Syntax-based Alignment of Multiple Translations: Extracting Paraphrases and Generating New Sentences

Bo Pang
Department of Computer Science
Cornell University
Ithaca, NY 14853 USA
pabo@cs.cornell.edu

Kevin Knight and Daniel Marcu
Information Sciences Institute
University of Southern California
Marina Del Rey, CA 90292 USA
{knight,marcu}@isi.edu

Abstract

We describe a syntax-based algorithm that automatically builds Finite State Automata (word lattices) from semantically equivalent translation sets. These FSAs are good representations of paraphrases. They can be used to extract lexical and syntactic paraphrase pairs and to generate new, unseen sentences that express the same meaning as the sentences in the input sets. Our FSAs can also predict the correctness of alternative semantic renderings, which may be used to evaluate the quality of translations.

1 Introduction

In the past, paraphrases have come under the scrutiny of many research communities. Information retrieval researchers have used paraphrasing techniques for query reformulation in order to increase the recall of information retrieval engines (Sparck Jones and Tait, 1984). Natural language generation researchers have used paraphrasing to increase the expressive power of generation systems (Iordanskaja et al., 1991; Lenke, 1994; Stede, 1999). And researchers in multi-document text summarization (Barzilay et al., 1999), information extraction (Shinyama et al., 2002), and question answering (Lin and Pantel, 2001; Hermjakob et al., 2002) have focused on identifying and exploiting paraphrases in the context of recognizing redundancies, alternative formulations of the same meaning, and improving the performance of question answering systems.

In previous work (Barzilay and McKeown, 2001; Lin and Pantel, 2001; Shinyama et al., 2002), paraphrases are represented as sets or pairs of semantically equivalent words, phrases, and patterns. Although this is adequate in the context of some applications, it is clearly too weak from a generative perspective. Assume, for example, that we know that text pairs (*stock market rose*, *stock market gained*) and (*stock market rose*, *stock prices rose*) have the same meaning. If we memorized only these two pairs, it would be impossible to infer that, in fact, consistent with our intuition, any of the following sets of phrases are also semantically equivalent: {*stock market rose*, *stock market gained*, *stock prices rose*, *stock prices gained* } and {*stock market*, *stock prices* } in the context of *rose* or *gained*; {*market rose* }, {*market gained* }, {*prices rose* } and {*prices gained* } in the context of *stock*; and so on.

In this paper, we propose solutions for two problems: the problem of paraphrase representation and the problem of paraphrase induction. We propose a new, finite-state-based representation of paraphrases that enables one to encode compactly large numbers of paraphrases. We also propose algorithms that automatically derive such representations from inputs that are now routinely released in conjunction with large scale machine translation evaluations (DARPA, 2002): multiple English translations of many foreign language texts. For instance, when given as input the 11 semantically equivalent English translations in Figure 1, our algorithm automatically induces the FSA in Figure 2, which represents compactly 49 distinct renderings of the same semantic meaning. Our FSAs capture both lexical paraphrases, such as {*fighting, battle*}, {*died, were killed*} and structural paraphrases such as {*last week's fighting, the battle of last week*}. The contexts in which these are correct paraphrases are also conveniently captured in the representation.

In previous work, Langkilde and Knight (1998) used word lattices for language generation, but their method involved hand-crafted rules. Bangalore et al. (2001) and Barzilay and Lee (2002) both applied the technique of multi-sequence alignment (MSA) to align parallel corpora and produced similar FSAs. For their purposes, they mainly need to ensure the correctness of consensus among different translations, so that different constituent orderings in input sentences do not pose a serious prob-

1.	At least 12 people were killed in the battle last week.	2.	At least 12 people lost their lives in last week's fighting.
3.	Last week's fight took at least 12 lives.	4.	The fighting last week killed at least 12.
5.	The battle of last week killed at least 12 persons.	6.	At least 12 persons died in the fighting last week.
7.	At least 12 died in the battle last week.	8.	At least 12 people were killed in the fighting last week.
9.	During last week's fighting, at least 12 people died.	10.	Last week at least twelve people died in the fighting.
11.	Last week's fighting took the lives of twelve people.		

Figure 1: Sample Sentence Group from the Chinese-English DARPA Evaluation Corpus: 11 English translations of the same Chinese sentence.

