dialogue strategies for new applications: Empirical methods for optimization from small data sets. *Comput. Linguist.*, 37(1):153–196, March.

M. Robnik-Sikonja and I. Kononenko. 2008. Explaining classifications for individual instances. *Knowledge and Data Engineering, IEEE Transactions on*, 20(5):589–600, May.

Marko Robnik-Šikonja, Aristidis Likas, Constantinos Constantinopoulos, Igor Kononenko, and Erik Strumbelj. 2011. Efficiently explaining decisions of probabilistic rbf classification networks. In *Proceedings of the 10th International Conference on Adaptive and Natural Computing Algorithms - Volume Part I*, ICANNGA'11, pages 169–179, Berlin, Heidelberg. Springer-Verlag.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In Llus Mrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *EMNLP*, pages 379–389. The Association for Computational Linguistics.

Gavin Saldanha, Or Biran, Kathleen McKeown, and Alfio Gliozzo. 2016. An entity-focused approach to generating company descriptions. In *Proceedings of the Association for Computational Linguistics (ACL)*, Stroudsburg, PA, USA. Association for Computational Linguistics.

Satoshi Sekine. 2005. Automatic paraphrase discovery based on context and keywords between ne pairs. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, pages 80–87.

Satoshi Sekine. 2006. On-demand information extraction. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 731–738, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bhuban Mohan Seth and Pushpak Bhattacharyya. 2011. Rule extraction from a trained conditional random field model. In *1Proceedings of ICON-2011: 9th International Conference on Natural Language Processing.*

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 134–141, Stroudsburg, PA, USA. Association for Computational Linguistics.

Siwei Shen, Dragomir R. Radev, Agam Patel, and Güneş Erkan. 2006. Adding syntax to dynamic programming for aligning comparable texts for the generation of paraphrases. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 747–754, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stuart M. Shieber. 1986. *An introduction to Unification-based approaches to Grammar*. Center for the Study of Language and Information, Stanford University, Stanford, CA. CLSI Lecture Notes, No. 4.

Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 313–318, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Edward H Shortliffe and Bruce G Buchanan. 1975. A model of inexact reasoning in medicine. *Mathematical biosciences*, 23(3):351–379.

Robert F. Simmons. 1970. Natural language question-answering systems: 1969. *Commun. ACM*, 13(1):15–30, January.

Rashmi Sinha and Kirsten Swearingen. 2002. The role of transparency in recommender systems. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '02, pages 830–831, New York, NY, USA. ACM.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 801–808. Association for Computational Linguistics.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 151–161, Stroudsburg, PA, USA. Association for Computational Linguistics.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 149–156, Stroudsburg, PA, USA. Association for Computational Linguistics.

Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 803–810, Stroudsburg, PA, USA. Association for Computational Linguistics.

Somayajulu G. Sripada, Ehud Reiter, Ian Davy, and Kristian Nilssen. 2004. Lessons from deploying nlg technology for marine weather forecast text generation. In *in: Proceedings of PAIS-2004, 2004*, pages 760–764.

Fabian Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A core of semantic knowledge - unifying WordNet and Wikipedia. In Carey L. Williamson, Mary Ellen Zurko, and Prashant J. Patel-Schneider, Peter F. Shenoy, editors, *16th International World Wide Web Conference (WWW 2007)*, pages 697–706, Banff, Canada. ACM.

Henri Jacques Suermondt. 1992. *Explanation in Bayesian Belief Networks*. Ph.D. thesis, Stanford, CA, USA. UMI Order No. GAX92-21673.

William R. Swartout, Ccile Paris, and Johanna D. Moore. 1991. Explanations in knowledge systems: Design for explainable expert systems. *IEEE Expert*, 6(3):58–64.

W. R. Swartout. 1981. Producing explanations and justifications of expert consulting programs. Technical report, Cambridge, MA, USA.

William R. Swartout. 1983. Xplain: A system for creating and explaining expert consulting programs. *Artif. Intell.*, 21(3), September.

Zareen Syed and Tim Finin. 2010. Unsupervised techniques for discovering ontology elements from wikipedia article links. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 78–86, Los Angeles, California, June. Association for Computational Linguistics.

Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. 2009. Moviexplain: A recommender system with explanations. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09.

Duane Szafron, Russell Greiner, Paul Lu, David Wishart, Cam Macdonell, John Anvik, Brett Poulin, Zhiyong Lu, and Roman Eisner. 2003. Explaining naive bayes classifications. Technical report.

R. Teach and E. Shortliffe. 1981. An Analysis of Physician Attitudes Regarding Computer-Based Clinical Consultation Systems. *Computers and Biomedical Research*, 14:542–558.

Kapil Thadani and Kathleen McKeown. 2011. Towards strict sentence intersection: Decoding and evaluation strategies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, MTTG '11, pages 43–53, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kapil Thadani and Kathleen McKeown. 2013. Supervised sentence fusion with single-stage inference. In *IJCNLP*.

Nava Tintarev and Judith Masthoff. 2007. A survey of explanations in recommender systems. In *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*, ICDEW '07.

Stephen E. Toulmin. 1958. *The Uses of Argument.* Cambridge University Press, July.

Joe Tullio, Anind K. Dey, Jason Chalecki, and James Fogarty. 2007. How it works: A field study of non-technical users interacting with an intelligent system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07.

Marian Van Der Meulen, Robert H Logie, Yvonne Freer, Cindy Sykes, Neil McIntosh, and Jim Hunter. 2010. When a graph is poorer than 100 words: A comparison of computerised natural language generation, human generated descriptions and graphical displays in neonatal intensive care. *Applied Cognitive Psychology*, 24(1):77–89.

Marilyn Walker, Amanda Stent, François Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *J. Artif. Int. Res.*, 30(1):413–456, November.

Rui Wang and Chris Callison-Burch. 2011. Paraphrase fragment extraction from monolingual comparable corpora. In *Proceedings of the 4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web*, BUCC '11, pages 52–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

Qixin Wang, Tianyi Luo, Dong Wang, and Chao Xing. 2016. Chinese song iambics generation with neural attention-based model. In Kambhampati (Kambhampati, 2016), pages 2943–2949.

Joseph Weizenbaum. 1966. Eliza&mdash;a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, January.

Ben Wellner, James Pustejovsky, Catherine Havasi, Anna Rumshisky, and Roser Saurí. 2006. Classification of discourse coherence relations: An exploratory study using multiple knowledge sources. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, SigDIAL '06, pages 117–125, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ben Wellner. 2009. *Sequence Models and Ranking Methods for Discourse Parsing.* Ph.D. thesis, Waltham, MA, USA. AAI3339383.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei hao Su, David Vandyke, and Steve J. Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In Llus Mrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *EMNLP*, pages 1711–1721. The Association for Computational Linguistics.

Michael White and Rajakrishnan Rajkumar. 2009. Perceptron reranking for CCG realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 410–419, Singapore, August. Association for Computational Linguistics.

William Williams. 1890. *Composition and rhetoric.*

Fei Wu and Daniel S. Weld. 2008. Automatically refining the wikipedia infobox ontology. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 635–644, New York, NY, USA. ACM.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi PrasadO Christopher Bryant, and Attapol T Rutherford. 2015. The conll-2015 shared task on shallow discourse parsing. In *Proceedings of CoNLL*, page 2.

Ghim-Eng Yap, Ah-Hwee Tan, and Hwee-Hwa Pang. 2008. Explaining inferences in bayesian networks. *Applied Intelligence*, 29(3):263–278.

Majid Yazdani and Andrei Popescu-Belis. 2010. A random walk framework to compute textual semantic similarity: a unified model for three benchmark tasks. In *Proceedings of the 4th IEEE International Conference on Semantic Computing (ICSC 2010), Carnegie Mellon University, Pittsburgh, PA, USA*, 0.

L. Richard Ye and Paul E. Johnson. 1995. The impact of explanation facilities on user acceptance of expert systems advice. *MIS Q.*, 19(2):157–172, June.

Fahri Yetim. 2008. A framework for organizing justifications for strategic use in adaptive interaction contexts. In Willie Golden, Thomas Acton, Kieran Conboy, Hans van der Heijden, and Virpi Kristiina Tuunainen, editors, *ECIS*, pages 815–825.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In Kambhampati (Kambhampati, 2016), pages 2972–2978.

