Towards Statistical Paraphrase Generation: Preliminary Evaluations of Grammaticality

Stephen Wan[12] **Mark Dras**[1] **Robert Dale**[1]
[1]Center for Language Technology
Div of Information Communication Sciences
Macquarie University
Sydney, NSW 2113
swan,madras,rdale@ics.mq.edu.au

Cécile Paris[2]
[2]Information and Communication Technologies
CSIRO
Sydney, Australia
Cecile.Paris@csiro.au

Abstract

Summary sentences are often paraphrases of existing sentences. They may be made up of recycled fragments of text taken from important sentences in an input document. We investigate the use of a statistical sentence generation technique that recombines words probabilistically in order to create new sentences. Given a set of event-related sentences, we use an extended version of the Viterbi algorithm which employs dependency relation and bigram probabilities to find the most probable summary sentence. Using precision and recall metrics for verb arguments as a measure of grammaticality, we find that our system performs better than a bigram baseline, producing fewer spurious verb arguments.

1 Introduction

Human authored summaries are more than just a list of extracted sentences. Often the summary sentence is a paraphrase of a sentence in the source text, or else a combination of phrases and words from important sentences that have been pieced together to form a new sentence. These sentences, referred to as *Non-Verbatim Sentences*, can replace extracted text to improve readability and coherence in the summary.

Consider the example in Figure 1 which presents an alignment between a human authored summary sentence and a source sentence. The text is taken from a corpus of Humanitarian Aid Proposals[1] produced by the United Nations for the purpose of convincing donors to support a relief effort.

Summary Sentence:
Every province in the country, except one, endured sporadic fighting, looting or armed banditry in 2003.

Source Sentence:
However, as the year unfolded, every province has been subjected to fighting, looting or armed banditry, with the exception of just one province (Kirundo, in northern Burundi).

Figure 1: An aligned summary and source sentence.

The example illustrates that sentence extraction alone cannot account for the breadth of human authored summary sentences. This is supported by evidence presented in (Jing and McKeown, 1999) and (Daumé III and Marcu, 2004).

Moving towards the goal of abstract-like automatic summary generation challenges us to consider mechanisms for generating non-verbatim sentences. Such a mechanism can usefully be considered as automatically generating a paraphrase.[2] We treat the problem as one in which a new and previously unseen summary sentence is to be automatically produced given some closely related sentences extracted from a source text.

Following on from (Witbrock and Mittal, 1999), we use and extend the Viterbi algorithm (Forney, 1973) for the purposes of generating non-verbatim sentences. This approach treats

[1]These are available publically at http://www.reliefweb.com.

[2]Paraphrase here includes sentences generated in an Information Fusion task (Barzilay et al., 1999).

sentence generation as a search problem. Given a set of words (taken from some set of sentences to paraphrase), we search for the most likely sequence given some language model. Intuitively, we want the generated string to be grammatical and to accurately reflect the content of the source text.

Within the Viterbi search process, each time we append a word to the partially generated sentence, we consider how well it attaches to a dependency structure. The focus of this paper is to evaluate whether or not a series of iterative considerations of dependency structure results in a grammatical generated sentence. Previous preliminary evaluations (Wan et al., 2005) indicate that the generated sequences contain less fragmented text as measured by an off-the-shelf dependency parser; more fragments would indicate a grammatically problematic sentence.

However, while encouraging, such an evaluation says little about what the actual sentence looks like. For example, such generated text might only be useful if it contains complete clauses. Thus, in this paper, we use the precision and recall metric to measure how many generated verb arguments, as extracted from dependency relations, are correct.

The remainder of this paper is structured as follows. Section 2 provides an overview introducing our approach. In Section 3, we briefly illustrate our algorithm with examples. A brief survey of related work is presented in Section 4. We present our grammaticality experiments in Section 5. We conclude with further work in Section 6.

Original Text
A military transporter was scheduled to take off in the afternoon from Yokota air base on the outskirts of Tokyo and fly to Osaka with 37,000 blankets .

Mondale said the United States, which has been flying in blankets and is sending a team of quake relief experts, was prepared to do more if Japan requested .