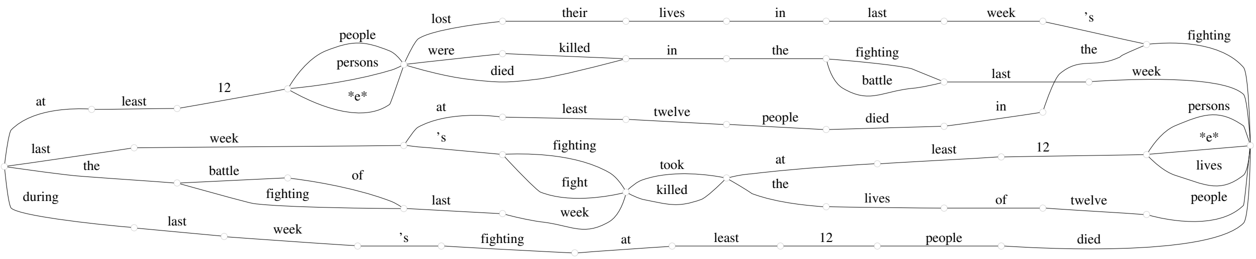


Figure 2: FSA produced by our syntax-based alignment algorithm from the input in Figure 1.

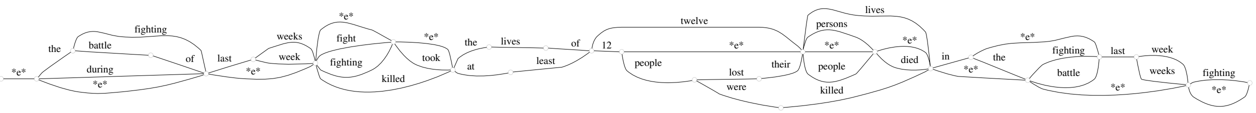


Figure 3: FSA produced by a Multi-Sequence Alignment algorithm from the input in Figure 1.

lem. In contrast, we want to ensure the correctness of all paths represented by the FSAs, and direct application of MSA in the presence of different constituent orderings can be problematic. For example, when given as input the same sentences in Figure 1, one instantiation of the MSA algorithm produces the FSA in Figure 3, which contains many "bad" paths such as *the battle of last week's fighting took at least 12 people lost their people died in the fighting last week's fighting* (See Section 4.2.2 for a more quantitative analysis.). It's still possible to use MSA if, for example, the input is pre-clustered to have the same constituent ordering (Barzilay and Lee (2003)). But we chose to approach this problem from another direction. As a result, we propose a new syntax-based algorithm to produce FSAs.

In this paper, we first introduce the multiple translation corpus that we use in our experiments (see Section 2). We then present the algorithms that we developed to induce finite-state paraphrase representations from such data (see Section 3). An important part of the paper is dedicated to evaluating the quality of the finite-state representations that we derive (see Section 4). Since our representations encode thousands and sometimes millions of equivalent verbalizations of the same meaning, we use both manual and automatic evaluation techniques. Some of the automatic evaluations we perform are novel as well.

2 Data

The data we use in this work is the LDC-available Multiple-Translation Chinese (MTC) Corpus[1] developed for machine translation evaluation, which contains 105 news stories (993 sentences) from three sources of journalistic Mandarin Chinese text. These stories were independently translated into English by 11 translation agencies. Each *sentence group*, which consists of 11 semantically equivalent translations, is a rich source for learning lexical and structural paraphrases. In our experiments, we use 899 of the sentence groups — the sentence groups with sentences longer than 45 words were dropped.

[1]Linguistic Data Consortium (LDC) Catalog Number LDC2002T01, ISBN 1-58563-217-1.

```
1. ParseForest = ϵ
2. foreach s ∈ SentenceGroup
3.         t = parseTree(s);
4.         ParseForest = Merge(ParseForest, t);
5. endfor
6. Extract FSA from ParseForest;
7. Squeeze FSA;
```

Figure 4: The Syntax-Based Alignment Algorithm.

3 A Syntax-Based Alignment Algorithm

Our syntax-based alignment algorithm, whose pseudocode is shown in Figure 4, works in three steps. In the first step (lines 1-5 in Figure 4), we parse every sentence in a sentence group and merge all resulting parse trees into a parse forest. In the second step (line 6), we extract an FSA from the parse forest and then we compact it further using a limited form of bottom-up alignment, which we call squeezing (line 7). In what follows, we describe each step in turn.