Victor H. Yngve. 1961. Random generation of english sentences. In *Proceedings of the International Conference on Machine Translation of Languages and Applied Language Analysis.*

S.R. Young and P. J. Hayes. 1985. Automatic classification and summarisation of banking telexes. In *Proceedings of The Second Conference on Artificial Intelligence Applications.*

Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 834–842, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Proceedings of the 23rd International Conference on Computational Linguistics.*

Part IV

Appendices

Appendix A

Type Level Relation Sequences Association Ratios

All *class* level relation sequences and their association ratio scores, for the *across* $\rightarrow$ *intra* form:

Relation 1	Relation 2	α
expansion.list	expansion.list	3.775
contingency.pragmatic_cause	expansion.instantiation	2.819
expansion.instantiation	expansion.list	2.66
contingency.pragmatic_cause	comparison.concession	2.412
expansion.alternative	entrel	2
temporal.asynchronous	temporal.asynchronous	1.978
contingency.pragmatic_cause	expansion.conjunction	1.973
temporal.synchrony	expansion.instantiation	1.961
comparison.concession	expansion.alternative	1.791
contingency.pragmatic_cause	temporal.asynchronous	1.636
temporal.asynchronous	contingency.pragmatic_cause	1.621
expansion.alternative	expansion.restatement	1.539
comparison.concession	expansion.instantiation	1.532
temporal.synchrony	entrel	1.463

comparison.contrast	contingency.condition	1.444
temporal.synchrony	temporal.synchrony	1.376
expansion.restatement	expansion.list	1.373
contingency.cause	contingency.pragmatic_condition	1.338
temporal.asynchronous	temporal.synchrony	1.321
expansion.instantiation	expansion.conjunction	1.316
contingency.pragmatic_cause	contingency.condition	1.265
contingency.cause	contingency.condition	1.252
expansion.instantiation	temporal.asynchronous	1.242
expansion.restatement	expansion.conjunction	1.242
temporal.synchrony	expansion.list	1.215
expansion.restatement	comparison.contrast	1.181
expansion.instantiation	comparison.contrast	1.178
contingency.cause	comparison.concession	1.161
temporal.asynchronous	expansion.conjunction	1.143
contingency.cause	comparison.contrast	1.126
comparison.contrast	expansion.instantiation	1.113
comparison.contrast	contingency.cause	1.111
entrel	entrel	1.109
contingency.cause	expansion.conjunction	1.105
expansion.list	comparison.concession	1.098
comparison.concession	contingency.cause	1.082
expansion.instantiation	temporal.synchrony	1.06
comparison.concession	expansion.conjunction	1.058
entrel	contingency.pragmatic_condition	1.055
temporal.asynchronous	expansion.alternative	1.055
contingency.cause	expansion.alternative	1.051
comparison.concession	expansion.restatement	1.042
comparison.concession	comparison.concession	1.019
expansion.restatement	entrel	1.014

comparison.concession	contingency.condition	1.007
expansion.restatement	expansion.alternative	1.002
contingency.cause	temporal.synchrony	0.991
comparison.concession	temporal.asynchronous	0.991
expansion.conjunction	contingency.pragmatic_cause	0.99
comparison.concession	expansion.list	0.968
temporal.synchrony	expansion.alternative	0.94
contingency.cause	contingency.cause	0.933
comparison.contrast	expansion.alternative	0.922
expansion.instantiation	comparison.concession	0.919
expansion.restatement	contingency.condition	0.918
norel	expansion.instantiation	0.914
norel	expansion.restatement	0.912
expansion.restatement	comparison.concession	0.889
contingency.cause	entrel	0.881
expansion.restatement	contingency.cause	0.88
expansion.restatement	temporal.asynchronous	0.874
temporal.asynchronous	contingency.condition	0.87
contingency.cause	temporal.asynchronous	0.869
expansion.instantiation	contingency.cause	0.868
comparison.contrast	expansion.conjunction	0.868
expansion.conjunction	comparison.concession	0.855
expansion.alternative	expansion.conjunction	0.854
comparison.concession	temporal.synchrony	0.841
expansion.conjunction	expansion.restatement	0.834
expansion.conjunction	entrel	0.829
temporal.asynchronous	expansion.restatement	0.829
expansion.instantiation	entrel	0.828
expansion.restatement	temporal.synchrony	0.817
norel	contingency.pragmatic_cause	0.817

