Adding Syntax to Dynamic Programming for Aligning Comparable Texts for the Generation of Paraphrases

Siwei Shen[1], **Dragomir R. Radev**[1,2], **Agam Patel**[1], **Güneş Erkan**[1]
Department of Electrical Engineering and Computer Science
School of Information
University of Michigan
Ann Arbor, MI 48109
{shens, radev, agamrp, gerkan}@umich.edu

Abstract

Multiple sequence alignment techniques have recently gained popularity in the Natural Language community, especially for tasks such as machine translation, text generation, and paraphrase identification. Prior work falls into two categories, depending on the type of input used: (a) parallel corpora (e.g., multiple translations of the same text) or (b) comparable texts (non-parallel but on the same topic). So far, only techniques based on parallel texts have successfully used syntactic information to guide alignments. In this paper, we describe an algorithm for incorporating syntactic features in the alignment process for non-parallel texts with the goal of generating novel paraphrases of existing texts. Our method uses dynamic programming with alignment decision based on the local syntactic similarity between two sentences. Our results show that syntactic alignment outrivals syntax-free methods by 20% in both grammaticality and fidelity when computed over the novel sentences generated by alignment-induced finite state automata.

1 Introduction

In real life, we often encounter comparable texts such as news on the same events reported by different sources and papers on the same topic authored by different people. It is useful to recognize if one text cites another in cases like news sharing among media agencies or citations in academic work. Applications of such recognition include machine translation, text generation, paraphrase identification, and question answering, all of which have recently drawn the attention of a number of researchers in natural language processing community.

Multiple sequence alignment (MSA) is the basis for accomplishing these tasks. Previous work aligns a group of sentences into a compact word lattice (Barzilay and Lee, 2003), a finite state automaton representation that can be used to identify commonality or variability among comparable texts and generate paraphrases. Nevertheless, this approach has a drawback of over-generating ungrammatical sentences due to its "almost-free" alignment. Pang et al. provide a remedy to this problem by performing alignment on the Charniak parse trees of the clustered sentences (Pang et al., 2003). Although it is so far the most similar work to ours, Pang's solution assumes the input sentences to be semantically equivalent. Two other important references for string-based alignments algorithms, mostly with applications in Biology, are (Gusfield, 1997) and (Durbin et al., 1998).

In our approach, we work on comparable texts (not necessarily equivalent in their semantic meanings) as Barzilay and Lee did. However, we use local syntactic similarity (as opposed to lexical similarity) in doing the alignment on the raw sentences instead of on their parse trees. Because of the semantic discrepancies among the inputs, applying syntactic features in the alignment has a larger impact on the grammaticality and fidelity of the generated unseen sentences. While previous work positions the primary focus on the quality of paraphrases and/or translations, we are more interested in the relation between the use of syntactic features and the correctness of the sentences being generated, including those that are not paraphrases of the original input. Figure 1 illustrates the difference between alignment based solely on lexical similarity and alignment with consideration of syntactic features.

Ignoring syntax, the word "Milan" in both sentences is aligned. But it would unfortunately generate an ungrammatical sentence "I went to Milan is beautiful". Aligning according to syntac-

Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 747–754,
Sydney, July 2006.

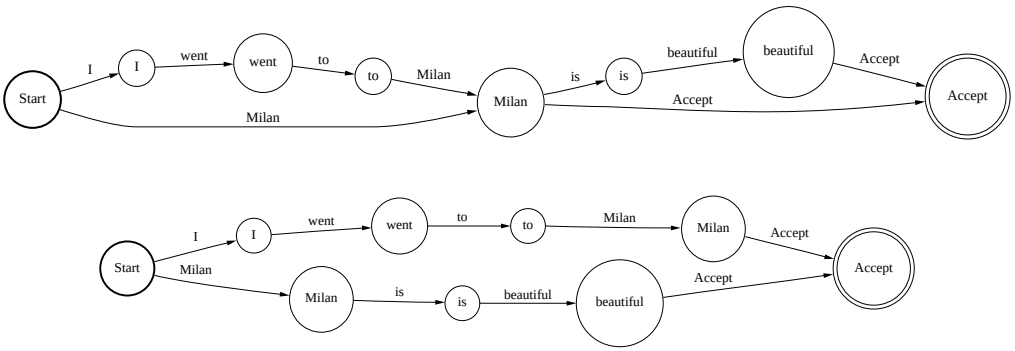

Figure 1: Alignment on lexical similarity and alignment with syntactic features of the sentences "Milan is beautiful" and "I went to Milan".