Top-down merging. Given a sentence group, we pass each of the 11 sentences to Charniak's (2000) parser to get 11 parse trees. The first step in the algorithm is to merge these parse trees into one parse-forest-like structure using a top-down process.

Let's consider a simple case in which the parse forest contains one single tree, Tree 1 in Figure 5, and we are adding Tree 2 to it. Since the two trees correspond to sentences that have the same meaning and since both trees expand an S node into an NP and a VP, it is reasonable to assume that NP_1 is a paraphrase of NP_2 and VP_1 is a paraphrase of VP_2. We merge NP_1 with NP_2 and VP_1 with VP_2 and continue the merging process on each of the subtrees recursively, until we either reach the leaves of the trees or the two nodes that we examine are expanded using different syntactic rules.

When we apply this process to the trees in Figure 5, the NP nodes are merged all the way down to the leaves, and we get "*12*" as a paraphrase of "*twelve*" and "*people*" as a paraphrase of "*persons*"; in contrast, the two VPs are expanded in different ways, so no merging is done beyond this level, and we are left with the information that "*were killed*" is a paraphrase of "*died*".

We repeat this top-down merging procedure with each of the 11 parse trees in a sentence group. So far, only constituents with same syntactic type are treated as paraphrases. However, later we shall see that we can match word spans whose syntactic types differ.

Keyword checking. The matching process described above appears quite strict – the expansions must match exactly for two nodes to be merged. But consider the following parse trees:

1.(S (NP_1 people)(VP_1 were killed in this battle))
2.(S (NP_2 this battle)(VP_2 killed people))

If we applied the algorithm described above, we would mistakenly align NP_1 with NP_2 and VP_1 with VP_2 — the algorithm described so far makes no use of lexical information.

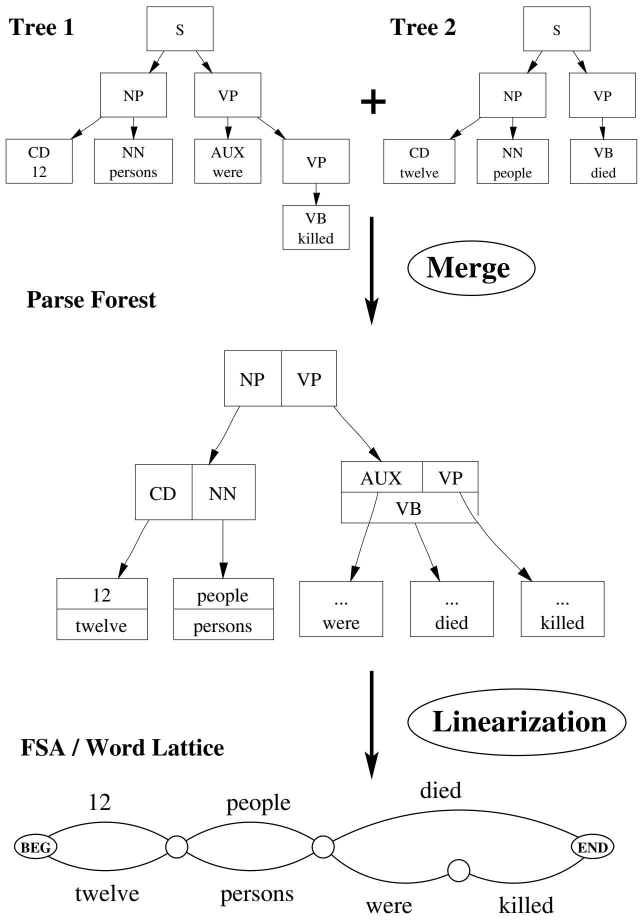
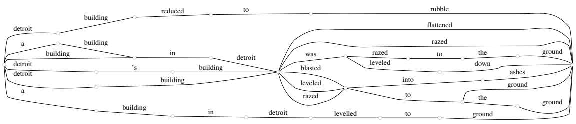


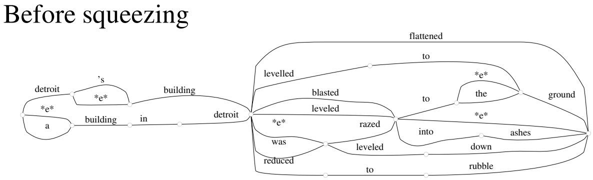
Figure 5: Top-down merging of parse trees and FSA extraction.

