Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment

Regina Barzilay and **Lillian Lee**
Department of Computer Science
Cornell University
Ithaca, NY 14853-7501
{regina,llee}@cs.cornell.edu

Abstract

We address the text-to-text generation problem of sentence-level paraphrasing — a phenomenon distinct from and more difficult than word- or phrase-level paraphrasing. Our approach applies *multiple-sequence alignment* to sentences gathered from unannotated comparable corpora: it learns a set of paraphrasing patterns represented by *word lattice* pairs and automatically determines how to apply these patterns to rewrite new sentences. The results of our evaluation experiments show that the system derives accurate paraphrases, outperforming baseline systems.

1 Introduction

This is a late parrot! It's a stiff! Bereft of life, it rests in peace! If you hadn't nailed him to the perch he would be pushing up the daisies! Its metabolical processes are of interest only to historians! It's hopped the twig! It's shuffled off this mortal coil! It's rung down the curtain and joined the choir invisible! This is an EX-PARROT! — Monty Python, "Pet Shop"

A mechanism for automatically generating multiple paraphrases of a given sentence would be of significant practical import for text-to-text generation systems. Applications include summarization (Knight and Marcu, 2000) and rewriting (Chandrasekar and Bangalore, 1997): both could employ such a mechanism to produce candidate sentence paraphrases that other system components would filter for length, sophistication level, and so forth.[1] Not surprisingly, therefore, paraphrasing has been a focus of generation research for quite some time (McKeown, 1979; Meteer and Shaked, 1988; Dras, 1999).

One might initially suppose that sentence-level paraphrasing is simply the result of word-for-word or phrase-by-phrase substitution applied in a domain- and context-independent fashion. However, in studies of paraphrases across several domains (Iordanskaja et al., 1991; Robin, 1994; McKeown et al., 1994), this was generally not the case. For instance, consider the following two sentences (similar to examples found in Smadja and McKeown (1991)):

After the latest Fed rate cut, stocks rose across the board.
Winners strongly outpaced losers after Greenspan cut interest rates again.

Observe that "Fed" (Federal Reserve) and "Greenspan" are interchangeable only in the domain of US financial matters. Also, note that one cannot draw one-to-one correspondences between single words or phrases. For instance, nothing in the second sentence is really equivalent to "across the board"; we can only say that the entire clauses "stocks rose across the board" and "winners strongly outpaced losers" are paraphrases. This evidence suggests two consequences: (1) we cannot rely solely on generic domain-independent lexical resources for the task of paraphrasing, and (2) *sentence-level* paraphrasing is an important problem extending beyond that of paraphrasing smaller lexical units.

Our work presents a novel knowledge-lean algorithm that uses multiple-sequence alignment *(MSA) to* learn *to generate sentence-level paraphrases essentially from unannotated corpus data alone.* In contrast to previous work using MSA for generation (Barzilay and Lee,

[1] Another interesting application, somewhat tangential to generation, would be to expand existing corpora by providing several versions of their component sentences. This could, for example, aid machine-translation evaluation, where it has become common to evaluate systems by comparing their output against a bank of several reference translations for the same sentences (Papineni et al., 2002). See Bangalore et al. (2002) and Barzilay and Lee (2002) for other uses of such data.

2002), we need neither parallel data nor explicit information about sentence semantics. Rather, we use two *comparable corpora*, in our case, collections of articles produced by two different newswire agencies about the same events. The use of related corpora is key: we can capture paraphrases that on the surface bear little resemblance but that, by the nature of the data, must be descriptions of the same information. Note that we also acquire paraphrases from each of the individual corpora; but the lack of clues as to sentence equivalence in single corpora means that we must be more conservative, only selecting as paraphrases items that are structurally very similar.

Our approach has three main steps. First, working on each of the comparable corpora separately, we compute *lattices* — compact graph-based representations — to find commonalities within (automatically derived) groups of structurally similar sentences. Next, we identify pairs of lattices from the two different corpora that are paraphrases of each other; the identification process checks whether the lattices take similar arguments. Finally, given an input sentence to be paraphrased, we match it to a lattice and use a paraphrase from the matched lattice's mate to generate an output sentence. The key features of this approach are:

Focus on paraphrase generation. In contrast to earlier work, we not only extract paraphrasing rules, but also automatically determine which of the potentially relevant rules to apply to an input sentence and produce a revised form using them.