expansion.list	temporal.synchrony	0.815
expansion.instantiation	expansion.restatement	0.808
comparison.contrast	comparison.contrast	0.785
norel	expansion.alternative	0.766
expansion.conjunction	contingency.condition	0.762
entrel	comparison.contrast	0.736
expansion.alternative	norel	0.73
entrel	norel	0.729
expansion.conjunction	expansion.conjunction	0.723
norel	norel	0.719
temporal.synchrony	norel	0.711
comparison.contrast	temporal.asynchronous	0.708
expansion.list	norel	0.706
expansion.conjunction	comparison.contrast	0.703
temporal.asynchronous	entrel	0.703
expansion.conjunction	norel	0.698
entrel	expansion.conjunction	0.697
norel	contingency.cause	0.692
entrel	temporal.asynchronous	0.687
expansion.instantiation	contingency.condition	0.666
expansion.instantiation	contingency.pragmatic_condition	0.656
expansion.conjunction	temporal.synchrony	0.652
entrel	comparison.concession	0.647
contingency.pragmatic_cause	contingency.cause	0.646
entrel	contingency.pragmatic_cause	0.641
norel	contingency.pragmatic_condition	0.64
temporal.synchrony	contingency.cause	0.636
comparison.contrast	norel	0.634
norel	comparison.concession	0.627
entrel	contingency.condition	0.627

comparison.contrast	temporal.synchrony	0.625
expansion.alternative	temporal.asynchronous	0.623
expansion.list	contingency.cause	0.614
expansion.conjunction	temporal.asynchronous	0.611
expansion.conjunction	contingency.cause	0.609
comparison.concession	norel	0.607
norel	comparison.contrast	0.603
expansion.restatement	norel	0.593
expansion.instantiation	expansion.alternative	0.593
norel	temporal.synchrony	0.589
contingency.cause	norel	0.577
temporal.synchrony	expansion.conjunction	0.576
norel	entrel	0.561
temporal.asynchronous	norel	0.556
norel	temporal.asynchronous	0.543
expansion.instantiation	norel	0.531
expansion.conjunction	expansion.list	0.517
expansion.instantiation	expansion.instantiation	0.516
expansion.alternative	contingency.cause	0.508
contingency.cause	expansion.instantiation	0.504
comparison.contrast	contingency.pragmatic_condition	0.5
norel	expansion.conjunction	0.49
entrel	temporal.synchrony	0.48
temporal.synchrony	comparison.contrast	0.478
temporal.asynchronous	contingency.cause	0.463
comparison.contrast	expansion.restatement	0.455
temporal.synchrony	comparison.concession	0.455
temporal.asynchronous	comparison.concession	0.437
norel	contingency.condition	0.437
contingency.pragmatic_cause	norel	0.436

expansion.conjunction	expansion.alternative	0.424
norel	expansion.list	0.382
temporal.synchrony	temporal.asynchronous	0.373
comparison.contrast	entrel	0.346
contingency.cause	expansion.restatement	0.332
expansion.list	temporal.asynchronous	0.323
comparison.contrast	contingency.pragmatic_cause	0.309
temporal.asynchronous	comparison.contrast	0.305
expansion.alternative	comparison.contrast	0.281
comparison.contrast	comparison.concession	0.27
comparison.contrast	expansion.list	0.242
contingency.cause	contingency.pragmatic_cause	0.23
expansion.list	comparison.contrast	0.204
entrel	contingency.cause	0.2
expansion.conjunction	expansion.instantiation	0.164
expansion.restatement	contingency.pragmatic_condition	0.149
expansion.list	expansion.conjunction	0.149
comparison.concession	comparison.contrast	0.125
comparison.concession	entrel	0.117
expansion.conjunction	contingency.pragmatic_condition	0.082
expansion.alternative	contingency.condition	0.028
expansion.alternative	temporal.synchrony	0.016
expansion.restatement	expansion.instantiation	0.008
entrel	expansion.instantiation	-0.002
expansion.restatement	expansion.restatement	-0.035
entrel	expansion.alternative	-0.058
entrel	expansion.restatement	-0.269
expansion.list	contingency.condition	-0.272
entrel	expansion.list	-0.342
contingency.cause	expansion.list	-0.348

expansion.list	expansion.restatement	-0.37
temporal.asynchronous	expansion.instantiation	-0.409

