Supervised and Unsupervised Learning for Sentence Compression

Jenine Turner and Eugene Charniak
Department of Computer Science
Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University
Providence, RI 02912
{jenine|ec}@cs.brown.edu

Abstract

In *Statistics-Based Summarization - Step One: Sentence Compression*, Knight and Marcu (Knight and Marcu, 2000) (K&M) present a noisy-channel model for sentence compression. The main difficulty in using this method is the lack of data; Knight and Marcu use a corpus of 1035 training sentences. More data is not easily available, so in addition to improving the original K&M noisy-channel model, we create unsupervised and semi-supervised models of the task. Finally, we point out problems with modeling the task in this way. They suggest areas for future research.

1 Introduction

Summarization in general, and sentence compression in particular, are popular topics. Knight and Marcu (henceforth K&M) introduce the task of statistical sentence compression in *Statistics-Based Summarization - Step One: Sentence Compression* (Knight and Marcu, 2000). The appeal of this problem is that it produces summarizations on a small scale. It simplifies general compression problems, such as text-to-abstract conversion, by eliminating the need for coherency between sentences. The model is further simplified by being constrained to word deletion: no rearranging of words takes place. Others have performed the sentence compression task using syntactic approaches to this problem (Mani et al., 1999) (Zajic et al., 2004), but we focus exclusively on the K&M formulation. Though the problem is simpler, it is still pertinent to current needs; generation of captions for television and audio scanning services for the blind (Grefenstette, 1998), as well as compressing chosen sentences for headline generation (Angheluta et al., 2004) are examples of uses for sentence compression. In addition to simplifying the task, K&M's noisy-channel formulation is also appealing.

In the following sections, we discuss the K&M noisy-channel model. We then present our cleaned up, and slightly improved noisy-channel model. We also develop unsupervised and semi-supervised (our term for a combination of supervised and unsupervised) methods of sentence compression with inspiration from the K&M model, and create additional constraints to improve the compressions. We conclude with the problems inherent in both models.

2 The Noisy-Channel Model

2.1 The K&M Model

The K&M probabilistic model, adapted from machine translation to this task, is the noisy-channel model. In machine translation, one imagines that a string was originally in English, but that someone adds some noise to make it a foreign string. Analogously, in the sentence compression model, the short string is the original sentence and someone adds noise, resulting in the longer sentence. Using this framework, the end goal is, given a long sentence l, to determine the short sentence s that maximizes

Proceedings of the 43rd Annual Meeting of the ACL, pages 290–297,
Ann Arbor, June 2005.

$P(s \mid l)$. By Bayes Rule,

$$P(s \mid l) = \frac{P(l \mid s)P(s)}{P(l)} \quad (1)$$

The probability of the long sentence, $P(l)$ can be ignored when finding the maximum, because the long sentence is the same in every case.

$P(s)$ is the source model: the probability that s is the original sentence. $P(l \mid s)$ is the channel model: the probability the long sentence is the expanded version of the short. This framework independently models the grammaticality of s (with $P(s)$) and whether s is a good compression of l ($P(l \mid s)$).

The K&M model uses parse trees for the sentences. These allow it to better determine the probability of the short sentence and to obtain alignments from the training data. In the K&M model, the sentence probability is determined by combining a probabilistic context free grammar (PCFG) with a word-bigram score. The joint rules used to create the compressions are generated by aligning the nodes of the short and long trees in the training data to determine expansion probabilities ($P(l \mid s)$).

Recall that the channel model tries to find the probability of the long string with respect to the short string. It obtains these probabilities by aligning nodes in the parsed parallel training corpus, and counting the nodes that align as "joint events." For example, there might be *S* $\rightarrow$ *NP VP PP* in the long sentence and *S* $\rightarrow$ *NP VP* in the short sentence; we count this as one joint event. Non-compressions, where the long version is the same as the short, are also counted. The expansion probability, as used in the channel model, is given by

$$P_{expand}(l \mid s) = \frac{count(joint(l,s))}{count(s)} \quad (2)$$

where $count(joint(l,s))$ is the count of alignments of the long rule and the short. Many compressions do not align exactly. Sometimes the parses do not match, and sometimes there are deletions that are too complex to be modeled in this way. In these cases sentence pairs, or sections of them, are ignored.

