Statistical Machine Translation Using Labeled Semantic Dependency Graphs

Anthony Aue, Arul Menezes, Bob Moore, Chris Quirk, Eric Ringger
Microsoft Research
1 Microsoft Way
Redmond, WA, USA
{anthaue, arulm, bobmoore, chrisq, ringger}@microsoft.com

Abstract

We present a series of models for doing statistical machine translation based on labeled semantic dependency graphs. We describe how these models were employed to augment an existing example-based MT system, and present results showing that doing so led to a significant improvement in translation quality as measured by the BLEU metric.

1. Introduction

Much research of late has been devoted to the invention and implementation of statistical machine translation (hereafter: SMT) systems of the kind originally described in (Brown et al., 1993). What these systems have in common is that they try to predict the most likely target language string given an input string in the source language using statistical methods.

While traditional SMT systems do not explicitly model structural linguistic properties of the languages they are translating, recent trends in both SMT and speech recognition research suggest that it may be worthwhile to pay more attention to these properties when creating language models. Language models and SMT systems that take syntactic information into account are becoming more numerous. A representative sample includes the work of Chelba (1997), Charniak (2001), Knight and Yamada (2001), Charniak, Knight and Yamada (2003), and Eisner (2003). The intuition is that syntax-based models ought to be able to capture long-distance dependencies that surface-string models are unable to capture because events that depend on each other are often closer together in the syntax tree than they are in the surface string, in the sense that the distance to a sibling or common parent may be significantly shorter than distance between the same events in the surface string.

The system described in this paper is an attempt to take one step further in the same direction: if syntactic models are good because they move events that depend on each other closer together than they are in the surface string, then semantic models ought to be even better. A semantic representation of a sentence ought to bring related events into nearer proximity by explicitly representing semantic dependencies still latent in the syntactic representation.

In this paper, we show how our existing example-based MT system using labeled semantic dependency graph translation mappings benefited from the application of SMT techniques while still preserving the benefits provided by rich hierarchical linguistic information.