United States forces based in Japan will take blankets to help earthquake survivors Thursday, in the U.S. military's first disaster relief operation in Japan since it set up bases here.

Our approach with Dependencies
6: united states forces based in blankets
8: united states which has been flying in blankets
11: a military transporter was prepared to osaka with 37,000 blankets
18: mondale said the afternoon from yokota air base on the united states which has been flying in blankets
20: mondale said the outskirts of tokyo and is sending a military transporter was prepared to osaka with 37,000 blankets
23: united states forces based in the afternoon from yokota air base on the outskirts of tokyo and fly to osaka with 37,000 blankets
27: mondale said the afternoon from yokota air base on the outskirts of tokyo and is sending a military transporter was prepared to osaka with 37,000 blankets
29: united states which has been flying in the afternoon from yokota air base on the outskirts of tokyo and is sending a team of quake relief operation in blankets
31: united states which has been flying in the afternoon from yokota air base on the outskirts of tokyo and is sending a military transporter was prepared to osaka with 37,000 blankets
34: mondale said the afternoon from yokota air base on the united states which has been flying in the outskirts of tokyo and is sending a military transporter was prepared to osaka with 37,000 blankets
36: united states which has been flying in japan will take off in the afternoon from yokota air base on the outskirts of tokyo and is sending a military transporter was prepared to osaka with 37,000 blankets

Figure 2: A selection of example output. Sentences are prefixed by their length.

2 An Overview of our Approach to Statistical Sentence Generation

One could characterise the search space as being a series of nested sets. The outer most set would contain all possible word sequences. Within this, a smaller set of strings exhibiting some semblance of grammaticality might be found, though many of these might be gibberish. Further nested sets are those that are grammatical, and within those, the set of paraphrases that are entailed by the input text.

However, given that we limit ourselves to statistical techniques and avoid symbolic logic, we cannot make any claim of strict entailment. We thus propose an intermediate set of sentences which conserve the content of the source text without necessarily being entailed. These are referred to as the set of verisimilitudes, of which properly entailed sentences are a subset. The aim of our choice of features and our algorithm extension is to reduce the search space from gibberish strings to that of verisimilitudes. While generating verisimilitudes is our end goal, in this paper, we are concerned principally with the generating of grammatical sentences.

To do so, the extension adds an extra feature propagation mechanism to the Viterbi algorithm such that features are passed along a word sequence path in the search space whenever a new word is appended to it. Propagated features are used to influence the choice of subsequent words suitable for appending to a partially generated sentence. In our case, our feature is a dependency structure of the word sequence corresponding to the search path. Our present dependency representation is based on that of (Kittredge and

Mel'cuk, 1983). However, it contains only the head and modifier of a relation, ignoring relationship labels for the present.

Algorithmically, after appending a word to a path, a dependency structure of the partially generated string is obtained probabilistically. Along with bigram information, the long-distance context of dependency head information of the preceding word sequence will be useful in generating better sentences by filtering out all words that might, at a particular position in the string, lead to a spurious dependency relation in the final sentence. Example output is presented in Figure 2.

As the dependency "parsing" mechanism is linear[3] and is embedded within the Viterbi algorithm, the result is an O(n^4) algorithm.

By examining surface-syntactic dependency structure at each step in the search, resulting sentences are likely to be more grammatical. This marraige of models has been tested in other fields such as speech recognition (Chelba and Jelinek, 1998) with success. Although it is an impoverished representation of semantics, considering dependency features in our application context may also serendipitously assist verisimilitude generation.

3 The Extended Viterbi Algorithm: Propagating Dependency Structure

In this section, we present an overview of the main features of our algorithm extension. We direct the interested reader to our technical paper (Wan et al., 2005) for full details.

The Viterbi algorithm (for a comprehensive overview, see (Manning and Schütze, 1999)) is used to search for the best path across a network of nodes, where each node represents a word in the vocabulary. The best sentence is a string of words, each one emitted by the corresponding visited node on the path.