The K&M model creates a packed parse forest of all possible compressions that are grammatical with respect to the Penn Treebank (Marcus et al., 1993). Any compression given a zero expansion probability according to the training data is instead assigned a very small probability. A tree extractor (Langkilde, 2000) collects the short sentences with the highest score for $P(s \mid l)$.

2.2 Our Noisy-Channel Model

Our starting implementation is intended to follow the K&M model fairly closely. We use the same 1067 pairs of sentences from the Ziff-Davis corpus, with 32 used as testing and the rest as training. The main difference between their model and ours is that instead of using the rather ad-hoc K&M language model, we substitute the syntax-based language model described in (Charniak, 2001).

We slightly modify the channel model equation to be $P(l \mid s) = P_{expand}(l \mid s)P_{deleted}$, where $P_{deleted}$ is the probability of adding the deleted subtrees back into s to get l. We determine this probability also using the Charniak language model.

We require an extra parameter to encourage compression. We create a development corpus of 25 sentences from the training data in order to adjust this parameter. That we require a parameter to encourage compression is odd as K&M required a parameter to discourage compression, but we address this point in the penultimate section.

Another difference is that we only generate short versions for which we have rules. If we have never before seen the long version, we leave it alone, and in the rare case when we never see the long version as an expansion of itself, we allow only the short version. We do not use a packed tree structure, because we make far fewer sentences. Additionally, as we are traversing the list of rules to compress the sentences, we keep the list capped at the 100 compressions with the highest $P_{expand}(l \mid s)$. We eventually truncate the list to the best 25, still based upon $P_{expand}(l \mid s)$.

2.3 Special Rules

One difficulty in the use of training data is that so many compressions cannot be modeled by our simple method. The rules it does model, immediate constituent deletion, as in taking out the *ADVP* , of *S* $\rightarrow$ *ADVP* , *NP VP* ., are certainly common, but many good deletions are more structurally complicated. One particular type of rule, such as *NP(1)* $\rightarrow$

NP(2) CC NP(3), where the parent has at least one child with the same label as itself, and the resulting compression is one of the matching children, such as, here, *NP(2)*. There are several hundred rules of this type, and it is very simple to incorporate into our model.

There are other structures that may be common enough to merit adding, but we limit this experiment to the original rules and our new "special rules."

3 Unsupervised Compression

One of the biggest problems with this model of sentence compression is the lack of appropriate training data. Typically, abstracts do not seem to contain short sentences matching long ones elsewhere in a paper, and we would prefer a much larger corpus. Despite this lack of training data, very good results were obtained both by the K&M model and by our variant. We create a way to compress sentences without parallel training data, while sticking as closely to the K&M model as possible.

The source model stays the same, and we still pay a probability cost in the channel model for every subtree deleted. However, the way we determine $P_{expand}(l \mid s)$ changes because we no longer have a parallel text. We create joint rules using only the first section (0.mrg) of the Penn Treebank. We count all probabilistic context free grammar (PCFG) expansions, and then match up similar rules as unsupervised joint events.

We change Equation 2 to calculate $P_{expand}(s \mid l)$ without parallel data. First, let us define svo (shorter version of) to be: r_1 svo r_2 iff the righthand side of r_1 is a subsequence of the righthand side of r_2. Then define

$$P_{expand}(l \mid s) = \frac{count(l)}{\sum_{l' s.t.\ s\ svo\ l'} count(l')} \qquad (3)$$

This is best illustrated by a toy example. Consider a corpus with just 7 rules: 3 instances of *NP* $\rightarrow$ *DT JJ NN* and 4 instances of *NP* $\rightarrow$ *DT NN*.