To prevent such erroneous alignments, we also implement a simple keyword checking procedure. We note that since the word "*battle*" appears in both VP_1 and NP_2, this can serve as an evidence against the merging of (NP_1, NP_2) and (VP_1, VP_2). A similar argument can be constructed for the word "*people*". So in this example we actually have double evidence against merging; in general, one such clue suffices to stop the merging.

Our keyword checking procedure acts as a filter. A list of keywords is maintained for each node in a syntactic tree. This list contains all the nouns, verbs, and adjectives that are spanned by a syntactic node. Before merging two nodes, we check to see whether the keyword lists asso-



a. Before squeezing



b. After squeezing

Figure 6: Squeezing effect

ciated with them share words with other nodes. That is, supposed we just merged nodes A and B, and they are expanded with the same syntactic rule into $A_1A_2...A_n$ and $B_1B_2...B_n$ respectively; before we merge each A_i with B_i, we check for each B_i if its keyword list shares common words with any A_j $(j \neq i)$. If they do not, we continue the top-down merging process; otherwise we stop. In our current implementation, a pair of synonyms can not stop an otherwise legitimate merging, but it's possible to extend our keyword checking process with the help of lexical resources such as WordNet in future work.

Mapping Parse Forests into Finite State Automata. The process of mapping Parse Forests into Finite State Automata is simple. We simply traverse the parse forest top-down and create alternative paths for every merged node. For example, the parse forest in Figure 5 is mapped into the FSA shown at the bottom of the same figure. In the FSA, there is a word associated with each edge. Different paths between any two nodes are assumed to be paraphrases of each other. Each path that starts from the $BEGIN$ node and ends at the END node corresponds to either an original input sentence or a paraphrase sentence.

Squeezing. Since we adopted a very strict matching criterion in top-down merging, a small difference in the syntactic structure of two trees prevents some legitimate mergings from taking place. This behavior is also exacerbated by errors in syntactic parsing. Hence, for instance, three edges labeled *detroit* at the leftmost of the top FSA in Figure 6 were kept apart. To compensate for this effect, our algorithm implements an additional step, which we call squeezing. If two different edges that go into (or out of) the same node in an FSA are labeled with the same word, the nodes on the other end of the edges are merged. We apply this operation exhaustively over the FSAs produced by the top-down merging procedure. Figure 6 illustrates the effect of this operation: the FSA at the top of this figure is compressed into the more compact FSA shown at the bottom of it. Note that in addition to reducing the redundant edges, this also gives us paraphrases not available in the FSA before squeezing (e.g. $\{$*reduced to rubble, blasted to ground*$\}$). Therefore, the squeezing operation, which implements a limited form of lexically driven alignment similar to that exploited by MSA algorithms, leads to FSAs that have a larger number of paths and paraphrases.

4 Evaluation

The evaluation for our finite state representations and algorithm requires careful examination. Obviously, what counts as a good result largely depends on the application one has in mind. If we are extracting paraphrases for question-reformulation, it doesn't really matter if we output a few syntactically incorrect paraphrases, as long as we produce a large number of semantically correct ones. If we want to use the FSA for MT evaluation (for example, comparing a sentence to be evaluated with the possible paths in FSA), we would want all paths to be relatively good (which we will focus on in this paper), while in some other applications, we may only care about the quality of the best path (not addressed in this paper). Section 4.1 concentrates on evaluating the paraphrase pairs that can be extracted from the FSAs built by our system, while Section 4.2 is dedicated to evaluating the FSAs directly.

4.1 Evaluating paraphrase pairs

4.1.1 Human-based evaluation of paraphrases

By construction, different paths between any two nodes in the FSA representations that we derive are paraphrases (in the context in which the nodes occur). To evaluate our algorithm, we extract paraphrases from our FSAs and ask human judges to evaluate their correctness. We compare the paraphrases we collect with paraphrases that are derivable from the same corpus using a co-training-based paraphrase extraction algorithm (Barzilay and McKeown, 2001). To the best of our knowledge, this is the most relevant work to compare against since it aims at extracting paraphrase pairs from parallel corpus. Unlike our syntax-based algorithm which treats a sentence as a tree structure and uses this hierarchical structural information to guide the merging process, their algorithm treats a sentence as a sequence of phrases with surrounding contexts (no hierarchical structure involved) and co-trains classifiers to detect paraphrases and contexts for paraphrases. It would be interesting to compare the results from two algorithms so different from each other.