Arcs between nodes are weighted using a combination of two pieces of information: a bigram probability corresponding to that pair of words; and a probability corresponding to the likelihood of a dependency relation between that pair of words. Specifically, the transition probability defining these weights is the average of the dependency transition probability and the bigram probability.

[3]The parse is thus not necessarily optimal, in the sense of guaranteeing the most likely parse.

To simplify matters in this evaluation, we assume that the emission probability is always one. The emission probability is interpreted as being a *Content Selection* mechanism that chooses words that are likely to be in a summary. Thus, in this paper, each word has an equally likely chance of being selected for the sentence.

Transition Probability is defined as:

$$p_{tr}(w_{i+1}|w_i) = \\ average(p_{tr_{ngram}}(w_{i+1}|w_i), p_{tr_{dep}}(w_{i+1}|w_i))$$

where,

$$p_{tr_{ngram}}(w_{i+1}|w_i) = \frac{count(w_i, w_{i+1})}{count(w_i)}$$

The second function, $p_{tr_{dep}}$, is the focus of this paper and discussed in Section 3.1.
Emission Probability (for this paper, always set to 1):

$$p_{em}(w) = 1$$

Path Probability is defined recursively as:

$$p_{path}(w_0, \ldots, w_{i+1}) = \\ p_{tr_{ngram}}(w_{i+1}|w_i) \times p_{em}(w) \times p_{path}(w_0 \ldots w_i)$$

In the remaining subsections, we present an example-based discussion of how dependency-based transitions are used, and a discussion of how the dependency structure of the unfolding path is maintained and propagated within the search process.

3.1 Word Selection Using Dependency Transitions

Given two input sentences "*The relief workers distributed food to the hungry.*" and "*The UN workers requested medicine and blankets.*", the task is to generate a single sentence that contains material from these two sentences. As in (Barzilay et al., 1999), we assume that the sentences stem from the same event and thus, references can be fused together.

Imagine also that bigram frequencies have been collected from a relevant UN Humanitarian corpus. Figure 3 presents bigram probabilities and two sample paths through the lattice. The path could follow one of two forks after encountering

Graph nodes:

w_1 is *workers*

w_2 is *distributed*

w_3 is *food*

w_4 is *blankets*

e is the *end-of-sentence* state

$$\begin{tikzcd}
w_1 \arrow[r, "p(w_2|w_1)"] & w_2 \arrow[r, "p(w_3|w_2)"] \arrow[dr, "p(w_4|w_2) \neq 0"'] & w_3 \arrow[r, "p(END|w_4)"] & e \\
 & & w_4 \arrow[ur, "p(END|w_5)"'] &
\end{tikzcd}$$

Figure 3: Two search paths. One is consistent with the input text, the other is not. Assume that the probabilities are taken from a relevant corpus such that $p(blankets|distributed)$ is not zero.

the word *distributed*, since the corpus may have examples of the word pairs *distributed food* and *distributed blankets*. Since both *food* and *blankets* can reach the end-of-sentence state, both might conceivably be generated by considering just *n*-grams. However, only one is consistent with the input text.

To encourage the generation of verisimilitudes, we check for a dependency relation between *blankets* and *distributed* in the input sentence. As no evidence is found, we score this transition with a low weight. In contrast, there is evidence for the alternative path since the input text does contain a dependency relation between *food* and *distributed*.

In reality, multiple words might still conceivably be modified by future words, not just the immediately preceding word. In this example, *distributed* is the root of a dependency tree structure representing the preceding string. However, any node along the rightmost root-to-leaf branch of the dependency tree (that represents the partially generated string) could be modified. This dependency structure is determined statistically using a probabilistic model of dependency relations. To represent the rightmost branch, we use a stack data structure (referred to as the *head stack*) whereby older stack items correspond to nodes closer to the root of the dependency tree.

The probability of the dependency-based transition is estimated as follows:

$$p_{tr_{dep}}(w_{i+1}|w_i) \approx p(Dep_{sym}(w_{i+1}, headStack(w_i))) = \max_{h \in headStack(w_i)} p(Dep_{sym}(w_{i+1}, h))$$

where $p(Dep_{sym}(w_{i+1}, h))$ is inspired by and closely resembles the probabilistic functions in (Collins, 1996).