tic features, on the other hand, would avoid this improper alignment by detecting that the syntactic feature values of the two "Milan" differ too much. We shall explain syntactic features and their usages later. In this small example, our syntax-based alignment will align nothing (the bottom FSA in Figure 1) since "Milan" is the only lexically common word in both sentences. For much larger clusters in our experiments, we are able to produce a significant number of novel sentences from our alignment with such tightened syntactic conditions. Figure 2 shows one of the actual clusters used in our work that has 18 unique sentences. Two of the many automatically generated grammatical sentences are also shown.

Another piece of related work, (Quirk et al., 2004), starts off with parallel inputs and uses monolingual Statistical Machine Translation techniques to align them and generate novel sentences. In our work, the input text does not need to be nearly as parallel.

The main contribution of this paper is a syntax-based alignment technique for generating novel paraphrases of sentences that describe a particular fact. Such techniques can be potentially useful in multi-document summarizers such as Newsblaster (`http://newsblaster.cs.columbia.edu`) and NewsInEssence (`http://www.newsinessence.com`). Such systems are notorious for mostly reusing text from existing news stories. We believe that allowing them to use novel formulations of known facts will make these systems much more successful.

2 Related work

Our work is closest in spirit to the two papers that inspired us (Barzilay and Lee, 2003) and (Pang et al., 2003). Both of these papers describe how multiple sequence alignment can be used for extracting paraphrases from clustered texts. Pang et al. use as their input the multiple human English translations of Chinese documents provided by the LDC as part of the NIST machine translation evaluation. Their approach is to merge multiple parse trees into a single finite state automaton in which identical input subconstituents are merged while alternatives are converted to parallel paths in the output FSA. Barzilay and Lee, on the other hand, make use of classic techniques in biological sequence analysis to identify paraphrases from comparable texts (news from different sources on the same event).

In summary, Pang et al. use syntactic alignment of parallel texts while Barzilay and Lee use comparable (not parallel) input but ignore syntax. Our work differs from the two in that we apply syntactic information on aligning comparable texts and that the syntactic clues we use are drawn from Chunklink `ilk.uvt.nl/~sabine/homepage/software.html` output, which is further analysis from the syntactic parse trees.

Another related paper using multiple sequence alignment for text generation was (Barzilay and Lee, 2002). In that work, the authors were able to automatically acquire different lexicalizations of the same concept from "multiple-parallel corpora". We also draw some ideas from the Fitch-Margoliash method for building evolutionary trees

1. A police official said it was a Piper tourist plane and that the crash had set the top floors on fire.
2. According to ABCNEWS aviation expert John Nance, Piper planes have no history of mechanical troubles or other problems that would lead a pilot to lose control.
3. April 18, 2002 8212; A small Piper aircraft crashes into the 417-foot-tall Pirelli skyscraper in Milan, setting the top floors of the 32-story building on fire.
4. Authorities said the pilot of a small Piper plane called in a problem with the landing gear to the Milan's Linate airport at 5:54 p.m., the smaller airport that has a landing strip for private planes.
5. Initial reports described the plane as a Piper, but did not note the specific model.
6. Italian rescue officials reported that at least two people were killed after the Piper aircraft struck the 32-story Pirelli building, which is in the heart of the city s financial district.
7. MILAN, Italy AP A small piper plane with only the pilot on board crashed Thursday into a 30-story landmark skyscraper, killing at least two people and injuring at least 30.
8. Police officer Celerissimo De Simone said the pilot of the Piper Air Commander plane had sent out a distress call at 5:50 p.m. just before the crash near Milan's main train station.
9. Police officer Celerissimo De Simone said the pilot of the Piper aircraft had sent out a distress call at 5:50 p.m. 11:50 a.m.
10. Police officer Celerissimo De Simone said the pilot of the Piper aircraft had sent out a distress call at 5:50 p.m. just before the crash near Milan's main train station.
11. Police officer Celerissimo De Simone said the pilot of the Piper aircraft sent out a distress call at 5:50 p.m. just before the crash near Milan's main train station.
12. Police officer Celerissimo De Simone told The AP the pilot of the Piper aircraft had sent out a distress call at 5:50 p.m. just before crashing.
13. Police say the aircraft was a Piper tourism plane with only the pilot on board.
14. Police say the plane was an Air Commando 8212; a small plane similar to a Piper.
15. Rescue officials said that at least three people were killed, including the pilot, while dozens were injured after the Piper aircraft struck the Pirelli high-rise in the heart of the city s financial district.
16. The crash by the Piper tourist plane into the 26th floor occurred at 5:50 p.m. 1450 GMT on Thursday, said journalist Desideria Cavina.
17. The pilot of the Piper aircraft, en route from Switzerland, sent out a distress call at 5:54 p.m. just before the crash, said police officer Celerissimo De Simone.
18. There were conflicting reports as to whether it was a terrorist attack or an accident after the pilot of the Piper tourist plane reported that he had lost control.