For the purpose of this experiment, we randomly selected 300 paraphrase pairs (S_{syn}) from the FSAs pro-

		Correct	Partial	Incorrect
Judge 1	S_{syn}	85%	12%	3%
	S_{cotr}	68%	13%	19%
Judge 2	S_{syn}	80%	13%	7%
	S_{cotr}	63%	13%	24%
Judge 3	S_{syn}	81%	5%	13%
	S_{cotr}	68%	3%	29%
Judge 4	S_{syn}	77%	17%	5%
	S_{cotr}	68%	16%	16%
Average of All Judges	S_{syn}	81%	12%	7%
	S_{cotr}	66%	11%	22%

Table 1: A comparison of the correctness of the paraphrases produced by the syntax-based alignment (S_{syn}) and co-training-based (S_{cotr}) algorithms.

duced by our system. Since the co-training-based algorithm of Barzilay and McKeown (2001) takes parallel corpus as input, we created out of the MTC corpus 55×993 sentence pairs (Each equivalent translation set of cardinality 11 was mapped into $\binom{11}{2}$ equivalent translation pairs.). Regina Barzilay kindly provided us the list of paraphrases extracted by their algorithm from this parallel corpus, from which we randomly selected another set of 300 paraphrases (S_{cotr}).

The resulting 600 paraphrase pairs were mixed and presented in random order to four human judges. Each judge was asked to assess the correctness of 150 paraphrase pairs (75 pairs from each system) based on the context, i.e., the sentence group, from which the paraphrase pair was extracted. Judges were given three choices: "Correct", for perfect paraphrases, "Partially correct", for paraphrases in which there is only a partial overlap between the meaning of two paraphrases (e.g. while {*saving set, aid package*} is a correct paraphrase pair in the given context, {*set, aide package*} is considered partially correct), and "Incorrect". The results of the evaluation are presented in Table 1.

Although the four evaluators were judging four different sets, each clearly rated a higher percentage of the outputs produced by the syntax-based alignment algorithm as "Correct". We should note that there are parameters specific to the co-training algorithm that we did not tune to work for this particular corpus. In addition, the co-training algorithm recovered more paraphrase pairs: the syntax-based algorithm extracted 8666 pairs in total with 1051 of them extracted at least twice (i.e. more or less reliable), while the numbers for the co-training algorithm is 2934 out of a total of 16993 pairs. This means we are not comparing the accuracy on the same recall level.

Aside from evaluating the correctness of the paraphrases, we are also interested in the degree of overlap between the paraphrase pairs discovered by the two algorithms so different from each other. We find that out of the 1051 paraphrase pairs that were extracted from more than one sentence group by the syntax-based algorithm, 62.3% were also extracted by the co-training algorithm; and out of the 2934 paraphrase pairs from the results of co-training algorithm, 33.4% were also extracted by the syntax-based algorithm. This shows that in spite of the very different cues the two different algorithms rely on, they do discover a lot of common pairs.

range of ASL	1-10	10-20	20-30	30-45
recall	30.7%	16.3%	7.8%	3.8%

Table 2: Recall of WordNet-consistent synonyms.

4.1.2 WordNet-based analysis of paraphrases

In order to (roughly) estimate the recall (of lexical synonyms) of our algorithm, we use the synonymy relation in WordNet to extract all the synonym pairs present in our corpus. This extraction process yields the list of all WordNet-consistent synonym pairs that are present in our data. (Note that some of the pairs identified as synonyms by WordNet, like "*follow/be*", are not really synonyms in the contexts defined in our data set, which may lead to artificial deflation of our recall estimate.) Once we have the list of WordNet-consistent paraphrases, we can check how many of them are recovered by our method. Table 2 gives the percentage of pairs recovered for each range of average sentence length (ASL) in the group.