2. Description of the baseline system and the statistical framework

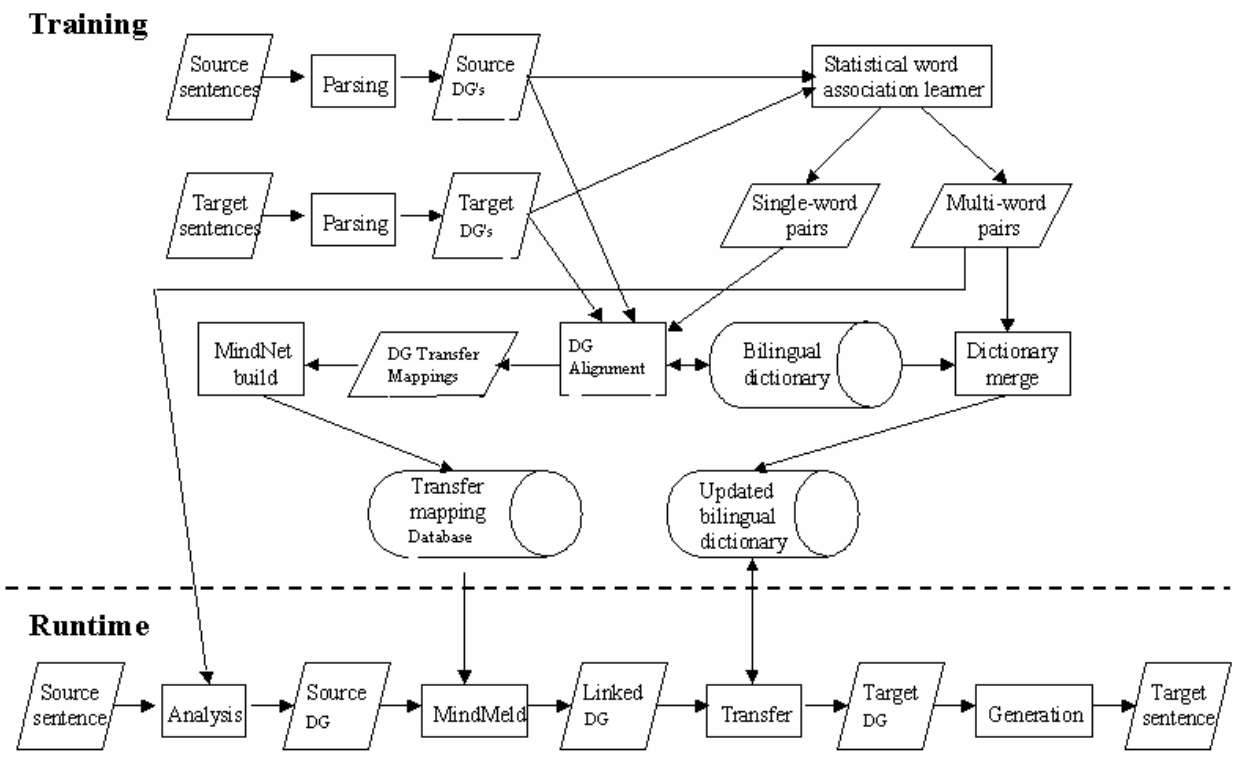
2.1. The baseline system

For an overview of the example-based system to which we've applied these statistical techniques, see (Richardson et al, 2001). In a nutshell, the system parses aligned source and target language training sentences using a bottom-up, multi-path chart parsing algorithm, aligns the resulting

dependency graphs and creates a set of mappings from source language dependency graph fragments to target language dependency graph fragments (hereafter referred to as DG mappings). The nodes in a given dependency graph represent the lemmas of all concept words in the corresponding sentence. A pair of nodes may be connected by a directed edge labeled by a semantic or deep-syntactic dependency relationship. Associated with each node in the graph are zero or more binary features which specify additional syntactic or semantic information about the lemma of the node to which they are attached. Corresponding source and target dependency graphs are aligned to one another with the aid of a learned translation dictionary.

At runtime, the input string is parsed and converted to a dependency graph which is then matched against the database of learned dependency graph fragments. This process yields the set of all (possibly overlapping) mappings whose source dependency graph chunk matches a portion of the input dependency graph.

Figure 1: System diagram



The baseline system then employs a greedy heuristic decoder to decide which set of transfer mappings to use. The mappings are sorted by size, then by the number of binary features matched, and finally by frequency. The first compatible set of mappings found that covers the input dependency graph is selected. Any gaps (nodes in the input graph not covered by the learned mappings) are filled in using a learned translation dictionary. The target sides of the chosen mappings are then combined into a complete target language dependency graph and passed to a generation module which generates the target language surface string.

2.2. The statistical framework

The statistical framework described in this paper is designed as a replacement for the heuristic decoding mentioned above. It attempts to answer the question: "Which of all of the possible combinations of transfer mappings whose source component is covered by a portion of the input dependency graph will yield the best target language dependency graph?" The rest of the system components (i.e. the alignment and partitioning of the training data and the analysis and generation components) did not have to be modified for the new system.

The traditional noisy-channel SMT model attempts to find the highest-probability translation for a sentence

$$T = \arg\max(P(T \mid S)), \qquad (1)$$

where S is the source language sentence and T the target language sentence. By Bayes' rule,

$$T = \arg\max(P(T \mid S)) = \arg\max(P(S \mid T)P(T)). \quad (2)$$

A target language model trained on monolingual target language data is used to compute an estimate of *P(T)*, and channel models of varying complexity are built to compute and estimate *P(S|T)*.

In our system, individual candidate mappings and combinations of mappings are scored using a linearly interpolated combination of scores from several heterogeneous information sources. This "kitchen sink" approach to SMT is superficially similar to the system described in (Och and Ney, 2002), except that it works with dependency graph mappings instead of with surface strings. Formally, then, we are looking for the set of transfer mappings T_{max} out of all sets of transfer mappings T such that:

$$T_{\max} = \arg\max\{\sum_{\mu=1}^{M} \lambda_\mu Score_m(T)\}, \qquad (3)$$

where $\mu \in M$ are the individual models, each of which assigns a score to a set of transfer mappings.

The sources consulted for our system include a probabilistic channel model, target language model, and simple fertility model. Additional information sources whose scores are interpolated with the traditional SMT models include mapping size and number of binary features matched. We train weights for the models using Powell's algorithm to maximize the BLEU score on the output of the system (Papineni et al., 2001).