1. Police officer Celerissimo De Simone said the pilot of the Piper aircraft, en route from Switzerland, sent out a distress call at 5:54 p.m. just before the crash near Milan's main train station.
2. Italian rescue officials reported that at least three people were killed, including the pilot, while dozens were injured after the Piper aircraft struck the 32-story Pirelli building, which is in the heart of the city s financial district.

Figure 2: A comparable cluster of size 18 and 2 novel sentences produced by syntax-based alignment.

described in (Fitch and Margoliash, 1967). That method and related techniques in Bioinformatics such as (Felsenstein, 1995) also make use of a similarity matrix for aligning a number of sequences.

3 Alignment Algorithms

Our alignment algorithm can be described as modifying Levenshtein Edit Distance by assigning different scores to lexically matched words according to their syntactic similarity. And the decision of whether to align a pair of words is based on such syntax scores.

3.1 Modified Levenshtein Edit Distance

The Levenshtein Edit Distance (LED) is a measure of similarity between two strings named after the Russian scientist Vladimir Levenshtein, who devised the algorithm in 1965. It is the number of substitutions, deletions or insertions (hence "edits") needed to transform one string into the other. We extend LED to sentence level by counting the substitutions, deletions and insertions of words necessary to transform a sentence into the other. We abbreviate this sentence-level edit distance as MLED. Similar to LED, MLED computation produces an M+1 by N+1 distance matrix, D, given two input sentences of length M and N respectively. This matrix is constructed through dynamic programming as shown in Figure 3.

$$D[i][j] = \begin{cases} 0 & \text{if } j = 0 \\ 0 & \text{if } i = 0 \\ \max \begin{pmatrix} D[i-1][j-1] + \text{match}, \\ D[i-1][j] + \text{gap}, \\ D[i][j-1] + \text{gap} \end{pmatrix} & \text{otherwise} \end{cases}$$

Figure 3: Dynamic programming in computing MLED of two sentences of length M and N.

"match" is 2 if the i^{th} word in Sentence 1 and the j^{th} word in Sentence 2 syntactically match, and is -1 otherwise. "gap" represents the score for inserting a gap rather than aligning, and is set to -1. The matching conditions of two words are far more complicated than lexical equality. Rather, we judge whether two lexically equal words match based on a predefined set of syntactic features.

The output matrix is used to guide the alignment. Starting from the bottom right entry of the matrix, we go to the matrix entry from which the value of the current cell is derived in the recursion of the dynamic programming. Call the current entry $D[i][j]$. If it gets its value from $D[i-1][j-1]$, the i^{th} word in Sentence 1 and the j^{th} word in Sentence 2 are either aligned or both aligned to a gap depending on whether they syntactically match; if the value of $D[i][j]$ is derived from $D[i][j-1]$ +

"gap", the i^{th} word in Sentence 1 is aligned to a gap inserted into Sentence 2 (the j^{th} word in Sentence 2 is not consumed); otherwise, the j^{th} word in Sentence 2 is aligned to a gap inserted into Sentence 1.

Now that we know how to align two sentences, aligning a cluster of sentences is done progressively. We start with the overall most similar pair and then respect the initial ordering of the cluster, aligning remaining sentences sequentially. Each sentence is aligned against its best match in the pool of already-aligned ones. This approach is a hybrid of the Feng-Doolittle's Algorithm (Feng and Doolittle, 1987) and a variant described in (Fitch and Margoliash, 1967).