Flexible paraphrase types. Previous approaches to paraphrase acquisition focused on certain rigid types of paraphrases, for instance, limiting the number of arguments. In contrast, our method is not limited to a set of *a priori*-specified paraphrase types.

Use of comparable corpora and minimal use of knowledge resources. In addition to the advantages mentioned above, comparable corpora can be easily obtained for many domains, whereas previous approaches to paraphrase acquisition (and the related problem of phrase-based machine translation (Wang, 1998; Och et al., 1999; Vogel and Ney, 2000)) required parallel corpora. We point out that one such approach, recently proposed by Pang et al. (2003), also represents paraphrases by lattices, similarly to our method, although their lattices are derived using parse information.

Moreover, our algorithm does not employ knowledge resources such as parsers or lexical databases, which may not be available or appropriate for all domains — a key issue since paraphrasing is typically domain-dependent. Nonetheless, our algorithm achieves good performance.

2 Related work

Previous work on automated paraphrasing has considered different levels of paraphrase granularity. Learning synonyms via distributional similarity has been well-studied (Pereira et al., 1993; Grefenstette, 1994; Lin, 1998). Jacquemin (1999) and Barzilay and McKeown (2001) identify phrase-level paraphrases, while Lin and Pantel (2001) and Shinyama et al. (2002) acquire structural paraphrases encoded as templates. These latter are the most closely related to the sentence-level paraphrases we desire, and so we focus in this section on template-induction approaches.

Lin and Pantel (2001) extract inference rules, which are related to paraphrases (for example, X wrote Y implies X is the author of Y), to improve question answering. They assume that *paths* in dependency trees that take similar arguments (leaves) are close in meaning. However, only two-argument templates are considered. Shinyama et al. (2002) also use dependency-tree information to extract templates of a limited form (in their case, determined by the underlying information extraction application). Like us (and unlike Lin and Pantel, who employ a single large corpus), they use articles written about the same event in different newspapers as data.

Our approach shares two characteristics with the two methods just described: pattern comparison by analysis of the patterns' respective arguments, and use of non-parallel corpora as a data source. However, *extraction* methods are not easily extended to *generation* methods. One problem is that their templates often only match small fragments of a sentence. While this is appropriate for other applications, deciding whether to use a given template to generate a paraphrase requires information about the surrounding context provided by the entire sentence.

3 Algorithm

Overview We first sketch the algorithm's broad outlines. The subsequent subsections provide more detailed descriptions of the individual steps.

The major goals of our algorithm are to learn:

- recurring patterns in the data, such as X (injured/wounded) Y people, Z seriously, where the capital letters represent variables;
- pairings between such patterns that represent paraphrases, for example, between the pattern X (injured/wounded) Y people, Z of them seriously and the pattern Y were (wounded/hurt) by X, among them Z were in serious condition.

Figure 1 illustrates the main stages of our approach. During training, pattern induction is first applied independently to the two datasets making up a pair of comparable corpora. Individual patterns are learned by applying

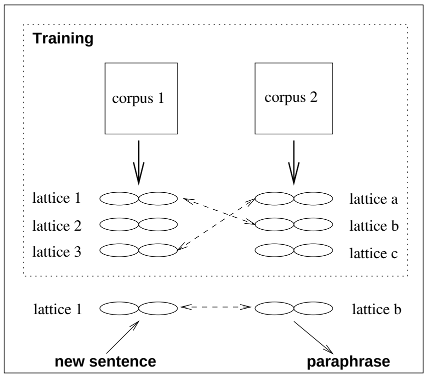

Figure 1: System architecture.

(1) **A Palestinian suicide bomber blew himself up in** a southern city Wednesday, **killing** two other **people and wounding** 27.
(2) **A suicide bomber blew himself up in** the settlement of Efrat, on Sunday, **killing** himself **and injuring** seven people.
(3) **A suicide bomber blew himself up in** the coastal resort of Netanya on Monday, **killing** three other **people and wounding** dozens more.
(4) **A Palestinian suicide bomber blew himself up in** a garden cafe on Saturday, **killing** 10 **people and wounding** 54.
(5) **A suicide bomber blew himself up in** the centre of Netanya on Sunday, **killing** three **people** as well as himself **and injuring** 40.