While the general idea behind these approaches is not new, the innovation in both cases is to apply these techniques at the labeled dependency graph level rather than at the string level. We know of no other system to date that has used statistical techniques to maximize the probability of labeled dependency-graph-to-dependency-graph mappings for machine translation.

3. Models

3.1. Target language model

Most SMT systems use surface string *n*-gram models for their target language model. This has a number of advantages, particularly in a string-to-string translation system. Perhaps most

importantly, since *n*-gram models have been used in speech recognition for quite some time, there is considerable literature devoted to the subject of smoothing and compressing them (c.f. Stolcke, 1998; Goodman, 2000). The models are simple to train and to use, and it is relatively easy to optimize the size of the models to find a good tradeoff between minimizing space requirements and getting the precision necessary for the given task.

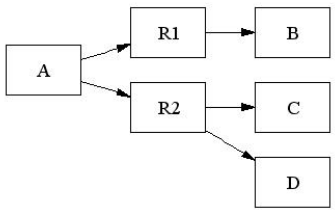
However, surface-string *n*-gram models have some limitations as well. Perhaps the greatest limitation is their primary independence assumption, that word w_i depends only on words $w_{i-1}, w_{i-2} \ldots w_{n-1}$. While it's sometimes true that a word can be accurately predicted from one or two of its immediate predecessors, a number of linguistic constructs place highly predictive words sufficiently far from the words they predict that they are excluded from the scope of the *n*-gram. (Think, for example, of extraposition or of extended adjective constructions).

We attempt to circumvent this particular shortcoming by building an *n*-gram model based on dependency graphs instead of on surface strings. Our target language model is a 5-gram model based on a vertical Markovization[1] of dependency graphs that have been converted to trees, in which relations are represented as first-class events (nodes) along with the lemmas they relate. We make the Markov independence assumption that the probability of a given tree is the product of the probability of each of the nodes given its (*n-1*) previous ancestors. Thus we compute the probability of the target dependency graph τ by the following formula:

$$P_{\mu_T}(\tau) = \prod_{i}^{|\tau|} P(c_i \mid c_{i-1} \ldots c_{i-(n-1)}, \mu_T), \qquad (4)$$

where C are all of the nodes in τ, c_{i-1} denotes the parent of c_i, and *n* is the order of the model. The model is pruned by removing infrequently-occurring *n*-grams and smoothed using interpolated absolute discounting.

Consider the following dependency graph:



The probability of the graph according to a trigram DG model would be:

```
P(A | ROOT) * P(R1 | ROOT A) * P (R2 | ROOT A) * P (B | A
R1) * P(C | A R2) * P(D | A R2) * P (LEAF | R1 B) * P(LEAF |
R2 C) * P(LEAF | R2 D)
```

[1] See Klein and Manning, 2003 for further discussion of vertical Markovization. It would have been nice to include horizontal (i.e. sibling) information in our model as well, but the top-down nature of our decoder (and the performance-motivated incorporation of the Markov assumption therein) made this impractical.

Treating semantic relationships (in this example: R1 and R2) as first-class entities simplifies the creation of the model a great deal, since it means we don't have to train and store two separate models for lemmas and semantic relationships.

To see why dependency graph-based models might model certain phenomena better than string-based *n*-gram models or syntax trees, consider the following sentences:

1. *John hit the ball.*
2. *The balls were hit by Lucy.*

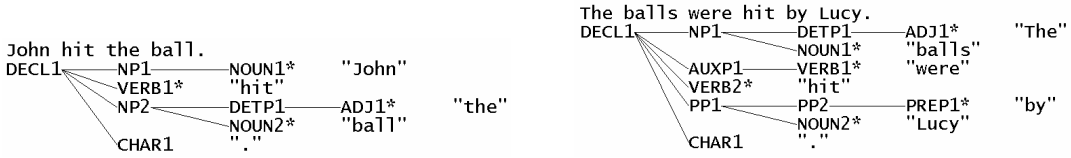
A surface-string-based 3-gram model would count the following n-grams, each n-gram getting count 1:

Table 1: Surface string N-gram Counts

<Pr> <Pr> John 1	<Pr> John hit 1	John hit the 1	hit the ball 1	the ball <POST> 1
<Pr><Pr> The 1	<Pr> The balls 1	the balls were 1	balls were hit 1	were hit by 1
hit by Lucy 1	By Lucy <Po> 1			

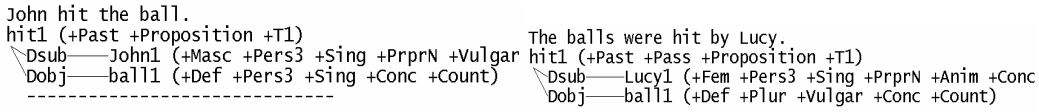
Note that each of these trigrams occurs only once, even though the event (the hitting of a ball) is the same in both cases.

If we look at syntax trees, we get a slightly different picture:



One could, as in (Chelba and Jelinek, 1998) build up a statistical grammar from a corpus of parse trees such as these. This grammar could then be used in a syntax-tree-based SMT system in order to assign a probability to target language syntax tree candidates. Note, however, that here, too, the hitting of the ball would be split into two separate buckets (one set of rules for the active voice construction and another for the passive voice), and so the system would fail to learn a useful generalization.

The dependency graphs our system produces for the two sentences looks like this:



The dependency graph target language model would generate the following trigram counts for this sentence pair:

Table 2: Dependency Graph N-gram Counts

<PR> <PR> hit 2	<PR> hit Dsub 2	<PR> hit Dobj 2	hit Dsub John 1
hit Dsub Lucy 1	hit Dobj ball 2	Dsub John <PL> 1	Dsub Lucy <PL> 1

The DG n-gram model has captured several important generalizations here, for instance that "hit" is likely to be transitive, and that "ball" is a likely direct object for the verb "hit."

While the example above is trivial, we believe that phenomena similar to the one described above contribute to the quality of the target language model.

3.2. Fertility model

One problem the target language model suffers from is the fact that it always prefers shorter graphs to longer ones. This can be a problem when we consider transfer mappings that delete or insert nodes. Ideally, dependency graphs for translated sentence pairs would have similar structures and sizes. Unfortunately, we often have to deal with free translations (and, even more unfortunately, we have imperfect parsers), and so this is not always the case.

To combat this problem, we apply a simple fertility model. The parameters for the model are calculated directly by walking through the training data and counting, for each node in each the source graph, how often there is a corresponding node in the target graph. In order to avoid sparse data problems, counts for the lemmas are grouped together by part of speech, while counts for the semantic relationships (of which there are an approximately equal number as there are parts of speech) are all counted separately. These frequencies are then used to compute the maximum likelihood estimate of a deletion occurring. For each part of speech or semantic relationship constituent *c*, we insert an entry in our fertility table *F* such that

$$F[c] = \frac{count(c \in m_s, c \in m_t)}{count(c \in m_s)}, \quad (5)$$

where *count(x)* refers to the number of times x applies in all mappings M in the training data, m_s is the source side of the individual transfer mapping *m*, and m_t the target. The probability of a mapping according to the fertility model can be computed as:

$$P_{\mu_F}(m) = \prod_{c_s \in m_s} f(c), \quad (6)$$

where

$$f(c) = \begin{cases} F[x] & \text{if } \exists c_t \in m_t : c_t \text{ corresponds to } c_s \\ 1 - F[x] & \text{otherwise} \end{cases} \quad (7)$$

In theory, we should attempt to model insertions as well, but experiments thus far have not shown this to be useful.

3.3. Channel model

The traditional Bayes-inspired noisy channel SMT model calls for a channel model that predicts *P(S | T)*.

We estimate *P(S|T)* of mapping m as

$$\frac{count(m_S, m_T)}{count(m_T)}, \quad (8)$$

where *count(x)* is the number of times *x* was encountered during training, m_s is the source side of the transfer mapping, and m_t is the target side of the transfer mapping.

Since we allow overlapping mappings, the probability given by the model has to be normalized so that the probability mass contributed by the mappings covering each input node is identical. This way we avoid penalizing mappings for which there is more overlap.