3.2 Syntax-based Alignment

As remarked earlier, our alignment scheme judges whether two words match according to their syntactic similarity on top of lexical equality. The syntactic features are obtained from running Chunklink (Buchholz, 2000) on the Charniak parses of the clustered sentences.

3.2.1 Syntactic Features

Among all the information Chunklink provides, we use in particular the part-of-speech tags, the Chunk tags, and the syntactic dependence traces. The Chunk tag shows the constituent of a word and its relative position in that constituent. It can take one of the three values,

- "O" meaning that the word is outside of any chunk;
- "I-XP" meaning that this word is inside an XP chunk where X = N, V, P, ADV, ...;
- "B-XP" meaning that the word is at the beginning of an XP chunk.

From now on, we shall refer to the Chunk tag of a word as its IOB value (IOB was named by Tjong Kim Sang and Jorn Veeenstra (Tjong Kim Sang and Veenstra, 1999) after Ratnaparkhi (Ratnaparkhi, 1998)). For example, in the sentence "I visited Milan Theater", the IOB value for "I" is B-NP since it marks the beginning of a noun-phrase (NP). On the other hand, "Theater" has an IOB value of I-NP because it is inside a noun-phrase (Milan Theater) and is not at the beginning of that constituent. Finally, the syntactic dependence trace of a word is the path of IOB values from the root of the tree to the word itself. The last element in the trace is hence the IOB of the word itself.

3.2.2 The Algorithm

Lexically matched words but with different POS are considered not syntactically matched (e.g., race VB vs. race NN). Hence, our focus is really on pairs of lexically matched words with the same POS. We first compare their IOB values. Two IOB values are exactly matched only if they are identical (same constituent and same position); they are partially matched if they share a common constituent but have different position (e.g., B-PP vs. I-PP); and they are unmatched otherwise. For a pair of words with exactly matched IOB values, we assign 1 as their **IOB-score**; for those with partially matched IOB values, 0; and -1 for those with unmatched IOB values. The numeric values of the score are from experimental experience.

The next step is to compare syntactic dependence traces of the two words. We start with the second last element in the traces and go backward because the last one is already taken care of by the previous step. We also discard the front element of both traces since it is "I-S" for all words. The corresponding elements in the two traces are checked by the IOB-comparison described above and the scores accumulated. The process terminates as soon as one of the two traces is exhausted. Last, we adjust down the cumulative score by the length difference between the two traces. Such final score is named the **trace-score** of the two words.

We declare "unmatched" if the sum of the IOB-score and the trace-score falls below 0. Otherwise, we perform one last measurement – the relative position of the two words in their respective sentences. The relative position is defined to be the word's absolute position divided by the length of the sentence it appears in (e.g. the 4th word of a 20-word sentence has a relative position of 0.2). If the difference between two relative positions is larger than 0.4 (empirically chosen before running the experiments), we consider the two words "unmatched". Otherwise, they are syntactically matched.

The pseudo-code of checking syntactic match is shown in Figure 4.

Algorithm Check Syntactic Match of Two Words
For a pair of words W_1, W_2
if $W_1 \neq W_2$ or $pos(W_1) \neq pos(W_2)$ **then**
return "unmatched"
endif
$score := 0$
$iob_1 := iob(W_1)$
$iob_2 := iob(W_2)$
$score \mathrel{+}= compare\_iobs(iob_1, iob_2)$
$trace_1 := trace(W_1)$
$trace_2 := trace(W_2)$
$score \mathrel{+}= compare\_traces(trace_1, trace_2)$
if score < 0 **then**
return "unmatched"
endif
$relpos_1 := pos(W_1)/lengthOf(S_1)$
$relpos_2 := pos(W_2)/lengthOf(S_2)$
if $|relpos_1 - relpos_2| \geq 0.4$ **then**
return "unmatched"
endif
return "matched"
Function $compare\_iobs(iob_1, iob_2)$
if $iob_1 = iob_2$ **then**
return 1
endif
if $substring(iob_1, 1) = substring(iob_2, 1)$ **then**
return 0
endif
return -1
Function $compare\_traces(trace_1, trace_2)$
Remove first and last elements from both traces
$score := 0$
$i := lengthOf(trace_1) - 1$
$j := lengthOf(trace_2) - 1$
while $i \geq 0$ and $j \geq 0$ **do**
$next := compare\_iobs(trace_1[i], trace_2[j])$
$score \mathrel{+}= next * 0.5$
$i--$
$j--$
endwhile
$score -= |lengthOf(trace_1) - lengthOf(trace_2)| * 0.5$
return $score$

Figure 4: Algorithm for checking the syntactic match between two words.