Not surprisingly, we get higher recall with shorter sentences, since long sentences tend to differ in their syntactic structures fairly high up in the parse trees, which leads to fewer mergings at the lexical level. The recall on the task of extracting lexical synonyms, as defined by WordNet, is not high. But after all, this is not what our algorithm has been designed for. It's worth noticing that the syntax-based algorithm also picks up many paraphrases that are not identified as synonyms in WordNet. Out of 3217 lexical paraphrases that are learned by our system, only 493 (15.3%) are WordNet synonyms, which suggests that paraphrasing is a much richer and looser relation than synonymy. However, the WordNet-based recall figures suggest that WordNet can be used as an additional source of information to be exploited by our algorithm.

4.2 Evaluating the FSA directly

We noted before that apart from being a natural representation of paraphrases, the FSAs that we build have their own merit and deserve to be evaluated directly. Since our FSAs contain large numbers of paths, we design automatic evaluation metrics to assess their qualities.

average length	N (# of paths)		$\log N$	
	max	ave	max	ave
1 - 10	22749	775	10.0	5.2
10 - 20	172386	4468	12.1	6.2
20 - 30	3479544	29202	15.1	5.8
30 - 45	684589	4135	13.4	4.5

Table 3: Statistics on Number of Paths in FSAs

random variable	mean	std. dev
$ent(FSA) - ent(SG)$	-0.11586	1.25162
$ent(MTS) - ent(SG)$	1.74259	1.05749

Table 4: Quality judged by LM

4.2.1 Language Model-based evaluation

If we take our claims seriously, each path in our FSAs that connects the start and end nodes should correspond to a well-formed sentence. We are interested in both quantity (how many sentences our automata are able to produce) and quality (how good these sentences are). To answer the first question, we simply count the number of paths produced by our FSAs.

Table 3 gives the statistics on the number of paths produced by our FSAs, reported by the average length of sentences in the input sentence groups. For example, the sentence groups that have between 10 and 20 words produce, on average, automata that can yield 4468 alternative, semantically equivalent formulations.

Note that if we always get the same degree of merging per word across all sentence groups, the number of paths would tend to increase with the sentence length. This is not the case here. Apparently we are getting less merging with longer sentences. But still, given 11 sentences, we are capable of generating hundreds, thousands, and in some cases even millions of sentences.

Obviously, we should not get too happy with our ability to boost the number of equivalent meanings if they are incorrect. To assess the quality of the FSAs generated by our algorithm, we use a language model-based metric.

We train a 4-gram model over one year of the Wall Street Journal using the CMU-Cambridge Statistical Language Modeling toolkit (v2). For each sentence group SG, we use this language model to estimate the average entropy of the 11 original sentences in that group ($ent(SG)$). We also compute the average entropy of all the sentences in the corresponding FSA built by our syntax-based algorithm ($ent(FSA)$). As the statistics in Table 4 show, there is little difference between the average entropy of the original sentences and the average entropy of the paraphrase sentences we produce. To better calibrate this result, we compare it with the average entropy of 6 corresponding machine translation outputs ($ent(MTS)$), which were also made available by LDC in conjunction with the same corpus. As one can see, the difference between the average entropy of the machine produced output and the average entropy of the original 11 sentences is much higher than the difference between the average entropy of the FSA-produced outputs and the average entropy of the original 11 sentences. Obviously, this does not mean that our FSAs only produce well-formed sentences. But it does mean that our FSAs produce sentences that look more like human produced sentences than machine produced ones according to a language model.

4.2.2 Word repetition analysis

Not surprisingly, the language model we used in Section 4.2.1 is far from being a perfect judge of sentence quality. Recall the example of "bad" path we gave in Section 1: *the battle of last week's fighting took at least 12 people lost their people died in the fighting last week's fighting*. Our 4-gram based language model will not find any fault with this sentence. Notice, however, that some words (such as "fighting" and "people") appear at least twice in this path, although they are not repeated in any of the source sentences. These erroneous repetitions indicate mis-alignment. By measuring the frequency of words that are mistakenly repeated, we can now examine quantitatively whether a direct application of the MSA algorithm suffers from different constituent orderings as we expected.