Figure 2: Five sentences (without date, number, and name substitution) from a cluster of 49, similarities emphasized.

multiple-sequence alignment to clusters of sentences describing approximately similar events; these patterns are represented compactly by *lattices* (see Figure 3). We then check for lattices from the two different corpora that tend to take the same arguments; these lattice pairs are taken to be paraphrase patterns.

Once training is done, we can generate paraphrases as follows: given the sentence "The surprise bombing injured twenty people, five of them seriously", we match it to the lattice X (injured/wounded) Y people, Z of them seriously which can be rewritten as Y were (wounded/hurt) by X, among them Z were in serious condition, and so by substituting arguments we can generate "Twenty were wounded by the surprise bombing, among them five were in serious condition" or "Twenty were hurt by the surprise bombing, among them five were in serious condition".

3.1 Sentence clustering

Our first step is to cluster sentences into groups from which to learn useful patterns; for the multiple-sequence techniques we will use, this means that the sentences within clusters should describe similar events and have similar structure, as in the sentences of Figure 2. This is accomplished by applying hierarchical complete-link clustering to the sentences using a similarity metric based on word n-gram overlap ($n = 1, 2, 3, 4$). The only subtlety is that we do not want mismatches on sentence details (e.g., the location of a raid) causing sentences describing the same type of occurrence (e.g., a raid) from being separated, as this might yield clusters too fragmented for effective learning to take place. (Moreover, variability in the *arguments* of the sentences in a cluster is needed for our learning algorithm to succeed; see below.) We therefore first replace all appearances of dates, numbers, and proper names[2] with generic tokens. Clusters with fewer than ten sentences are discarded.

3.2 Inducing patterns

In order to learn patterns, we first compute a *multiple-sequence alignment* (MSA) of the sentences in a given cluster. Pairwise MSA takes two sentences and a scoring function giving the similarity between words; it determines the highest-scoring way to perform insertions, deletions, and changes to transform one of the sentences into the other. Pairwise MSA can be extended efficiently to multiple sequences via the iterative pairwise alignment, a polynomial-time method commonly used in computational biology (Durbin et al., 1998).[3] The results can be represented in an intuitive form via a word *lattice* (see Figure 3), which compactly represents (n-gram) structural similarities between the cluster's sentences.

To transform lattices into generation-suitable patterns requires some understanding of the possible varieties of lattice structures. The most important part of the transformation is to determine which words are actually instances of arguments, and so should be replaced by *slots* (representing variables). The key intuition is that because the sentences in the cluster represent the same *type* of event, such as a bombing, but generally refer to different *instances* of said event (e.g. a bombing in Jerusalem versus in Gaza), areas of large variability in the lattice should correspond to arguments.

To quantify this notion of variability, we first formalize its opposite: commonality. We define *backbone* nodes

[2]Our crude proper-name identification method was to flag every phrase (extracted by a noun-phrase chunker) appearing capitalized in a non-sentence-initial position sufficiently often.

[3]Scoring function: aligning two identical words scores 1; inserting a word scores -0.01, and aligning two different words scores -0.5 (parameter values taken from Barzilay and Lee (2002)).

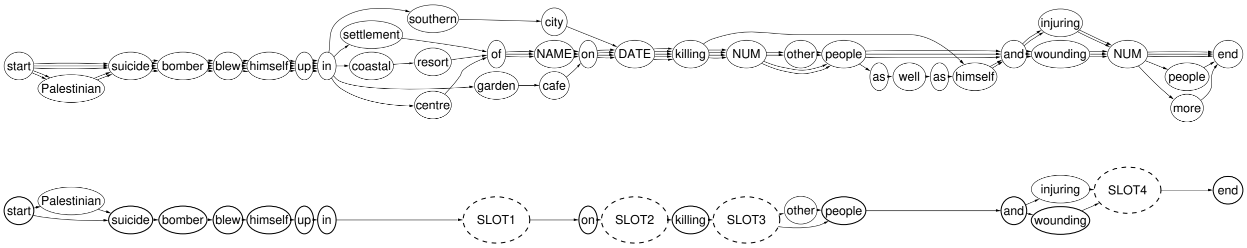

Figure 3: Lattice and slotted lattice for the five sentences from Figure 2. Punctuation and articles removed for clarity.

as those shared by more than 50% of the cluster's sentences. The choice of 50% is not arbitrary — it can be proved using the pigeonhole principle that our strict-majority criterion imposes a unique linear ordering of the backbone nodes that respects the word ordering within the sentences, thus guaranteeing at least a degree of well-formedness and avoiding the problem of how to order backbone nodes occurring on parallel "branches" of the lattice.

