Prenominal Modifier Ordering via Multiple Sequence Alignment

Aaron DunlopMargaret MitchellBrian RoarkOregon Health & Science UniversityUniversity of AberdeenOregon Health & Science UniversityPortland, ORAberdeen, Scotland, U.K.Portland, ORdunlopa@cslu.ogi.edum.mitchell@abdn.ac.ukroark@cslu.ogi.edu

Abstract

Producing a fluent ordering for a set of prenominal modifiers in a noun phrase (NP) is a problematic task for natural language generation and machine translation systems. We present a novel approach to this issue, adapting multiple sequence alignment techniques used in computational biology to the alignment of modifiers. We describe two training techniques to create such alignments based on raw text, and demonstrate ordering accuracies superior to earlier reported approaches.

1 Introduction

Natural language generation and machine translation systems must produce text which not only conforms to a reasonable grammatical model, but which also sounds smooth and natural to a human consumer. Ordering prenominal modifiers in noun phrases is particularly difficult in these applications, as the rules underlying these orderings are subtle and not well understood. For example, the phrase "big red ball" seems natural, while "red big ball" seems more marked, suitable only in specific contexts. There is some consensus that the order of prenominal modifiers in noun phrases is governed in part by semantic constraints, but there is no agreement on the exact constraints necessary to specify consistent orderings for any given set of modifiers. General principles of modifier ordering based on semantic constraints also fall short on larger domains, where it is not always clear how to map prenominal modifiers to proposed semantic groups.

With the recent advantages of large corpora and powerful computational resources, work on automatically ordering prenominal modifiers has moved away from approaches based on general principles, and towards learning ordering preferences empirically from existing corpora. Such approaches have several advantages: (1) The predicted orderings are based on prior evidence from 'real-world' texts, ensuring that they are therefore reasonably natural. (2) Many (if not all) prenominal modifiers can be ordered. (3) Expanding the training data with more and larger corpora often improves the system without requiring significant manual labor.

In this paper, we introduce a novel approach to prenominal modifier ordering adapted from multiple sequence alignment (MSA) techniques used in computational biology. MSA is generally applied to DNA, RNA, and protein sequences, aligning three or more biological sequences in order to determine, for example, common ancestry (Durbin et al., 1999; Gusfield, 1997; Carrillo and Lipman, 1988). MSA techniques have not been widely applied in NLP, but have produced some promising results for building a generation mapping dictionary (Barzilay and Lee, 2002), paraphrasing (Barzilay and Lee, 2003), and phone recognition (White et al., 2006).

We believe that multiple sequence alignment is well-suited for aligning linguistic sequences, and that these alignments can be used to predict prenominal modifier ordering for any given set of modifiers. Our technique utilizes simple features within the raw text, and does not require any semantic information. We achieve good performance using this approach, with results competitive with earlier work (Shaw and Hatzivassiloglou, 1999; Malouf, 2000; Mitchell, 2009) and higher recall and F-measure than that reported in Mitchell (2009) when tested on the same corpus.

Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL, pages 600–608, Los Angeles, California, June 2010. ©2010 Association for Computational Linguistics

2 Related work

In one of the first attempts at automatically ordering prenominal modifiers, Shaw and Hatzivassiloglou (1999) present three empirical methods to order a variety of prenominal modifier types. Their approach provides ordering decisions for adjectives, gerunds (such as "running" in "running man"), and past participles (such as "heated" in "heated debate"), as well as for modifying nouns (such as "baseball" in "baseball field"). A morphology module transforms plural nouns and comparative/superlative forms into their base forms, increasing the frequency counts for each modifier. We will briefly recap their three methods, which are categorized as the *direct evidence* method, the *transitivity* method, and the *clustering* method.

Given prenominal modifiers a and b in a training corpus, the *direct evidence* method compares frequency counts of the ordered sequences $\langle a,b \rangle$ and $\langle b,a \rangle$. This approach works well, but is limited by data sparsity; groups of two or more modifiers before a noun are relatively infrequent in traditional corpora, and finding the same pair of modifiers together more than once is particularly rare.

To overcome this issue. Shaw and Hatzivassiloglou's transitivity and clustering methods make inferences about unseen orderings among prenominal modifiers. In the transitivity method, given three modifiers a, b, c, where a precedes b and b precedes c, the model concludes that a precedes c. The clustering method calculates a similarity score between modifiers based on where the modifiers occur in relation to the other modifiers in the corpus. Those modifiers that are most similar are clustered together, and ordering decisions can be made between modifiers in separate clusters. All three approaches are designed to order pairs of modifiers; it is unclear how to extend these approaches to order groups larger than a pair.

Shaw and Hatzivassiloglou find that NPs with only adjectives as modifiers (including gerunds and past participles) are considerably easier to order than those which contain both adjectives and nouns. They also find large differences in accuracy across domains; their systems achieve much lower overall accuracy on financial text (the Wall Street Journal (WSJ) corpus (Marcus et al., 1999)) than on medical discharge summaries.