The normalized probability P_n for a mapping *m* is computed by the following formula:

$$P_n(m \mid \mu_C) = P_u(m \mid \mu_C)^{\frac{new}{total}}, \qquad (9)$$

where "*new*" is the number of constituents in the input graph that haven't already been covered by another mapping chosen in an earlier iteration of the decoding algorithm, "*total*" is the total number of constituents in m_s, and P_u is the unnormalized probability.[2]

Since we are not guaranteed to have a compatible set of mappings covering any given input, we also add default nodes of size 1 for each input lemma and relation not already covered by a single-node mapping from the training data. For relations, the default node is simply a copy of the input relation. For lemmas, we create a new node by translating the input lemma using the most likely single-word translation according to the LLR-based word association model described in (Moore, 2001). Then we compute word translation probabilities over the corpus using IBM Model 1 and assign the resulting probability to the chosen mapping.

Since the IBM Model 1 probabilities are obtained by a completely different process than the dependency graph mappings, we have found it advantageous to train up a weight, λ_L, by which the Model 1 probabilities are multiplied before they are combined with the other channel model probabilities. In addition there is another free parameter, λ_A, which is the default value assigned to the single-node semantic relationship mappings.

Presently, the handling of default mappings is somewhat insufficient for both lemmas and semantic relationships, and in the future we hope to come up with better models in both scenarios. For the semantic relationships, we could fairly easily obtain a list of permissible single-node mappings by direct examination of the training data. In both cases, we should probably add all possible translation mappings instead of the greedily chosen "best" mapping.

3.4. Mapping size

This model assigns a score that is simply the number of input nodes covered by the mapping *m*, i.e.:

$$Score_{\mu_S}(m) = \mid m \mid \qquad (10)$$

What this does, in effect, is give precedence to larger mappings on the intuition that mappings with more context information are likely to be better. We believe that this assumption accounts for a great deal of the quality of the heuristic match selection algorithm used by the baseline system.

3.5. Binary features

The number of binary features that match between the input graph, *i,* and the source side of the transfer mapping, m_S,:

[2] This is not the most principled normalization strategy, but it was the best one we were able to find that was feasible given the limitations imposed by our decoding algorithm.

$$Score_{\mu_R}(m) = \sum_{c \in m_S} \sum_{f \in c} \delta(c, f, i), \qquad (11)$$

where $f \varepsilon c$ is the set of positive-valued features expressed on constituent c, and

$$\delta(c, f, i) = \begin{cases} 1 & f(c) \,\&\, f(i) \\ 0 & \text{otherwise} \end{cases} \qquad (12)$$

4. Decoding

The decoding process is accomplished using an optimal top-town dynamic-programming algorithm that takes advantage of the Markov assumption in the target language model and the context-independence of the other models. Despite the fact that it performs an optimal search whose worst-case time performance is exponential in the number of candidate mappings, in practice performance is acceptable in most cases. The key insight that makes the algorithm perform tolerably well while still guaranteeing that it returns the highest-probability set of mappings according to the model is that the target language model only cares about the previous n-1 ancestors in the tree when scoring any given n-gram, so it can reuse results from previous trips down the tree that resulted in the same context with different mappings. While the number of mappings covering a given input node might be quite high, a large number of the mapping combinations end up producing similar target dependency graph chunks, so in practice the hit rate in the memoization table is quite high.

5. Results

5.1. Presentation of results

As a baseline for our evaluation we used the heuristic system described in section 2.1. A Spanish-to-English system was trained on 460,000 aligned sentence pairs (about 5.7M words). Weights for the statistical models were trained using Powell's algorithm to maximize the BLEU score on 2000 sentences of held-out training data. The test data consisted of 10000 sentences. As Table 3 shows, the trained statistical framework outperformed the baseline system by a bleu score of .02. This number is statistically significant at 95% confidence.

The "untrained" score is the result of running the statistical system before training, using equal weights of 0.1 for all models. The "trained" score indicates the results using the weights found by running Powell's algorithm on the held-out training data.

In order to see how much each of the individual models contributed to the final results, we ran the series of ablation experiments summarized in Table 4. Here, "S" refers to the full statistical system, and "S-$_X$" refers to system S with the designated model removed. "Tgt only" is a system with all models removed except for the target language model.

The "Untrained" row represents a flat set of weights (all weights are set to 0.1). The "Trained" row represents the weights computed by training using all the models, but simply turning off the model in question.