For each sentence group, we get a list of words that never appear more than once in any sentence in this group. Given a word from this list and the FSA built from this group, we count the total number of paths that contain this word (C) and the number of paths in which this word appears at least twice (C_r, i.e. number of erroneous repetitions). We define the *repetition ratio* to be C_r/C, which is the proportion of "bad" paths in this FSA according to this word. If we compute this ratio for all the words in the lists of the first 499 groups[2] and the corresponding FSAs produced by an instantiation of the MSA algorithm[3], the average repetition ratio is 0.0304992 (14.76% of the words have a non-zero repetition ratio, and the average ratio for these words is 0.206671). In comparison, the average repetition ratio for our algorithm is 0.0035074 (2.16% of the words have a non-zero repetition ratio[4], and the average ratio for these

[2] MSA runs very slow for longer sentences, and we believe using the first 499 groups should be enough to make our point.

[3] We thank Regina Barzilay for providing us this set of results

[4] Note that FSAs produced right after keyword checking will not yield any non-zero repetition ratio. However, if there are mis-alignment not prevented by keyword checking in an FSA, it may contain paths with erroneous repetition of words after squeezing.

words is 0.162309). The presence of different constituent orderings does pose a more serious problem to the MSA algorithm.

4.2.3 MT-based evaluation

Recently, Papineni et al. (2002) have proposed an automatic MT system evaluation technique (the BLEU score). Given an MT system output and a set of reference translations, one can estimate the "goodness" of the MT output by measuring the n-gram overlap between the output and the reference set. The higher the overlap, i.e., the closer an output string is to a set of reference translations, the better a translation it is.

We hypothesize that our FSAs provide a better representation against which the outputs of MT systems can be evaluated because they encode not just a few but thousands of equivalent semantic formulations of the desired meaning. Ideally, if the FSAs we build accept all and only the correct renderings of a given meaning, we can just give a test sentence to the reference FSA and see if it is accepted by it. Since this is not a realistic expectation, we measure the edit distance between a string and an FSA instead: the smaller this distance is, the closer it is to the meaning represented by the FSA.

To assess whether our FSAs are more appropriate representations for evaluating the output of MT systems, we perform the following experiment. For each sentence group, we hold out one sentence as test sentence, and try to evaluate how much of it can be predicted from the other 10 sentences. We compare two different ways of estimating the predictive power. (a) we compute the edit distance between the test sentence and the other 10 sentences in the set. The minimum of this distance is $ed(input)$. (b) we use dynamic programming to efficiently compute the minimum distance ($ed(FSA)$) between the test sentence and all the paths in the FSA built from the other 10 sentences. The smaller the edit distance is, the better we are predicting a test sentence. Mathematically, the difference between these two measures $ed(input) - ed(FSA)$ characterizes how much is gained in predictive power by building the FSA.

We carry out the experiment described above in a "leave-one-out" fashion (i.e. each sentence serves as a test sentence once). Now let ed_{gain} be the average of $ed(input) - ed(FSA)$ over the 11 runs for a given group. We compute this for all 899 groups and find the mean for ed_{gain} to be 0.91 (std. dev = 0.78). Table 5 gives the count for groups whose ed_{gain} falls into the specified range. We can see that the majority of ed_{gain} falls under 2.

We are also interested in the relation between the predictive power of the FSAs and the number of reference translations they are derived from. For a given group, we randomly order the sentences in it, set the last one as the test sentence, and try to predict it with the first 1, 2, 3, ... 10 sentences. We investigate whether more sentences yield an increase in the predictive power.

range	0-1	1-2	2-3	3-4	4-5
count	546	256	80	15	2

Table 5: Statistics for ed_{gain}

	$ed(FSA_n)$ $-ed(FSA_{10})$		$ed(input_n)$ $-ed(FSA_n)$	
n	mean	std. dev	mean	std. dev
1	5.65	3.86	0	0
2	3.66	3.02	0.19	0.60
3	2.71	2.55	0.33	0.76
4	2.10	2.33	0.46	0.90
5	1.56	2.01	0.56	0.95
6	1.18	1.79	0.65	1.02
7	0.79	1.48	0.75	1.09
8	0.49	1.10	0.81	1.11
9	0.21	0.74	0.89	1.16
10	0	0	0.93	1.21

Table 6: Effect of monotonically increasing the number of reference sentences

Let $ed(FSA_n)$ be the edit distance from the test sentence to the FSA built on the first n sentences; similarly, let $ed(input_n)$ be the minimum edit distance from the test sentence to an input set that consists of only the first n sentences. Table 6 reports the effect of using different number of reference translations. The first column shows that each translation is contributing to the predictive power of our FSA. Even when we add the tenth translation to our FSA, we still improve its predictive power. The second column shows that the more sentences we add to the FSA the larger the difference between its predictive power and that of a simple set. The results in Table 6 suggest that our FSA may be used in order to refine the BLEU metric (Papineni et al., 2002).