After selecting and appending a new word, we update this representation containing the governing words of the extended string that can yet be modified. The new path is then annotated with this updated stack.

3.2 Maintaining the Head Stack

There are three possible alternative outcomes to the head stack update mechanism. Given a head stack representing the dependency structure of the partially generated sentence and a new word to append to the search path, the first possibility is that the new word has no dependency relation to any of the existing stack items, in which case we simply push the new word onto the stack. For the second and third cases, we check each item on the stack and keep a record only of the best probable dependency between the new word and the appropriate stack item. The second outcome, then, is that the new word is the head of some item on the stack. All items up to and including that stack item are popped off and the new word is pushed on. The third outcome is that it modifies some item on the stack. All stack items up to (but not including) the stack item are popped off and the new word is pushed on.

We now step through the generation of the sentence "*The UN relief workers distributed food to the hungry*" which is produced by the exploration of one path in the search process. Figure 4 shows how the head stack mechanism updates and propagates the stack of governing words as we append words to the path to produce this string.

We first append the determiner *the* to the new string and push it onto the empty stack. As dictated by a high n-gram probability, the word *UN* follows. However, there is no evidence of a relation with the preceding word, so we simply push it on the stack. Similarly, *relief* is appended and also pushed on the stack.

When we encounter the word *workers* we find evidence that it governs each of the preceding

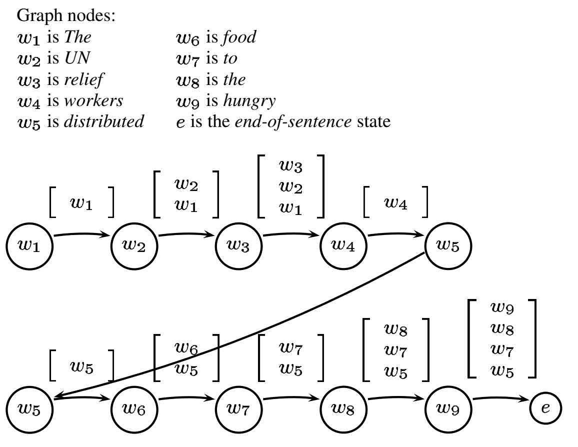

Figure 4: Propagating the head stack feature along the path.

three words. The modifiers are popped off and *workers* is pushed on. Skipping ahead, the transition *distribute food* has a high bigram probability and evidence for a dependency relation exists. This results in a strong overall path probability as opposed to the alternative fork in Figure 3. Since *distributed* can still be modified in the future by words, it is not popped off. The word *food* is pushed onto the stack as it too can still be modified.

The sentence could end there. Since we multiply path, transition and emission probabilities together, longer sentences will have a lower probability and will be penalised. However, we can choose to continue the generation process to produce a longer sentence. The word *to* modifies *distributed*. To prevent crossing dependencies, *food* is popped off the stack before pushing *to*. Appending the rest of the words is straightforward.

4 Related Work

In recent years, there has been a steady stream of research in statistical text generation (see Langkilde and Knight (1998), and Bangalore and Rambow (2000)). These approaches begin with a representation of sentence semantics that has been produced by a content planning stage. Competing realisations of the semantic representation are ranked using an n-gram model. Our approach differs in that we do not start with a semantic representation. Rather, we paraphrase the original text, searching for the best word sequence and dependency tree structure concurrently.

Summarization researchers have also studied the problem of generating non-verbatim sentences: see (Jing and McKeown, 1999), (Barzilay et al., 1999) and more recently (Daumé III and Marcu, 2004). Jing uses a HMM for learning alignments between summary and source sentences. Daume III also provides a mechanism for sub-sentential alignment but allows for alignments between multiple sentences. Both approaches provide models for later recombining sentence fragments. Our work differs primarily in granularity. Using words as a basic unit potentially offers greater flexibility in pseudo-paraphrase generation; however, like any approach that recombines text fragments, it incurs additional problems in ensuring that the generated sentence reflects the information in the input text.