P(NP $\rightarrow$ *DT JJ NN* $\mid$ *NP* $\rightarrow$ *DT JJ NN)* $= 1$. To determine this, you divide the count of *NP* $\rightarrow$ *DT JJ NN* $= 3$ by all the possible long versions of *NP* $\rightarrow$ *DT JJ NN* $= 3$.

P(NP $\rightarrow$ *DT JJ NN* $\mid$ *NP* $\rightarrow$ *DT NN*) $= 3/7$. The count of *NP* $\rightarrow$ *DT JJ NN* $= 3$, and the possible long versions of *NP* $\rightarrow$ *DT NN* are itself (with count of 3) and *NP* $\rightarrow$ *DT JJ NN* (with count of 4), yielding a sum of 7.

Finally, *P(NP* $\rightarrow$ *DT NN* $\mid$ *NP* $\rightarrow$ *DT NN*) $= 4/7$. The count of *NP* $\rightarrow$ *DT NN* $= 4$, and since the short (*NP* $\rightarrow$ *DT NN*) is the same as above, the count of the possible long versions is again 7.

In this way, we approximate $P_{expand}(l \mid s)$ without parallel data.

Since some of these "training" pairs are likely to be fairly poor compressions, due to the artificiality of the construction, we restrict generation of short sentences to not allow deletion of the head of any subtree. None of the special rules are applied. Other than the above changes, the unsupervised model matches our supervised version. As will be shown, this rule is not constraining enough and allows some poor compressions, but it is remarkable that any sort of compression can be achieved without training data. Later, we will describe additional constraints that help even more.

4 Semi-Supervised Compression

Because the supervised version tends to do quite well, and its main problem is that the model tends to pick longer compressions than a human would, it seems reasonable to incorporate the unsupervised version into our supervised model, in the hope of getting more rules to use. In generating new short sentences, if we have compression probabilities in the supervised version, we use those, including the special rules. The only time we use an unsupervised compression probability is when there is no supervised version of the unsupervised rule.

5 Additional Constraints

Even with the unsupervised constraint from section 3, the fact that we have artificially created our joint rules gives us some fairly ungrammatical compressions. Adding extra constraints improves our unsupervised compressions, and gives us better performance on the supervised version as well. We use a program to label syntactic arguments with the roles they are playing (Blaheta and Charniak, 2000), and the rules for complement/adjunct distinction given by (Collins, 1997) to never allow deletion of the complement. Since many nodes that should not

be deleted are not labeled with their syntactic role, we add another constraint that disallows deletion of NPs.

6 Evaluation

As with Knight and Marcu's (2000) original work, we use the same 32 sentence pairs as our Test Corpus, leaving us with 1035 training pairs. After adjusting the supervised weighting parameter, we fold the development set back into the training data.

We presented four judges with nine compressed versions of each of the 32 long sentences: A human-generated short version, the K&M version, our first supervised version, our supervised version with our special rules, our supervised version with special rules and additional constraints, our unsupervised version, our supervised version with additional constraints, our semi-supervised version, and our semi-supervised version with additional constraints. The judges were asked to rate the sentences in two ways: the grammaticality of the short sentences on a scale from 1 to 5, and the importance of the short sentence, or how well the compressed version retained the important words from the original, also on a scale from 1 to 5. The short sentences were randomly shuffled across test cases.

The results in Table 1 show compression rates, as well as average grammar and importance scores across judges.

There are two main ideas to take away from these results. First, we can get good compressions without paired training data. Second, we achieved a good boost by adding our additional constraints in two of the three versions.

Note that importance is a somewhat arbitrary distinction, since according to our judges, *all* of the computer-generated versions do as well in importance as the human-generated versions.

6.1 Examples of Results

In Figure 1, we give four examples of most compression techniques in order to show the range of performance that each technique spans. In the first two examples, we give only the versions with constraints, because there is little or no difference between the versions with and without constraints.