Once we have identified the backbone nodes as points of strong commonality, the next step is to identify the regions of variability (or, in lattice terms, many parallel disjoint paths) between them as (probably) corresponding to the arguments of the propositions that the sentences represent. For example, in the top of Figure 3, the words "southern city, "settlement of NAME","coastal resort of NAME", etc. all correspond to the location of an event and could be replaced by a single slot. Figure 3 shows an example of a lattice and the derived slotted lattice; we give the details of the slot-induction process in the Appendix.

3.3 Matching lattices

Now, if we were using a parallel corpus, we could employ sentence-alignment information to determine which lattices correspond to paraphrases. Since we do not have this information, we essentially approximate the parallel-corpus situation by correlating information from descriptions of (what we hope are) the same event occurring in the two different corpora.

Our method works as follows. Once lattices for each corpus in our comparable-corpus pair are computed, we identify lattice paraphrase pairs, using the idea that paraphrases will tend to take the same values as arguments (Shinyama et al., 2002; Lin and Pantel, 2001). More specifically, we take a pair of lattices from different corpora, look back at the sentence clusters from which the two lattices were derived, and compare the slot values of those cross-corpus sentence pairs that appear in articles written on the *same day* on the same topic; we pair the lattices if the degree of matching is over a threshold tuned on held-out data. For example, suppose we have two (linearized) lattices slot1 bombed slot2 and slot3 was bombed by slot4 drawn from different corpora. If in the first lattice's sentence cluster we have the sentence "the plane bombed the town", and in the second lattice's sentence cluster we have a sentence written on the same day reading "the town was bombed by the plane", then the corresponding lattices may well be paraphrases, where slot1 is identified with slot4 and slot2 with slot3.

To compare the set of argument values of two lattices, we simply count their word overlap, giving double weight to proper names and numbers and discarding auxiliaries (we purposely ignore order because paraphrases can consist of word re-orderings).

3.4 Generating paraphrase sentences

Given a sentence to paraphrase, we first need to identify which, if any, of our previously-computed sentence clusters the new sentence belongs most strongly to. We do this by finding the best alignment of the sentence to the existing lattices.[4] If a matching lattice is found, we choose one of its comparable-corpus paraphrase lattices to rewrite the sentence, substituting in the argument values of the original sentence. This yields as many paraphrases as there are lattice paths.

4 Evaluation

All evaluations involved judgments by native speakers of English who were not familiar with the paraphrasing systems under consideration.

We implemented our system on a pair of comparable corpora consisting of articles produced between September 2000 and August 2002 by the Agence France-Presse (AFP) and Reuters news agencies. Given our interest in domain-dependent paraphrasing, we limited attention to 9MB of articles, collected using a TDT-style document clustering system, concerning individual acts of violence in Israel and army raids on the Palestinian territories. From this data (after removing 120 articles as a held-

[4]To facilitate this process, we add "insert" nodes between backbone nodes; these nodes can match any word sequence and thus account for new words in the input sentence. Then, we perform multiple-sequence alignment where insertions score -0.1 and all other node alignments receive a score of unity.