4 Evaluation

4.1 Experimental Setup

4.1.1 Data

The data we use in our experiment come from a number of sentence clusters on a variety of topics, but all related to the Milan plane crash event. This cluster was collected manually from the Web of five different news agencies (ABC, CNN, Fox, MSNBC, and USAToday). It concerns the April 2002 crash of a small plane into a building in Milan, Italy and contains a total of 56 documents published over a period of 1.5 days. To divide this corpus into representative smaller clusters, we had a colleague thoroughly read all 56 documents in the cluster and then create a list of important facts surrounding the story. We then picked key terms related to these facts, such as names (Fasulo - the pilot) and locations (Locarno - the city from which the plane had departed). Finally, we automatically clustered sentences based on the presence of these key terms, resulting in 21 clusters of topically related (comparable) sentences. The 21 clusters are grouped into three categories: 7 in training set, 3 in dev-testing set, and the remaining 11 in testing set. Table 1 shows the name and size of each cluster.

Cluster	Number of Sentences
Training clusters	
ambulance	10
belie	14
built	6
malpensa	4
piper	18
president	17
route	11
Dev-test clusters	
hospital	17
rescue	12
witness	6
Test clusters	
accident	30
cause	18
fasulo	33
floor	79
government	22
injur	43
linate	21
rockwell	9
spokes	18
suicide	22
terror	62

Table 1: Experimental clusters.

4.1.2 Different Versions of Alignment

To test the usefulness of our work, we ran 5 different alignments on the clusters. The first three represent different levels of baseline performance (without syntax consideration) whereas the last two fully employ the syntactic features but treat stop words differently. Table 2 describes the 5 versions of alignment.

Run	Description
V1	Lexical alignment on everything possible
V2	Lexical alignment on everything but commas
V3	Lexical alignment on everything but commas and stop words
V4	Syntactic alignment on everything but commas and stop words
V5	Syntactic alignment on everything but commas

Table 2: Alignment techniques used in the experiments.

Alignment	Grammaticality	Fidelity
V1	2.89	2.98
V2	3.00	2.95
V3	3.15	3.22
V4	3.68	3.59
V5	3.47	3.30

Table 3: Evaluation results on training and dev-testing clusters. For the results on the test clusters, see Table 6

The motivation of trying such variations is as follows. Stop words often cause invalid alignment because of their high frequencies, and so do punctuations. Aligning on commas, in particular, is likely to produce long sentences that contain multiple sentence segments ungrammatically patched together.

4.1.3 Training and Testing

In order to get the best possible performance of the syntactic alignment versions, we use clusters in the training and dev-test sets to tune up the parameter values in our algorithm for checking syntactic match. The parameters in our algorithm are not independent. We pay special attention to the threshold of relative position difference, the discount factor of the trace length difference penalty, and the scores for exactly matched and partially matched IOB values. We try different parameter settings on the training clusters, and apply the top ranking combinations (according to human judgments described later) on clusters in the dev-testing set. The values presented in this paper are the manually selected ones that yield the best performance on the training and dev-testing sets.

Experimenting on the testing data, we have two hypotheses to verify: 1) the 2 syntactic versions outperform the 3 baseline versions by both grammaticality and fidelity (discussed later) of the novel sentences produced by alignment; and 2) disallowing alignment on stop words and commas enhances the performance.

4.2 Experimental Results

For each cluster, we ran the 5 alignment versions and produce 5 FSA's. From each FSA (corresponding to a cluster A and alignment version i), 100 sentences are randomly generated. We removed those that appear in the original cluster. The remaining ones are hence novel sentences, among which we randomly chose 10 to test the performance of alignment version i on cluster A.

In the human evaluation, each sentence received two scores – grammaticality and fidelity. These two properties are independent since a sentence could possibly score high on fidelity even if it is not fully grammatical. Four different scores are possible for both criteria: (4) perfect (fully grammatical or faithful); (3) good (occasional errors or quite faithful); (2) bad (many grammar errors or unfaithful pieces); and (1) nonsense.