Example 1 shows the additional compression obtained by using our special rules. Figure 2 shows the parse trees of the original pair of short and long versions. The relevant expansion is $NP \rightarrow NP1$, PP in the long version and simply $NP1$ in the short version. The supervised version that includes the special rules learned this particular common special joint rule from the training data and could apply it to the example case. This supervised version compresses better than either version of the supervised noisy-channel model that lacks these rules. The unsupervised version does not compress at all, whereas the semi-supervised version is identical with the better supervised version.

Example 2 shows how unsupervised and semi-supervised techniques can be used to improve compression. Although the final length of the sentences is roughly the same, the unsupervised and semi-supervised versions are able to take the action of deleting the parenthetical. Deleting parentheses was never seen in the training data, so it would be extremely unlikely to occur in this case. The unsupervised version, on the other hand, sees both $PRN \rightarrow$ *lrb NP rrb* and $PRN \rightarrow NP$ in its training data, and the semi-supervised version capitalizes on this particular unsupervised rule.

Example 3 shows an instance of our initial supervised versions performing far worse than the K&M model. The reason is that currently our supervised model only generates compressions that it has seen before, unlike the K&M model, which generates all possible compressions. $S \rightarrow S$, $NP\ VP$. never occurs in the training data, and so a good compression does not exist. The unsupervised and semi-supervised versions do better in this case, and the supervised version with the added constraints does even better.

Example 4 gives an example of the K&M model being outperformed by all of our other models.

7 Problems with Noisy Channel Models of Sentence Compression

To this point our presentation has been rather normal; we draw inspiration from a previous paper, and work at improving on it in various ways. We now deviate from the usual by claiming that while the K&M model works very well, there is a technical problem with formulating the task in this way.

We start by making our noisy channel notation a

original:	Many debugging features, including user-defined break points and variable-watching and message-watching windows, have been added.
human:	Many debugging features have been added.
K&M:	Many debugging features, including user-defined points and variable-watching and message-watching windows, have been added.
supervised:	Many features, including user-defined break points and variable-watching and windows, have been added.
super (+ extra rules, constraints):	Many debugging features have been added.
unsuper (+ constraints):	Many debugging features, including user-defined break points and variable-watching and message-watching windows, have been added.
semi-supervised (+ constraints):	Many debugging features have been added.
original:	Also, Trackstar supports only the critical path method (CPM) of project scheduling.
human:	Trackstar supports the critical path method of project scheduling.
K&M:	Trackstar supports only the critical path method (CPM) of scheduling.
supervised:	Trackstar supports only the critical path method (CPM) of scheduling.
super (+ extra rules, constraints):	Trackstar supports only the critical path method (CPM) of scheduling.
unsuper (+ constraints):	Trackstar supports only the critical path method of project scheduling.
semi-supervised (+ constraints):	Trackstar supports only the critical path method of project scheduling.
original:	The faster transfer rate is made possible by an MTI-proprietary data buffering algorithm that off-loads lock-manager functions from the Q-bus host, Raimondi said.
human:	The algorithm off-loads lock-manager functions from the Q-bus host.
K&M:	The faster rate is made possible by a MTI-proprietary data buffering algorithm that off-loads lock-manager functions from the Q-bus host, Raimondi said.
supervised:	Raimondi said.
super (+ extra rules):	Raimondi said.
super (+ extra rules, constraints):	The faster transfer rate is made possible by an MTI-proprietary data buffering algorithm, Raimondi said.
unsuper (+ constraints):	The faster transfer rate is made possible, Raimondi said.
semi-supervised (+ constraints):	The faster transfer rate is made possible, Raimondi said.
original:	The SAS screen is divided into three sections: one for writing programs, one for the system's response as it executes the program, and a third for output tables and charts.
human:	The SAS screen is divided into three sections.
K&M:	The screen is divided into one
super (+ extra rules):	SAS screen is divided into three sections: one for writing programs, and a third for output tables and charts.
super (+ extra rules, constraints):	The SAS screen is divided into three sections.
unsupervised:	The screen is divided into sections: one for writing programs, one for the system's response as it executes program, and third for output tables and charts.
unsupervised (+ constraints):	Screen is divided into three sections: one for writing programs, one for the system's response as it executes program, and a third for output tables and charts.
semi-supervised:	The SAS screen is divided into three sections: one for writing programs, one for the system's response as it executes the program, and a third for output tables and charts.
semi-super (+ constraints):	The screen is divided into three sections: one for writing programs, one for the system's response as it executes the program, and a third for output tables and charts.