In work describing summarisation as translation, Knight and Marcu (Knight and Marcu, 2002) also combine syntax models to help rank the space of possible candidate translations. Their work differs primarily in that they search over a space of trees representing the candidate translations and we search over a space of word sequences which are annotated by corresponding trees.

5 Evaluation

In this section, we describe two small experiments designed to evaluate whether a dependency-based statistical generator improves grammaticality. The first experiment uses a precision and recall styled metric on verb arguments. We find that our approach performs significantly better than the bigram baseline. The second experiment examines the precision and recall statistics on short and long distance verb arguments. We now describe these two experiments in more detail.

5.1 Improvements in Grammaticality: Verb Argument Precision and Recall

In this evaluation, we want to know what advantages a consideration of input text dependencies affords, compared to just using bigrams from the input text. Given a set of sentences which has been clustered on the basis of similarity of event, the system generates the most probable sentence by recombining words from the cluster.[4] The aim of the evaluation is to measure improvements in grammaticality. To do so, we compare our dependency based generation method against a bigram model baseline.

Since verbs are crucial in indicating the grammaticality of a clause, we examine the verb arguments of the generated sentence. We use a recall and precision metric over verb dependency relations and compare generated verb arguments with those from the input text. For any verbs included in the generated summary, we count how many generated verb-argument relations can be found amongst the input text relations for that verb. A relation match consists of an identical head, and also an identical modifier. Since word order in English is vital for grammaticality, a matching relation must also preserve the relative order of the two words within the generated sentence.
The precision metric is as follows:

$$\text{precision} = \frac{\text{count}(\text{matched-verb-relations})}{\text{count}(\text{generated-verb-relations})}$$

The corresponding recall metric is defined as:

$$\text{recall} = \frac{\text{count}(\text{matched-verb-relations})}{\text{count}(\text{source-text-verb-relations})}$$

The data for our evaluation cases is taken from the information fusion data collected by (Barzilay et al., 1999). This data is made up of news articles that have first been grouped by topic, and then component topic sentences further clustered by similarity of event. We use 100 sentence clusters and on average there are 4 sentences per cluster.

Each sentence cluster forms an evaluation case for which the task is to generate a single sentence. For each evaluation case, the baseline method and our method generates a set of answer strings, from 1 to 40 words in length.

For each cluster, sentences are parsed using the Connexor dependency parser (www.connexor.com) to obtain dependency relations used to build dependency models for that cluster. In the interests of minimising conflating factors in this comparison, we similarly

[4]This sentence could be an accurate replica of an original sentences, or a non-verbatim sentence that fuses information from various input sentences.

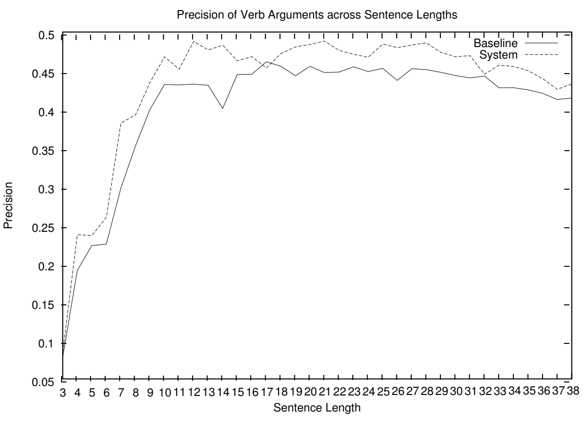

Figure 5: Verb-Argument Relation Precision scores for generated output compared to a bigram baseline

train bigram language models on the input cluster of text. This provides both the bigram baseline and our system with the best possible chance of producing a grammatical sentence given the vocabulary of the input cluster. Note that the baseline is a difficult one to beat because it is likely to reproduce long sequences from the original sentences of the input cluster. However, the exact regurgitation of input sentences is not necessarily the outcome of the baseline generator since, for each cluster, bigrams from multiple sentences are combined into a single model.

We do not use any smoothing algorithms for dependency counts in this evaluation since at present time. Thus, given the sparseness arising from a small set of sentences, our dependency probabilities tend towards boolean values. For both our approach and the baseline, the bigrams are smoothed using Katz's back-off method.