4.2.1 Results from the Training Phase

Four judges help our evaluation in the training phase. They are provided with the original clusters during the evaluation process, yet they are given the sentences in shuffled order so that they have no knowledge about from which alignment version each sentence is generated. Table 3 shows the averages of their evaluation on the 10 clusters in training and dev-testing set. Each cell corresponds to 400 data points as we presented 10 sentences per cluster per alignment version to each of the 4 judges (10 x 10 x 4 = 400).

4.2.2 Results from the Testing Phase

After we have optimized the parameter configuration for our syntactic alignment in the training phase, we ask another 6 human judges to evaluate our work on the testing data. These 6 judges come from diverse background including Information, Computer Science, Linguistics, and Bioinformatics. We distribute the 11 testing clusters among them so that each cluster gets evaluated by at least 3 judges. The workload for each judge is 6 clusters x 5 versions/cluster x 10 sentences/cluster-version = 300 sentences. Similar to the training phase, they receive the sentences in shuffled order without knowing the correspondence between

sentences and alignment versions. Detailed average statistics are shown in Table 4 and Table 5 for grammaticality and fidelity, respectively. Each cell is the average over 30 - 40 data points, and notice the last row is not the mean of the other rows since the number of sentences evaluated for each cluster varies.

Cluster	V1	V2	V3	V4	V5
rockwell	2.27	2.93	3.00	3.60	3.03
cause	2.77	2.83	3.07	3.10	2.93
spokes	2.87	3.07	3.57	3.83	3.50
linate	2.93	3.14	3.26	3.64	3.77
government	2.75	2.83	3.27	3.80	3.20
suicide	2.19	2.51	3.29	3.57	3.11
accident	2.92	3.27	3.54	3.72	3.56
fasulo	2.52	2.52	3.15	3.54	3.32
injur	2.29	2.92	3.03	3.62	3.29
terror	3.04	3.11	3.61	3.23	3.63
floor	2.47	2.77	3.40	3.47	3.27
Overall	2.74	2.75	3.12	3.74	3.29

Table 4: Average grammaticality scores on testing clusters.

Cluster	V1	V2	V3	V4	V5
rockwell	2.25	2.75	3.20	3.80	2.70
cause	2.42	3.04	2.92	3.48	3.17
spokes	2.65	2.50	3.20	3.00	3.05
linate	3.15	3.27	3.15	3.36	3.42
government	2.85	3.24	3.14	3.81	3.20
suicide	2.38	2.69	2.93	3.68	3.23
accident	3.14	3.42	3.56	3.91	3.57
fasulo	2.30	2.48	3.14	3.50	3.48
injur	2.56	2.28	2.29	3.18	3.22
terror	2.65	2.48	3.68	3.47	3.20
floor	2.80	2.90	3.10	3.70	3.30
Overall	2.67	2.69	3.07	3.77	3.23

Table 5: Average fidelity scores on testing clusters.

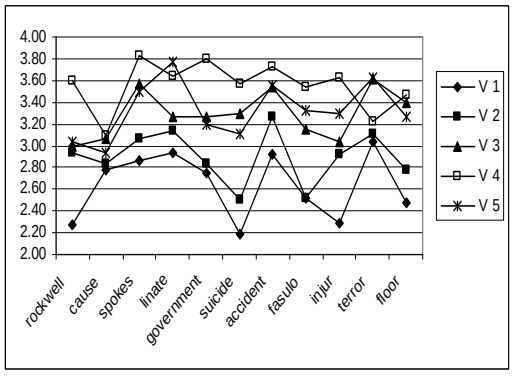

Figure 5: Performance of 5 alignment versions by grammaticality.

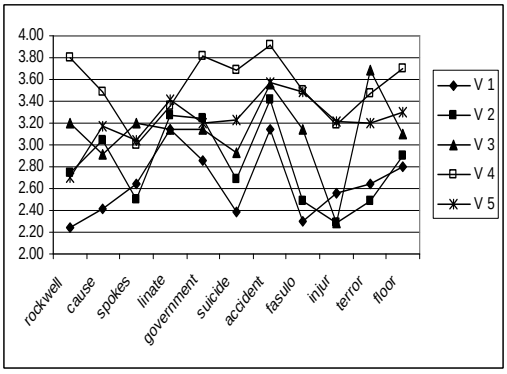

Figure 6: Performance of 5 alignment versions by fidelity.