5.2. Discussion of results

It should be noted that the baseline system is quite good. Because of the greediness of the decoding algorithm it's fast (it translates approximately 180 sentences/min on a modern desktop

machine), and the quality of the output is quite good (Richardson et al, 2001). The system is currently in daily use in a production environment. The "mapping size" heuristic has proved to be an excellent predictor of high-quality mappings. This makes sense intuitively, because larger mappings provide more context than smaller mappings.

However, the same greedy algorithm that allows the system to perform as fast as it does also prevents it from being able to take advantage of statistical models, which make much finer distinctions between mappings that the greedy algorithm would consider of the same quality.

The ablation experiments show that the target model clearly has the greatest impact. Removing the rest of the individual models didn't affect the BLEU score very much (or at all, in the case of the binary features). However, the score for the system with only the target model shows that their cumulative effect is indeed significant even if their individual contributions are small.

Table 3: Results of Statistical Systems vs. Baseline.[1]

System	BLEU Score
Baseline	0.4210
Statistical framework (untrained)	0.4322
Statistical framework (trained)	0.4365

Table 4: Results of removing each individual model from trained and untrained systems

	Untrained	Trained
S	0.4322	0.4365
S-size	0.4321	0.4359
S-bin. Feat.	0.4322	0.4365
S – channel	0.4298	0.4347
S – tgt.	0.3801	0.3657
Tgt only	0.4218	0.4218

6. Conclusion

We've demonstrated the feasibility of extending an example-based, data-driven machine translation system with statistical techniques and a new decoding algorithm, and that by doing so we can gain a significant improvement in translation quality over the baseline heuristic system. The performance of our statistical system, though significantly slower than that of the baseline system, is still quite good. Despite the fact that it performs an optimal search through the set of compatible transfer mappings, it still translates approximately 35 sentences/min. Thus the system is able to benefit from some of the quality improvements brought about by a statistically-based translation system without having to be rewritten from the ground up and without suffering too much performance loss.

As far as we know, our system is unique in that it is the only system that combines techniques from traditional string-based SMT with a labeled dependency-graph-to-dependency-graph translation model. We look forward to improving the system and extending it to more language pairs in the near future.

References

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics* 19(2).

E. Charniak. 2001. Immediate Head Parsing for Language Models. *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*.

E. Charniak, K. Knight, and K. Yamada. 2003. Syntax-based Language Models for Machine Translation. *Proceedings of MT Summit IX 2003.*

C. Chelba and F. Jelinek. 1998. Exploiting Syntactic Structure for Language Modeling. *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*.

Eisner, J. 2003. Learning non-isomorphic tree mappings for machine translation. *Proceedings of the 41st Meeting of the Association for Computational Linguistics (companion volume).*

Goodman, J. 2000. A Bit of Progress in Language Modeling. Technical report, Microsoft Research.

Klein, D. and Manning, C. 2003. Accurate Unlexicalized Parsing. ACL 2003:423-430.

Knight, K. 1999. Decoding Complexity in Word-Replacement Translation Models. *Computational Linguistics* 25(4).

Moore, R. 2001. Towards a Simple and Accurate Statistical Approach to Learning Translation Relationships among Words. Proceedings, Workshop on Data-driven Machine Translation, 39th Annual Meeting and 10th Conference of the European Chapter, Association for Computational Linguistics: 79-86.

Och, F. and Ney, H. 2002. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Papineni, K.A., Roukos, S., Ward, T., and Zhu, W.J. 2001. Bleu: a method for automatic evaluation of machine translation. *Technical Report RC22176 (W0109-022)*. IBM Research Division, Thomas J. Watson Research Center.

Powell, M. J. D. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivates. *Computer Journal* 7.

Richardson, S., Dolan, W., Menezes, A. and Pinkham, J. 2001. Achieving commercial-quality translation with example-based methods. *Proceedings of MT Summit VIII*: 293-298.

Stolcke, A. 1998. Entropy-based pruning of backoff language models. *Proc. DARPA Broadcast News Transcription and Understanding Workshop*.

Yamada, K., and Knight, K. 2001. A Syntax-Based Statistical Translation Model. *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*.