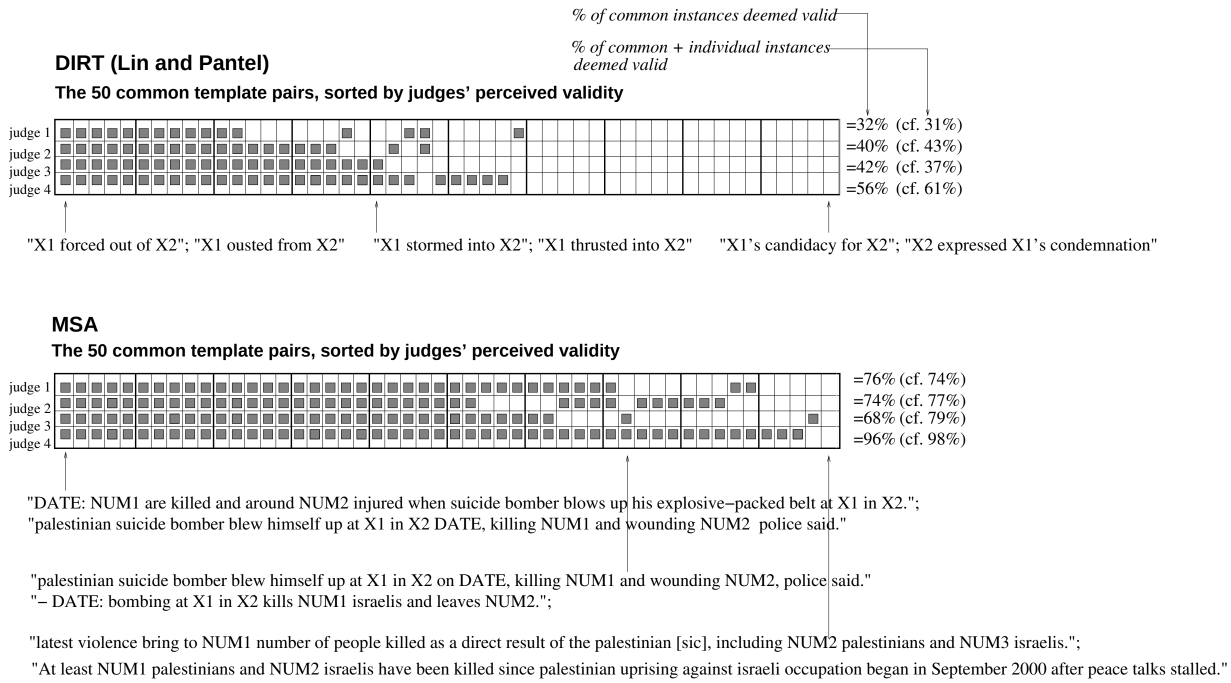

Figure 4: Correctness and agreement results. Columns = instances; each grey box represents a judgment of "valid" for the instance. For each method, a good, middling, and poor instance is shown. (Results separated by algorithm for clarity; the blind evaluation presented instances from the two algorithms in random order.)

out parameter-training set), we extracted 43 slotted lattices from the AFP corpus and 32 slotted lattices from the Reuters corpus, and found 25 cross-corpus matching pairs; since lattices contain multiple paths, these yielded 6,534 template pairs.[5]

4.1 Template Quality Evaluation

Before evaluating the quality of the rewritings produced by our templates and lattices, we first tested the quality of a random sample of just the template pairs. In our instructions to the judges, we defined two text units (such as sentences or snippets) to be paraphrases if one of them can generally be substituted for the other without great loss of information (but not necessarily vice versa). [6] Given a pair of *templates* produced by a system, the judges marked them as paraphrases if for many instantiations of the templates' variables, the resulting text units were paraphrases. (Several labelled examples were provided to supply further guidance).

To put the evaluation results into context, we wanted to compare against another system, but we are not aware of any previous work creating templates precisely for the task of generating paraphrases. Instead, we made a good-faith effort to adapt the DIRT system (Lin and Pantel, 2001) to the problem, selecting the 6,534 highest-scoring templates it produced when run on our datasets. (The system of Shinyama et al. (2002) was unsuitable for evaluation purposes because their paraphrase extraction component is too tightly coupled to the underlying information extraction system.) It is important to note some important caveats in making this comparison, the most prominent being that DIRT was not designed with sentence-paraphrase generation in mind — its templates are much shorter than ours, which may have affected the evaluators' judgments — and was originally implemented on much larger data sets.[7] The point of this evaluation is simply to determine whether another corpus-based paraphrase-focused approach could easily achieve the same performance level.

In brief, the DIRT system works as follows. Dependency trees are constructed from parsing a large corpus. Leaf-to-leaf paths are extracted from these dependency

[5] The extracted paraphrases are available at `http://www.cs.cornell.edu/Info/Projects/NLP/statpar.html`

[6] We switched to this "one-sided" definition because in initial tests judges found it excruciating to decide on equivalence. Also, in applications such as summarization some information loss is acceptable.

[7] To cope with the corpus-size issue, DIRT was trained on an 84MB corpus of Middle-East news articles, a strict superset of the 9MB we used. Other issues include the fact that DIRT's output needed to be converted into English: it produces paths like "N:of:N ⟨tide⟩ N:nn:N", which we transformed into "Y tide of X" so that its output format would be the same as ours.

trees, with the leaves serving as slots. Then, pairs of paths in which the slots tend to be filled by similar values, where the similarity measure is based on the mutual information between the value and the slot, are deemed to be paraphrases.