Figure 1: Compression Examples

	compression rate	grammar	importance
humans	53.33%	4.96	3.73
K&M	70.37%	4.57	3.85
supervised	79.85%	4.64	3.97
supervised with extra rules	67.41%	4.57	3.66
supervised with extra rules and constraints	68.44%	4.77	3.76
unsupervised	79.11%	4.38	3.93
unsupervised with constraints	77.93%	4.51	3.88
semi-supervised	81.19%	4.79	4.18
semi-supervised with constraints	79.56%	4.75	4.16

Table 1: Experimental Results

short:	(S (NP (JJ Many) (JJ debugging) (NNS features)) (VP (VBP have) (VP (VBN been) (VP (VBN added))))(. .))
long:	(S (NP (NP (JJ Many) (JJ debugging) (NNS features))(, ,) (PP (VBG including) (NP (NP (JJ user-defined)(NN break)(NNS points) (CC and)(NN variable-watching)) (CC and)(NP (JJ message-watching) (NNS windows))))(, ,)) (VP (VBP have) (VP (VBN been) (VP (VBN added))))(. .))

Figure 2: Joint Trees for special rules

bit more explicit:

$$\arg max_s p(s, L = s \mid l, L = l) \quad = \quad (4)$$
$$\arg max_s p(s, L = s) p(l, L = l \mid s, L = s)$$

Here we have introduced explicit conditioning events $L = l$ and $L = s$ to state that that the sentence in question is either the long version or the short version. We do this because in order to get the equation that K&M (and ourselves) start with, it is necessary to assume the following

$$p(s, L = s) \quad = \quad p(s) \qquad (5)$$
$$p(l, L = l \mid s, L = s) \quad = \quad p(l \mid s) \qquad (6)$$

This means we assume that the probability of, say, s as a short (compressed) sentence is simply its probability as a sentence. This will be, in general, false. One would hope that real compressed sentences are more probable as a member of the set of compressed sentences than they are as simply a member of all English sentences. However, neither K&M, nor we, have a large enough body of compressed and original sentences from which to create useful language models, so we both make this simplifying assumption. At this point it seems like a reasonable choice

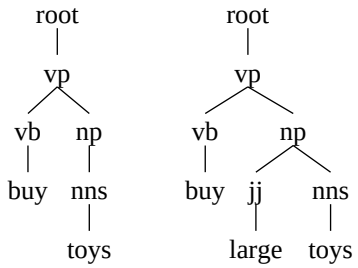

Figure 3: A compression example — trees A and B respectively

to make. In fact, it compromises the entire enterprise. To see this, however, we must descend into more details.

Let us consider a simplified version of a K&M example, but as reinterpreted for our model: how the noisy channel model assigns a probability of the compressed tree (A) in Figure 3 given the original tree B.

We compute the probabilities $p(A)$ and $p(B \mid A)$ as follows (Figure 4): We have divided the probabilities up according to whether they are contributed by the source or channel models. Those from the source