5.1.1 Results and Discussion

Figure 5 shows the average precision score across sentence lengths. That is, for each sentence length, there are 100 instances whose precisions are averaged. As can be seen, the system almost always achieves a higher precision than the baseline. As expected, precision decreases as sentence length increases.

Our approach is designed to minimise the number of spurious dependency relations generated in the resulting sentence. As this is typically measured by precision scores, recall scores are less in-

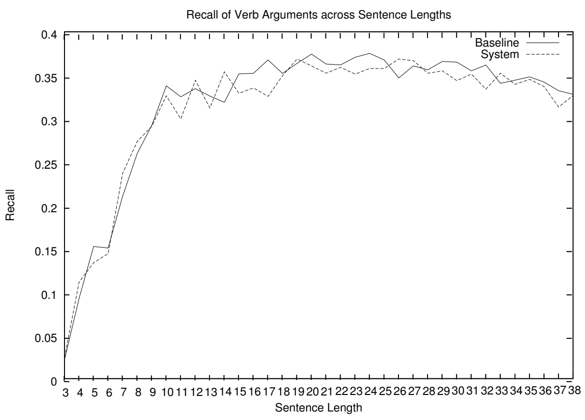

Figure 6: Verb-Argument Relation Recall scores for generated output compared to a bigram baseline

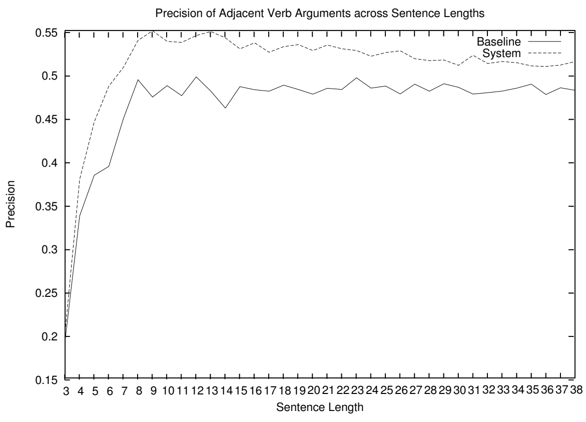

Figure 7: Adjacent Verb-Argument Relation Precision scores for generated output compared to a bigram baseline

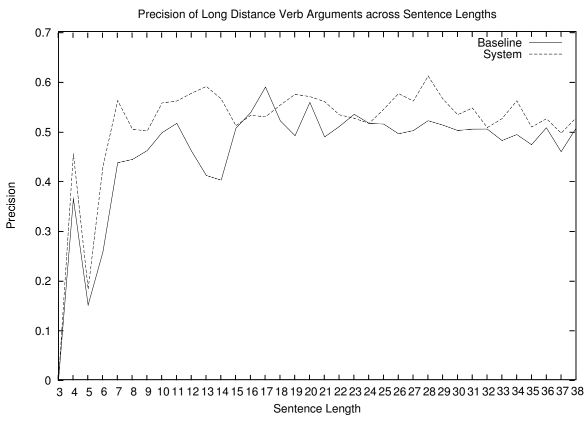

Figure 8: Long Distance Verb-Argument Relation Precision scores for generated output compared to a bigram baseline

teresting as a measure of the generated sentence. However, for completeness, they are presented Figure 6. Results indicate that our system was indistinguishable from the baseline. This is unsurprising as our approach is not designed to increase the retrieval of dependency relations from the source text.

Using a two-tailed Wilcoxon test (alpha = 0.05), we find that the differences in precision scores are significant for most sentence lengths except lengths 17 and 32. The failure to reject the null hypothesis[5] for these lengths is interpreted as idiosyncratic in our data set. In the case of the recall scores, differences are not significant.

The results support the claim that a dependency-based statistical generator improves grammaticality by reducing the number of spurious verb-argument dependency relations. It is also possible to treat dependency precision as being a superficial measure of content conservation between the generated sentence and the input sentences. Thus, it can also be seen as a poor measure of how well the summary captures the source text.