We randomly extracted 500 pairs from the two algorithms' output sets. Of these, 100 paraphrases (50 per system) made up a "common" set evaluated by all four judges, allowing us to compute agreement rates; in addition, each judge also evaluated another "individual" set, seen only by him- or herself, consisting of another 100 pairs (50 per system). The "individual" sets allowed us to broaden our sample's coverage of the corpus.[8] The pairs were presented in random order, and the judges were not told which system produced a given pair.

As Figure 4 shows, our system outperforms the DIRT system, with a consistent performance gap for all the judges of about 38%, although the absolute scores vary (for example, Judge 4 seems lenient). The judges' assessment of correctness was fairly constant between the full 100-instance set and just the 50-instance common set alone.

In terms of agreement, the Kappa value (measuring pairwise agreement discounting chance occurrences[9]) on the common set was 0.54, which corresponds to moderate agreement (Landis and Koch, 1977). Multiway agreement is depicted in Figure 4 — there, we see that in 86 of 100 cases, at least three of the judges gave the same correctness assessment, and in 60 cases all four judges concurred.

4.2 Evaluation of the generated paraphrases

Finally, we evaluated the quality of the paraphrase sentences generated by our system, thus (indirectly) testing all the system components: pattern selection, paraphrase acquisition, and generation. We are not aware of another system generating sentence-level paraphrases. Therefore, we used as a baseline a simple paraphrasing system that just replaces words with one of their randomly-chosen WordNet synonyms (using the most frequent sense of the word that WordNet listed synonyms for). The number of substitutions was set proportional to the number of words our method replaced in the same sentence. The point of this comparison is to check whether simple synonym substitution yields results comparable to those of our algorithm. [10]

For this experiment, we randomly selected 20 AFP articles about violence in the Middle East published later than the articles in our training corpus. Out of 484 sentences in this set, our system was able to paraphrase 59 (12.2%). (We chose parameters that optimized precision rather than recall on our small held-out set.) We found that after proper name substitution, only seven sentences in the test set appeared in the training set,[11] which implies that lattices boost the generalization power of our method significantly: from seven to 59 sentences. Interestingly, the coverage of the system varied significantly with article length. For the eight articles of ten or fewer sentences, we paraphrased 60.8% of the sentences per article on average, but for longer articles only 9.3% of the sentences per article on average were paraphrased. Our analysis revealed that long articles tend to include large portions that are unique to the article, such as personal stories of the event participants, which explains why our algorithm had a lower paraphrasing rate for such articles.

All 118 instances (59 per system) were presented in random order to two judges, who were asked to indicate whether the meaning had been preserved. Of the paraphrases generated by our system, the two evaluators deemed 81.4% and 78%, respectively, to be valid, whereas for the baseline system, the correctness results were 69.5% and 66.1%, respectively. Agreement according to the Kappa statistic was 0.6. Note that judging full sentences is inherently easier than judging templates, because template comparison requires considering a variety of possible slot values, while sentences are self-contained units.

Figure 5 shows two example sentences, one where our MSA-based paraphrase was deemed correct by both judges, and one where both judges deemed the MSA-generated paraphrase incorrect. Examination of the results indicates that the two systems make essentially orthogonal types of errors. The baseline system's relatively poor performance supports our claim that whole-sentence paraphrasing is a hard task even when accurate word-level paraphrases are given.

5 Conclusions

We presented an approach for generating sentence level paraphrases, a task not addressed previously. Our method learns structurally similar patterns of expression from data and identifies paraphrasing pairs among them using a comparable corpus. A flexible pattern-matching procedure allows us to paraphrase an unseen sentence by

[8] Each judge took several hours at the task, making it infeasible to expand the sample size further.

[9] One issue is that the Kappa statistic doesn't account for varying difficulty among instances. For this reason, we actually asked judges to indicate for each instance whether making the validity decision was difficult. However, the judges generally did not agree on difficulty. Post hoc analysis indicates that perception of difficulty depends on each judge's individual "threshold of similarity", not just the instance itself.