All *class* level relation sequences and their association ratio scores, for the *intra* → *across* form:

Relation 1	Relation 2	α
expansion.list	expansion.list	3.581
temporal.synchrony	contingency.pragmatic_cause	2.351
expansion.list	temporal.synchrony	2.314
expansion.instantiation	expansion.list	1.976
contingency.pragmatic_cause	comparison.concession	1.952
contingency.pragmatic_condition	expansion.restatement	1.759
temporal.asynchronous	temporal.asynchronous	1.741
expansion.restatement	comparison.concession	1.671
expansion.alternative	expansion.alternative	1.659
expansion.alternative	temporal.synchrony	1.633
contingency.pragmatic_cause	temporal.asynchronous	1.621
expansion.instantiation	expansion.alternative	1.582
expansion.restatement	expansion.alternative	1.539
expansion.instantiation	comparison.concession	1.532
entrel	temporal.synchrony	1.463
contingency.pragmatic_condition	comparison.concession	1.45
temporal.synchrony	temporal.asynchronous	1.392
contingency.cause	expansion.alternative	1.355
comparison.contrast	expansion.alternative	1.293
entrel	entrel	1.205
temporal.asynchronous	temporal.synchrony	1.184
temporal.synchrony	contingency.cause	1.157

comparison.concession	contingency.cause	1.144
contingency.condition	expansion.alternative	1.127
comparison.concession	comparison.concession	1.125
expansion.conjunction	temporal.synchrony	1.115
expansion.alternative	comparison.concession	1.098
comparison.concession	expansion.restatement	1.077
expansion.instantiation	comparison.contrast	1.052
expansion.instantiation	expansion.conjunction	1.04
entrel	comparison.concession	1.034
contingency.condition	expansion.conjunction	1.026
contingency.cause	comparison.contrast	1.005
temporal.asynchronous	comparison.concession	0.991
contingency.pragmatic_cause	expansion.conjunction	0.99
expansion.alternative	contingency.cause	0.986
contingency.condition	comparison.contrast	0.974
comparison.concession	expansion.instantiation	0.973
temporal.synchrony	expansion.list	0.969
contingency.pragmatic_cause	norel	0.96
entrel	expansion.restatement	0.957
temporal.synchrony	comparison.contrast	0.948
comparison.contrast	expansion.instantiation	0.941
temporal.asynchronous	comparison.contrast	0.926
comparison.contrast	contingency.cause	0.916
expansion.conjunction	comparison.contrast	0.9
contingency.condition	contingency.cause	0.895
temporal.synchrony	expansion.conjunction	0.87
expansion.conjunction	expansion.conjunction	0.866
contingency.condition	comparison.concession	0.853
contingency.cause	contingency.cause	0.848
expansion.restatement	comparison.contrast	0.846

norel	contingency.pragmatic_cause	0.841
contingency.cause	comparison.concession	0.841
temporal.asynchronous	expansion.conjunction	0.83
contingency.pragmatic_condition	contingency.cause	0.827
expansion.alternative	expansion.instantiation	0.816
expansion.restatement	contingency.cause	0.796
comparison.concession	expansion.conjunction	0.786
expansion.restatement	norel	0.78
expansion.list	norel	0.78
expansion.alternative	temporal.asynchronous	0.767
norel	entrel	0.767
expansion.instantiation	contingency.cause	0.766
comparison.contrast	norel	0.765
expansion.conjunction	norel	0.757
temporal.asynchronous	contingency.cause	0.756
norel	expansion.list	0.747
contingency.cause	norel	0.746
contingency.cause	expansion.conjunction	0.745
expansion.instantiation	expansion.instantiation	0.739
norel	expansion.instantiation	0.734
expansion.alternative	norel	0.731
norel	expansion.restatement	0.719
temporal.asynchronous	expansion.restatement	0.717
entrel	temporal.asynchronous	0.703
temporal.asynchronous	entrel	0.697
contingency.condition	temporal.synchrony	0.695
norel	norel	0.692
expansion.conjunction	temporal.asynchronous	0.691
comparison.contrast	comparison.contrast	0.68
comparison.contrast	expansion.conjunction	0.671