$p(A)$	$p(B \mid A)$
$p(\text{s} \rightarrow \text{vp} \mid \text{H}(\text{s}))$	$p(\text{s} \rightarrow \text{vp} \mid \text{s} \rightarrow \text{vp})$
$p(\text{vp} \rightarrow \text{vb np} \mid \text{H}(\text{vp}))$	$p(\text{vp} \rightarrow \text{vb np} \mid \text{vp} \rightarrow \text{vb np})$
$p(\text{np} \rightarrow \text{nns} \mid \text{H}(\text{np}))$	$p(\text{np} \rightarrow \text{jj nns} \mid \text{np} \rightarrow \text{nns})$
$p(\text{vb} \rightarrow \text{buy} \mid \text{H}(\text{vb}))$	$p(\text{vb} \rightarrow \text{buy} \mid \text{vb} \rightarrow \text{buy})$
$p(\text{nns} \rightarrow \text{toys} \mid \text{H}(\text{nns}))$	$p(\text{nns} \rightarrow \text{toys} \mid \text{nns} \rightarrow \text{toys})$
	$p(\text{jj} \rightarrow \text{large} \mid \text{H}(\text{jj}))$

Figure 4: Source and channel probabilities for compressing B into A

$p(B)$	$p(B \mid B)$
$p(\text{s} \rightarrow \text{vp} \mid \text{H}(\text{s}))$	$p(\text{s} \rightarrow \text{vp} \mid \text{s} \rightarrow \text{vp})$
$p(\text{vp} \rightarrow \text{vb np} \mid \text{H}(\text{vp}))$	$p(\text{vp} \rightarrow \text{vb np} \mid \text{vp} \rightarrow \text{vb np})$
$p(\text{np} \rightarrow \text{jj nns} \mid \text{H}(\text{np}))$	$p(\text{np} \rightarrow \text{jj nns} \mid \text{np} \rightarrow \text{jj nns})$
$p(\text{vb} \rightarrow \text{buy} \mid \text{H}(\text{vb}))$	$p(\text{vb} \rightarrow \text{buy} \mid \text{vb} \rightarrow \text{buy})$
$p(\text{nns} \rightarrow \text{toys} \mid \text{H}(\text{nns}))$	$p(\text{nns} \rightarrow \text{toys} \mid \text{nns} \rightarrow \text{toys})$
$p(\text{jj} \rightarrow \text{large} \mid \text{H}(\text{jj}))$	$p(\text{jj} \rightarrow \text{large} \mid \text{jj} \rightarrow \text{large})$

Figure 5: Source and channel probabilities for leaving B as B

model are conditioned on, e.g. $\text{H}(\text{np})$ the history in terms of the tree structure around the noun-phrase. In a pure PCFG this would only include the label of the node. In our language model it includes much more, such as parent and grandparent heads.

Again, following K&M, contrast this with the probabilities assigned when the compressed tree is identical to the original (Figure 5).

Expressed like this it is somewhat daunting, but notice that if all we want is to see which probability is higher (the compressed being the same as the original or truly compressed) then most of these terms cancel, and we get the rule, prefer the truly compressed if and only if the following ratio is greater than one.

$$\frac{p(\text{np} \rightarrow \text{nns} \mid \text{H}(\text{np}))}{p(\text{np} \rightarrow \text{jj nns} \mid \text{H}(\text{np}))} \frac{p(\text{np} \rightarrow \text{jj nns} \mid \text{np} \rightarrow \text{nns})}{p(\text{np} \rightarrow \text{jj nns} \mid \text{np} \rightarrow \text{jj nns})} \frac{1}{p(\text{jj} \rightarrow \text{large} \mid \text{jj} \rightarrow \text{large})} \quad (7)$$

In the numerator are the unmatched probabilities that go into the compressed sentence noisy channel probability, and in the denominator are those for when the sentence does not undergo any change. We can make this even simpler by noting that because tree-bank pre-terminals can only expand into words $p(\text{jj} \rightarrow \text{large} \mid \text{jj} \rightarrow \text{large}) = 1$. Thus the last fraction in Equation 7 is equal to one and can be ignored.

For a compression to occur, it needs to be less desirable to add an adjective in the channel model than in the source model. In fact, the opposite occurs. The likelihood of almost any constituent deletion is far lower than the probability of the constituents all being left in. This seems surprising, considering that the model we are using has had some success, but it makes intuitive sense. There are far fewer compression alignments than total alignments: identical parts of sentences are almost sure to align. So the most probable short sentence should be very barely compressed. Thus we add a weighting factor to compress our supervised version further.