Looking at all modifier pairs, the authors achieve their highest prediction accuracy of 90.7% using the transitivity technique on a medical corpus. We do not have access to this corpus, but we do have access to the WSJ corpus, which provides a way to compare our methods. On this corpus, their model produces predictions for 62.5% of all modifier pairs and achieves 83.6% accuracy when it is able to make a prediction. Random guessing on the remainder yields an overall accuracy of 71.0%.

Malouf (2000) also examines the problem of prenominal modifier ordering. He too proposes several statistical techniques, achieving results ranging from 78.3% to 91.9% accuracy. He achieves his best results by combining memorybased learning and positional probability to modifiers from the first 100 million tokens of the BNC. However, this evaluation is limited to the ordering of prenominal adjectives, which is a considerably simpler task than ordering all types of prenominal modifiers. Malouf's approaches are also limited to ordering pairs of modifiers.

Mitchell (2009) proposes another approach, grouping modifiers into classes and ordering based on those classes. A modifier's class is assigned based on its placement before a noun, relative to the other modifiers it appears with. Classes are composed of those modifiers that tend to be placed closer to the head noun, those modifiers that tend to be placed farther from the head noun, etc., with each class corresponding to a general positional preference. Unlike earlier work, these classes allow more than one ordering to be proposed for some pairs of modifiers.

Combining corpora of various genres, Mitchell's system achieves a *token precision* of 89.6% (see Section 4 for discussion and comparison of various evaluation metrics). However, the model only makes predictions for 74.1% of all modifier pairs in the test data, so recall is quite low (see Tables 4 and 6).

Overall, previous work in noun-phrase order-

ing has produced impressive accuracies in some domains, but currently available systems tend to adapt poorly to unseen modifiers and do not generalize well to unseen domains.

3 Methods

3.1 Multiple Sequence Alignment

Multiple sequence alignment algorithms align sequences of discrete tokens into a series of columns. They attempt to align identical or easily-substitutable tokens within a column, inserting gaps when such gaps will result in a better alignment (more homogeneous token assignments within each column). For example, consider the simple alignment shown in Table 1. The two sequences 'GAACTGAT' and 'AAGT-GTAT' are aligned to maximize the number of identical items that appear in the same column, substituting tokens (column 3), and inserting gaps (columns 1 and 6)¹.

A full MSA is generally constructed by iteratively aligning each new sequence with an identical or similar sequence already in the MSA (socalled "progressive alignment"). The costs of token substitution are often taken from a handtuned substitution matrix. A cost may also be associated with inserting a gap into the existing MSA (a "gap penalty"). Once the full MSA has been constructed, a Position Specific Score Matrix (PSSM) can be induced, in which each token (including a special gap token) is assigned a separate alignment cost for each column. An unseen sequence can then be aligned with the full MSA by Viterbi search.

Predicting sequence ordering within a noun phrase is a natural application for MSA techniques, and it seems reasonable to propose that aligning an unseen set of modifiers with such an MSA model will yield acceptable orderings. Table 2 illustrates how MSA may be applied to modifiers before a noun. Given an NP preceded by modifiers *hungry*, *big*, and *Grizzly*, alignment of the modifiers with NPs seen in the training corpus determines the prenominal ordering *big hungry Grizzly*. We then align every permuta-

G	А	С	Т	G	-	А	Т
-	А	G	Т	G	Т	А	Т
1	2	3	4	5	6	7	8

Table 1: Alignment of the two DNA sequences 'GAACTGAT' and 'AAGTGTAT'.

small	clumsy	black	bear
big	-	black	cow
two-story	-	brown	house
big	clumsy	-	bull
small	fuzzy	brown	duck
large	-	green	house
big	hungry	Grizzly	bear

Table 2: Example noun-phrase alignment.

tion of the NP and choose the best-scoring alignment.

The vocabulary for a linguistic alignment is large enough to render a hand-tuned substitution matrix impractical, so we instead construct a cost function based on features of the token under consideration and those of the other tokens already aligned in a column.

We know of no prior work on methods for training such an alignment. We present and compare two training methods, each of which produces competitive ordering accuracies. Both training methods share the feature-set described in Table 3. In each case, we train an MSA by aligning each instance in the training data.

3.2 Maximum Likelihood Training

In our alignment approach, the features listed in Table 3 are grouped into several classes. All observed words are a class, all observed stems are a class (Porter, 1980), and so on. We treat each indicator feature as a separate class, and make the assumption that classes are independent of one another. This assumption is clearly false, but serves as a reasonable first approximation, similar to the independence assumption in Naïve Bayesian analysis. After aligning each instance, we estimate the probability of a feature appearing in a column as the simple maximum likelihood estimate given the observed occurrences

¹See Durbin et al. (1999) for details on standard alignment techniques.