norel	temporal.synchrony	0.669
entrel	norel	0.666
norel	temporal.asynchronous	0.662
norel	expansion.conjunction	0.659
comparison.concession	norel	0.655
contingency.pragmatic_cause	expansion.restatement	0.651
contingency.condition	norel	0.65
comparison.concession	comparison.contrast	0.645
norel	contingency.cause	0.644
norel	comparison.contrast	0.64
expansion.restatement	expansion.conjunction	0.639
norel	comparison.concession	0.639
norel	expansion.alternative	0.623
contingency.condition	expansion.restatement	0.623
entrel	expansion.conjunction	0.622
contingency.condition	expansion.instantiation	0.62
expansion.list	expansion.conjunction	0.612
temporal.synchrony	expansion.instantiation	0.608
expansion.instantiation	norel	0.605
expansion.conjunction	expansion.alternative	0.603
expansion.conjunction	contingency.cause	0.594
contingency.cause	expansion.restatement	0.592
temporal.asynchronous	norel	0.589
comparison.concession	expansion.list	0.588
temporal.synchrony	entrel	0.587
temporal.synchrony	expansion.restatement	0.582
temporal.synchrony	norel	0.581
expansion.instantiation	expansion.restatement	0.568
comparison.contrast	expansion.restatement	0.559
expansion.alternative	comparison.contrast	0.554

temporal.synchrony	comparison.concession	0.554
contingency.pragmatic_condition	entrel	0.545
expansion.conjunction	comparison.concession	0.524
contingency.pragmatic_condition	norel	0.522
contingency.pragmatic_condition	comparison.contrast	0.5
comparison.contrast	comparison.concession	0.493
contingency.pragmatic_condition	expansion.conjunction	0.487
comparison.contrast	temporal.synchrony	0.478
expansion.list	comparison.contrast	0.424
expansion.alternative	expansion.conjunction	0.424
entrel	expansion.instantiation	0.423
contingency.cause	expansion.instantiation	0.422
temporal.synchrony	expansion.alternative	0.422
expansion.conjunction	expansion.restatement	0.415
temporal.synchrony	temporal.synchrony	0.395
temporal.asynchronous	expansion.instantiation	0.389
comparison.contrast	expansion.list	0.387
expansion.conjunction	entrel	0.378
comparison.concession	entrel	0.374
expansion.list	entrel	0.351
expansion.restatement	expansion.list	0.323
expansion.instantiation	entrel	0.317
contingency.pragmatic_cause	comparison.contrast	0.309
contingency.cause	temporal.synchrony	0.299
entrel	contingency.cause	0.268
expansion.conjunction	expansion.instantiation	0.264
expansion.restatement	expansion.instantiation	0.249
contingency.cause	entrel	0.246
comparison.contrast	entrel	0.242
expansion.restatement	temporal.asynchronous	0.241

contingency.condition	temporal.asynchronous	0.234
expansion.restatement	expansion.restatement	0.227
expansion.alternative	entrel	0.193
contingency.cause	temporal.asynchronous	0.191
entrel	comparison.contrast	0.179
comparison.contrast	temporal.asynchronous	0.162
contingency.condition	entrel	0.149
temporal.asynchronous	expansion.alternative	0.112
comparison.concession	temporal.asynchronous	0.1
expansion.alternative	expansion.restatement	0.086
expansion.conjunction	expansion.list	-0.033
expansion.restatement	entrel	-0.097
expansion.restatement	temporal.synchrony	-0.097
temporal.asynchronous	expansion.list	-0.188
comparison.concession	temporal.synchrony	-0.238
expansion.list	expansion.restatement	-0.332
expansion.list	contingency.cause	-0.348
contingency.cause	expansion.list	-0.485
expansion.list	expansion.instantiation	-0.518
contingency.condition	expansion.list	-0.965