[10] We chose not to employ a language model to re-rank either system's output because such an addition would make it hard to isolate the contribution of the paraphrasing component itself.

[11] Since we are doing unsupervised paraphrase acquisition, train-test overlap is allowed.

Original (1)	*The caller identified the bomber as Yussef Attala, 20, from the Balata refugee camp near Nablus.*
MSA	The caller named the bomber as 20-year old Yussef Attala from the Balata refugee camp near Nablus.
Baseline	The company placed the bomber as Yussef Attala, 20, from the Balata refugee camp near Nablus.
Original (2)	*A spokesman for the group claimed responsibility for the attack in a phone call to AFP in this northern West Bank town.*
MSA	The attack in a phone call to AFP in this northern West Bank town was claimed by a spokesman of the group.
Baseline	A spokesman for the grouping laid claim responsibility for the onslaught in a phone call to AFP in this northern West Bank town.

Figure 5: Example sentences and generated paraphrases. Both judges felt MSA preserved the meaning of (1) but not (2), and that neither baseline paraphrase was meaning-preserving.

matching it to one of the induced patterns. Our approach generates both lexical and structural paraphrases.

Another contribution is the induction of MSA lattices from non-parallel data. Lattices have proven advantageous in a number of NLP contexts (Mangu et al., 2000; Bangalore et al., 2002; Barzilay and Lee, 2002; Pang et al., 2003), but were usually produced from (multi-)parallel data, which may not be readily available for many applications. We showed that word lattices can be induced from a type of corpus that can be easily obtained for many domains, broadening the applicability of this useful representation.

Acknowledgments

We are grateful to many people for helping us in this work. We thank Stuart Allen, Itai Balaban, Hubie Chen, Tom Heyerman, Evelyn Kleinberg, Carl Sable, and Alex Zubatov for acting as judges. Eric Breck helped us with translating the output of the DIRT system. We had numerous very useful conversations with all those mentioned above and with Eli Barzilay, Noemie Elhadad, Jon Kleinberg (who made the "pigeonhole" observation), Mirella Lapata, Smaranda Muresan and Bo Pang. We are very grateful to Dekang Lin for providing us with DIRT's output. We thank the Cornell NLP group, especially Eric Breck, Claire Cardie, Amanda Holland-Minkley, and Bo Pang, for helpful comments on previous drafts. This paper is based upon work supported in part by the National Science Foundation under ITR/IM grant IIS-0081334 and a Sloan Research Fellowship. 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, Vanessa Murdock, and Giuseppe Riccardi. 2002. Bootstrapping bilingual data using consensus translation for a multilingual instant messaging system. In *Proc. of COLING*.

Regina Barzilay and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *Proc. of EMNLP*, pages 164–171.

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

Raman Chandrasekar and Srinivas Bangalore. 1997. Automatic induction of rules for text simplification. *Knowledge-Based Systems*, 10(3):183–190.

Mark Dras. 1999. *Tree Adjoining Grammar and the Reluctant Paraphrasing of Text*. Ph.D. thesis, Macquarie University.

Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological Sequence Analysis*. Cambridge University Press, Cambridge, UK.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*, volume 278. Kluwer.

L. Iordanskaja, R. Kittredge, and A. Polguere. 1991. Lexical selection and paraphrase in a meaning-text generation model. In C. Paris, W. Swartout, and W. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, chapter 11. Kluwer.

Christian Jacquemin. 1999. Syntagmatic and paradigmatic representations of term variations. In *Proc. of the ACL*, pages 341–349.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization — Step one: Sentence compression. In *Proc. of AAAI*.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33:159–174.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proc. of ACL/COLING*, pages 768–774.

Lidia Mangu, Eric Brill, and Andreas Stolcke. 2000. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Computer, Speech and Language*, 14(4):373–400.

Kathleen R. McKeown, Karen Kukich, and James Shaw. 1994. Practical issues in automatic documentation generation. In *Proc. of ANLP*, pages 7–14.

Kathleen R. McKeown. 1979. Paraphrasing using given and new information in a question-answer system. In *Proc. of the ACL*, pages 67–72.

Marie Meteer and Varda Shaked. 1988. Strategies for effective paraphrasing. In *Proc. of COLING*, pages 431–436.

Franz Josef Och, Christoph Tillman, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proc. of EMNLP*, pages 20–28.

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*.