4.3 Result Analysis

The results support both our hypotheses. For Hypothesis I, we see that the performance of the two syntactic alignments was higher than the non-syntactic versions. In particular, Version 4 outperforms the the best baseline version by 19.9% on grammaticality and by 22.8% on fidelity. Our second hypothesis is also verified – disallowing alignment on stop words and commas yields better results. This is reflected by the fact that Version 4 beats Version 5, and Version 3 wins over the other two baseline versions by both criteria.

At the level of individual clusters, the syntactic versions are also found to outrival the syntax-blind baselines. Applying a t-test on the score sets for the 5 versions, we can reject the null hypothesis with 99.5% confidence to ensure that the syntactic alignment performs better. Similarly, for hypothesis II, the same is true for the versions with and without stop word alignment. Figures 5 and 6 provide a graphical view of how each alignment version performs on the testing clusters. The clusters along the x-axis are listed in the order of increasing size.

We have also done an analysis on interjudge agreement in the evaluation. The judges are instructed about the evaluation scheme individually, and do their work independently. We do not enforce them to be mutually consistent, as long as they are self-consistent. However, Table 6 shows the mean and standard deviation of human judgments (grammaticality and fidelity) on each version. The small deviation values indicate a fairly high agreement.

Finally, because human evaluation is expensive, we additionally tried to use a language-model ap-

Alignment	Gr. Mean	Gr. StdDev	Fi. Mean	Fi. StdDev
V1	2.74	0.11	2.67	0.43
V2	2.75	0.08	2.69	0.30
V3	3.12	0.07	3.07	0.27
V4	3.74	0.08	3.77	0.16
V5	3.29	0.16	3.23	0.33

Table 6: Mean and standard deviation of human judgments.

proach in the training phase for automatic evaluation of grammaticality. We have used BLEU scores(Papineni et al., 2001), but have observed that they are not consistent with those of human judges. In particular, BLEU assigns too high scores to segmented sentences that are otherwise grammatical. It has been noted in the literature that metrics like BLEU that are solely based on N-grams might not be suitable for checking grammaticality.

5 Conclusion

In this paper, we presented a paraphrase generation method based on multiple sequence alignment which combines traditional dynamic programming techniques with linguistically motivated syntactic information. We apply our work on comparable texts for which syntax has not been successfully explored in alignment by previous work. We showed that using syntactic features improves the quality of the alignment-induced finite state automaton when it is used for generating novel sentences. The strongest syntax guided alignment significantly outperformed all other versions in both grammaticality and fidelity of the novel sentences. In this paper we showed the effectiveness of using syntax in the alignment of structurally diverse comparable texts as needed for text generation.

References

Regina Barzilay and Lillian Lee. 2002. Bootstrapping Lexical Choice via Multiple-Sequence Alignment. In *Proceedings of EMNLP 2002*, Philadelphia.

Regina Barzilay and Lillian Lee. 2003. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. In *Proceedings of NAACL-HLT03*, Edmonton.

Sabine Buchholz. 2000. Readme for perl script chunklink.pl. http://ilk.uvt.nl/ sabine/chunklink/README.html.

Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological Sequence Analysis. Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.

Joseph Felsenstein. 1995. PHYLIP: Phylogeny Inference Package. http://evolution.genetics.washington.edu/phylip.html.

DF. Feng and Russell F. Doolittle. 1987. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4).

Walter M. Fitch and Emanuel Margoliash. 1967. Construction of Phylogenetic Trees. *Science*, 155(3760):279–284, January.

Dan Gusfield, 1997. *Algorithms On Strings: A Dual View from Computer Science and Computational Molecular Biology*. Cambridge University Press.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based Alignment of Multiple Translations: Extracting Paraphrases and Generating New Sentences. In *Proceedings of HLT/NAACL 2003*, Edmonton, Canada.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: A Method for Automatic Evaluation of Machine Translation. Research Report RC22176, IBM.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 142–149, Barcelona, Spain, July. Association for Computational Linguistics.

A Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Phd. Thesis, University of Pennsylvania.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *EACL*, pages 173–179.