K&M also, in effect, weight shorter sentences more strongly than longer ones based upon their language model. In their papers on sentence compression, they give an example similar to our "buy large toys" example. The equation they get for the channel probabilities in their example is similar to the channel probabilities we give in Figures 3 and 4. However their source probabilities are different. K&M did not have a true syntax-based language model to use as we have. Thus they divided the language model into two parts. Part one assigns probabilities to the grammar rules using a probabilistic context-free grammar, while part two assigns probabilities to the words using a bi-gram model. As they acknowledge in (Knight and Marcu, 2002), the word bigram probabilities are also included in the PCFG probabilities. So in their versions of Figures 3 and 4 they have *both* $p(toys \mid nns)$ (from the PCFG) and $p(toys \mid buy)$ for the bigram probability. In this model, the probabilities do not sum to one, because they pay the probabilistic price for guessing the word "toys" twice, based upon two different conditioning events. Based upon this language model, they prefer shorter sentences.

To reiterate this section's argument: A noisy channel model is *not* by itself an appropriate model for sentence compression. In fact, the most likely short sentence will, in general, be the same length as the long sentence. We achieve compression by weighting to give shorter sentences more likelihood. In fact, what is really required is some model that takes "utility" into account, using a utility model

in which shorter sentences are more useful. Our term giving preference to shorter sentences can be thought of as a crude approximation to such a utility. However, this is clearly an area for future research.

8 Conclusion

We have created a supervised version of the noisy-channel model with some improvements over the K&M model. In particular, we learned that adding an additional rule type improved compression, and that enforcing some deletion constraints improves grammaticality. We also show that it is possible to perform an unsupervised version of the compression task, which performs remarkably well. Our semi-supervised version, which we hoped would have good compression rates and grammaticality, had good grammaticality but lower compression than desired.

We would like to come up with a better utility function than a simple weighting parameter for our supervised version. The unsupervised version probably can also be further improved. We achieved much success using syntactic labels to constrain compressions, and there are surely other constraints that can be added.

However, more training data is always the easiest cure to statistical problems. If we can find much larger quantities of training data we could allow for much richer rule paradigms that relate compressed to original sentences. One example of a rule we would like to automatically discover would allow us to compress *all of our design goals* or

```
(NP (NP (DT all))
 (PP (IN of)
  (NP (PRP$ our) (NN design) (NNS goals)))))}
```

to *all design goals* or

```
(NP (DT all) (NN design) (NNS goals))
```

In the limit such rules blur the distinction between compression and paraphrase.

9 Acknowledgements

This work was supported by NSF grant IIS-0112435. We would like to thank Kevin Knight and Daniel Marcu for their clarification and test sentences, and Mark Johnson for his comments.

References

Roxana Angheluta, Rudradeb Mitra, Xiuli Jing, and Francine-Marie Moens. 2004. K.U.Leuven summarization system at DUC 2004. In *Document Understanding Conference*.

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *The Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 234–240.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*. The Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *The Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, San Francisco. Morgan Kaufmann.

Gregory Grefenstette. 1998. Producing intelligent telegraphic text reduction to provide an audio scanning service for the blind. In *Working Notes of the AAAI Spring Symposium on Intelligent Text Summarization*, pages 111–118.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: sentence compression. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pages 703–71.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. In *Artificial Intelligence, 139(1): 91-107*.

Irene Langkilde. 2000. Forest-based statistical sentence generation. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the Association for Computationl Linguistics*.

Inderjeet Mani, Barbara Gates, and Eric Bloedorn. 1999. Improving summaries by revising them. In *The Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*. The Association for Computational Linguistics.

Michell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

David Zajic, Bonnie Dorr, and Richard Schwartz. 2004. BBN/UMD at DUC 2004: Topiary. In *Document Understanding Conference*.