Kishore A. Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proc. of the ACL*, pages 311–318.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of English words. In *Proc. of the ACL*, pages 183–190.

Jacques Robin. 1994. *Revision-Based Generation of Natural Language Summaries Providing Historical Background: Corpus-Based Analysis, Design, Implementation, and Evaluation*. Ph.D. thesis, Columbia University.

Yusuke Shinyama, Satoshi Sekine, Kiyoshi Sudo, and Ralph Grishman. 2002. Automatic paraphrase acquisition from news articles. In *Proc. of HLT*, pages 40–46.

Frank Smadja and Kathleen McKeown. 1991. Using collocations for language generation. *Computational Intelligence*, 7(4). Special issue on natural language generation.

Stephan Vogel and Hermann Ney. 2000. Construction of a hierarchical translation memory. In *Proc. of COLING*, pages 1131–1135.

Ye-Yi Wang. 1998. *Grammar Inference and Statistical Machine Translation*. Ph.D. thesis, CMU.

Appendix

In this appendix, we describe how we insert slots into lattices to form slotted lattices.

Recall that the backbone nodes in our lattices represent words appearing in many of the sentences from which the lattice was built. As mentioned above, the intuition is that areas of high variability between backbone nodes may correspond to arguments, or slots. But the key thing to note is that there are actually two different phenomena giving rise to multiple parallel paths: *argument variability*, described above, and *synonym variability*. For example, Figure 6(b) contains parallel paths corresponding to the synonyms "injured" and "wounded". Note that we want to *remove* argument variability so that we can generate paraphrases of sentences with arbitrary arguments; but we want to *preserve* synonym variability in order to generate a variety of sentence rewritings.

To distinguish these two situations, we analyze the *split level* of backbone nodes that begin regions with multiple paths. The basic intuition is that there is probably more variability associated with arguments than with synonymy: for example, as datasets increase, the number of locations mentioned rises faster than the number of synonyms appearing. We make use of a *synonymy threshold* s (set by held-out parameter-tuning to 30), as follows.

- If no more than $s\%$ of all the edges out of a backbone node lead to the same next node, we have high enough variability to warrant inserting a slot node.
- Otherwise, we incorporate reliable synonyms[12] into the backbone structure by preserving all nodes that are reached by at least $s\%$ of the sentences passing through the two neighboring backbone nodes.

Furthermore, all backbone nodes labelled with our special generic tokens are also replaced with slot nodes, since they, too, probably represent arguments (we condense adjacent slots into one). Nodes with in-degree lower than the synonymy threshold are removed under the assumption that they probably represent idiosyncrasies of individual sentences. See Figure 6 for examples.

Figure 3 shows an example of a lattice and the slotted lattice derived via the process just described.

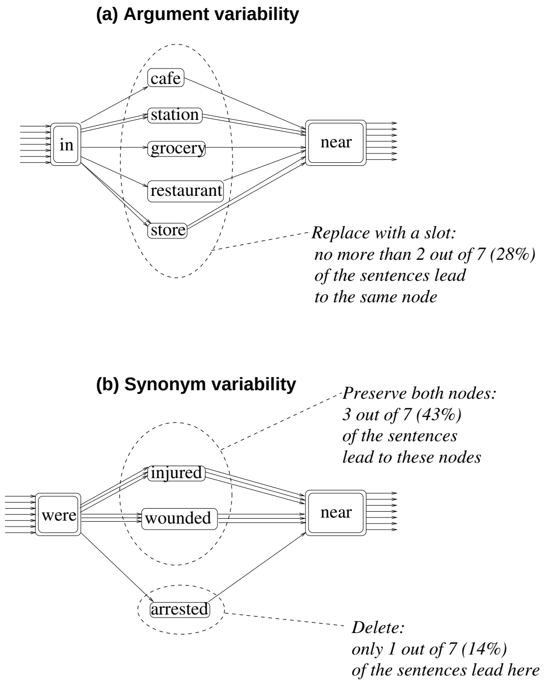

Figure 6: Simple seven-sentence examples of two types of variability. The double-boxed nodes are backbone nodes; edges show consecutive words in some sentence. The synonymy threshold (set to 30% in this example) determines the type of variability.

[12] While our original implementation, evaluated in Section 4, identified only single-word synonyms, phrase-level synonyms can similarly be acquired by considering chains of nodes connecting backbone nodes.