5 Conclusion & Future Work

In this paper, we presented a new syntax-based algorithm that learns paraphrases from a newly available dataset. The multiple translation corpus that we use in this paper is the first instance in a series of similar corpora that are built and made publicly available by LDC in the context of a series of DARPA-sponsored MT evaluations. The algorithm we proposed constructs finite state representations of paraphrases that are useful in many contexts: to induce large lists of lexical and structural paraphrases; to generate semantically equivalent renderings of a given meaning; and to estimate the quality of machine translation systems. More experiments need to be carried out in order to assess extrinsically whether the FSAs we produce can be used to yield higher agreement scores between human and automatic assessments of translation quality.

In our future work, we wish to experiment with more flexible merging algorithms and to integrate better the top-down and bottom-up processes that are used to induce FSAs. We also wish to extract more abstract paraphrase patterns from the current representation. Such patterns are more likely to get reused – which would help us get reliable statistics for them in the extraction phase, and also have a better chance of being applicable to unseen data.

Acknowledgments

We thank Hal Daumé III, Ulrich Germann, and Ulf Hermjakob for help and discussions; Eric Breck, Hubert Chen, Stephen Chong, Dan Kifer, and Kevin O'Neill for participating in the human evaluation; and the Cornell NLP group and the reviewers for their comments on this paper. We especially want to thank Regina Barzilay and Lillian Lee for many valuable suggestions and help at various stages of this work. Portions of this work were done while the first author was visiting Information Sciences Institute. This work was supported by the Advanced Research and Development Activity (ARDA)'s Advance Question Answering for Intelligence (AQUAINT) Program under contract number MDA908-02-C-0007, the National Science Foundation under ITR/IM grant IIS-0081334 and a Sloan Research Fellowship to Lillian Lee. Any opinions, findings, and conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of the National Science Foundation or the Sloan Foundation.

References

Srinivas Bangalore, German Bordel, and Giuseppe Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *Workshop on Automatic Speech Recognition and Understanding*.

Regina Barzilay and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 164–171.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of HLT/NAACL*.

Regina Barzilay and Kathleen McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the ACL/EACL*, pages 50–57.

Regina Barzilay, Kathleen McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the ACL*, pages 550–557.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the NAACL*.

DARPA. 2002. In *DARPA IAO Machine Translation Workshop*, Santa Monica, CA, July 22-23.

Ulf Hermjakob, Abdessamad Echihabi, and Daniel Marcu. 2002. Natural language based reformulation resource and web exploitation for question answering. In *Proceedings of the Text Retrieval Conference (TREC–2002)*. November.

Lidija Iordanskaja, Richard Kittredge, and Alain Polgére. 1991. Lexical selection and paraphrase in a meaning-text generation model. In Cécile L. Paris, William R. Swartout, and William C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 293–312. Kluwer Academic Publisher.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of of ACL/COLING*.

Nils Lenke. 1994. Anticipating the reader's problems and the automatic generation of paraphrases. In *Proceedings of the 15th International Conference on Computational Linguistics*, volume 1, pages 319–323, Kyoto, Japan, August 5–9.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2001*, pages 323–328.

Kishore Papineni, Salim Roukos, Todd Ward, John Henderson, and Florence Reeder. 2002. Corpus-based comprehensive and diagnostic MT evaluation: Initial Arabic, Chinese, French, and Spanish results. In *Proceedings of the Human Language Technology Conference*, pages 124–127, San Diego, CA, March 24-27.

Yusuke Shinyama, Satoshi Sekine, Kiyoshi Sudo, and Ralph Grishman. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of the Human Language Technology Conference (HLT–02)*, San Diego, CA, March 24-27. Poster presentation.

Karen Sparck Jones and John I. Tait. 1984. Automatic search term variant generation. *Journal of Documentation*, 40(1):50–66.

Manfred Stede. 1999. *Lexical Semantics and Knowledge Representation in Multilingual Text Generation*. Kluwer Academic Publishers, Boston/Dordrecht/London.