5.2 Examining Short and Long Distance Verb Arguments

Intuitively, one would expect the result from the first experiment to be reflected in both short (ie. adjacent) and long distance verb dependencies. To test this intuition, we examined the precision and recall statistics for the two types of dependencies separately. The same experimental setup is used as in the first experiment.

The results for adjacent (short) dependencies echo that of the first experiment. The precision results for adjacent dependencies are presented in Figure 7. Again, our system performs better than the baseline in terms of precision. Our system is indistinguishable in recall performance from the baseline. Due to space constraints, we omit the recall graph. Using the same significance test as before, we find that the differences in precision are generally significant across sentence lengths.

That our approach should achieve a better precision for adjacent relations supports the claim of improved grammaticality. The result resonates

[5]That is, the means of scores by our system and the baseline are not different.

well with the earlier finding that sentences generated by the dependency-based statistical generator contain fewer instances of fragmented text. If this is so, one would expect that a parser is able to identify more of the original intended dependencies.

The results for the long distance verb argument precision and recall tests are slightly different. Whilst the graph of precision scores, presented in Figure 8, shows our system often performing better than the baseline, this difference is not significant. As expected, the recall scores between our system and the baseline are on par and we again omit the results.

This result is interesting because one would expect that what our approach offers most is the ability to preserve long distance dependencies from the input text. However, long distance relations are fewer in number than adjacent relations, which account for approximately 70% of dependency relations (Collins, 1996). As the generator still does not produce perfect text, if the intermediate text between the head and modifier of a long distance relation contains any grammatical errors, the parser will obviously have difficulty in identifying the original intended relation. Given that there are fewer long distance relations, the presence of such errors quickly reduces the performance margin for the precision metric and hence no significant effect is detected. We expect that as we fine-tune the probabilistic models, the precision of long distance relations is likely to improve.

6 Conclusion and Future Work

In this paper, we presented an extension to the Viterbi algorithm which selects words in the string that are likely result in probable dependency structures. In a preliminary evaluation using precision and recall of dependency relations, we find that it improves grammaticality over a bigram model. In future work, we intend re-introduce the emission probabilities to model content selection. We also intend to use corpus-based dependency relation statistics and we would like to compare the two language models using perplexity. Finally, we would like to compare our system to that described in (Barzilay et al., 1999).

References

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of COLING*, Universität des Saarlandes, Saarbrücken, Germany.

Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of ACL*, Morristown, NJ, USA.

Ciprian Chelba and Fred Jelinek. 1998. Exploiting syntactic structure for language modelling. In *Proceedings of ACL-COLING*, Montreal, Canada.

Michael John Collins. 1996. A new statistical parser based on bigram lexical dependencies. In Arivind Joshi and Martha Palmer, editors, *Proceedings of ACL*, San Francisco.

Hal Daumé III and Daniel Marcu. 2004. A phrase-based hmm approach to document/abstract alignment. In *Proceedings of EMNLP 2004*, Barcelona, Spain.

G. David Forney. 1973. The viterbi algorithm. *Proceedings of The IEEE*, 61(3):268–278.

Hongyan Jing and Kathleen McKeown. 1999. The decomposition of human-written summary sentences. In *Research and Development in Information Retrieval*.

Richard I. Kittredge and Igor Mel'cuk. 1983. Towards a computable model of meaning-text relations within a natural sublanguage. In *The Proceedings of IJCAI*.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artif. Intell.*, 139(1):91–107.

Irene Langkilde and Kevin Knight. 1998. The practical value of N-grams in derivation. In *Proceedings of INLG*, New Brunswick, New Jersey.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Stephen Wan, Robert Dale, Mark Dras, and Cecile Paris. 2005. Searching for grammaticality and consistency: Propagating dependencies in the viterbi algorithm. In *The Proceedings of EWNLG*, Aberdeen, Scotland.

Michael J. Witbrock and Vibhu O. Mittal. 1999. Ultra-summarization (poster abstract): a statistical approach to generating highly condensed non-extractive summaries. In *The Proceedings of SIGIR*, New York, NY, USA.