Identity Features				
Word	Token			
Stem	Word stem, derived by the Porter Stemmer			
Length	'Binned' length indicators: 1, 2, 3, 4, 5-6, 7-8, 9-12, 13-18, >18 characters			
Indicator Features				
Capitalized	Token begins with a capital			
All-caps	Entire token is capitalized			
Hyphenated	Token contains a hyphen			
Numeric	Entire token is numeric (e.g. 234)			
Initial Numeric	Token begins with a numeral (e.g. 123, 2-sided)			
Endings	Token ends with -al, -ble, -ed, -er, -est, -ic, -ing, -ive, -ly			

Table 3: Description of the feature-set.

within its class.² This produces a new PSSM with which to align the next instance.

Our problem differs from alignment of biological sequences in that we have little prior knowledge of the similarity between sequences. 'Similarity' can be defined in many ways; for biological sequences, a simple Levenshtein distance is effective, using a matrix of substitution costs or simple token identity (equivalent to a matrix with cost 0 on the diagonal and 1 everywhere else). These matrices are constructed and tuned by domain experts, and are used both in choosing alignment order (i.e., which sequence to align next) and during the actual alignment. When aligning biological sequences, it is customary to first calculate the pairwise distance between each two sequences and then introduce new sequences into the MSA in order of similarity. In this way, identical sequences may be aligned first, followed by less similar sequences (Durbin et al., 1999).

However, we have no principled method of determining the 'similarity' of two words in an NP. We have no a priori notion of what the cost of substituting 'two-story' for 'red' should be. Lacking this prior knowledge, we have no optimal alignment order and we must in effect learn the substitution costs as we construct the MSA. Therefore, we choose to add instances in the order they occur in the corpus, and to iterate over the entire MSA, re-introducing each sequence. This allows a word to 'move' from its original column to a column which became more likely as more sequences were aligned. Each iteration is similar to a step in the EM algorithm: create a model (build up an MSA and PSSM), apply the model to the data (re-align all sequences), and repeat. Randomly permuting the training corpus did not change our results significantly, so we believe our results are not greatly dependent on the initial sequence order.

Instead of assigning substitution costs, we compute the cost of aligning a word into a particular column, as follows:

$$\mathbb{C}$$
 = The set of *i* feature classes, $C_i \in \mathbb{C}$

$$j =$$
Features $1 \dots |C_i|$ from class C_i

$$cnt(i, j, k) =$$
 The count of instances of
feature j from class
 i in column k

 $\lambda_i = \text{Laplace smoothing count}$ for feature class C_i

A = The number of aligned instances

$$f(w, i, j) = \begin{cases} 1 & \text{if word } w \text{ has feature } j \text{ from} \\ & C_i, \\ 0 & \text{otherwise} \end{cases}$$

These help define feature positional probabilities for column k:

$$p(i,j,k) = \frac{cnt(i,j,k) + \lambda_i}{A + \lambda_i \cdot |C_i|} \tag{1}$$

 $^{^2 \}rm We$ treat two special symbols for gaps and unknown words as members of the word class.

That is, the probability of feature j from class i occurring in column k is a simple maximumlikelihood estimate — count the number of times we have already aligned that feature in the column and divide by the number of sequences aligned. We smooth that probability with simple Laplace smoothing.

We can now calculate the probability of aligning a word w into column k by multiplying the product of the probabilities of aligning each of the word's features. Taking the negative log to convert that probability into a cost function:

$$c(w,k) = -\sum_{i=1}^{|\mathbb{C}|} \sum_{j=1}^{|C_i|} \log \left(p(i,j,k) \cdot f(w,i,j) \right)$$
(2)

Finally, we define the cost of inserting a new column into the alignment to be equal to the number of columns in the existing alignment, thereby increasingly penalizing each inserted column until additional columns become prohibitively expensive.

 $i(j) = I \cdot$ Length of existing alignment (3)

The longest NPs aligned were 7 words, and most ML MSAs ended with 12-14 columns. We experimented with various column insertion costs and values for the smoothing λ and found no significant differences in overall performance.

3.3 Discriminative Training

We also trained a discriminative model, using the same feature-set. Discriminative training does not require division of the features into classes or the independence assumption discussed in Section 3.2. We again produced a cost vector for each column. We fixed the alignment length at 8 columns, allowing alignment of the longest instances in our test corpus.

Our training data consists of ordered sequences, but the model we are attempting to learn is a set of column probabilities. Since we have no gold-standard MSAs, we instead align the ordered NPs with the current model and treat the least cost alignment of the correct ordering as the reference for training. We trained this model using the averaged perceptron algorithm (Collins, 2002). A perceptron learns from classifier errors, i.e., when it misorders an NP. At each training instance, we align all possible permutations of the modifiers with the MSA. If the least cost alignment does not correspond to the correct ordering of the modifiers, we update the perceptron to penalize features occurring in that alignment and to reward features occurring in the least cost alignment corresponding to the correct ordering, using standard perceptron updates.

Examining every permutation of the NP involves a non-polynomial cost, but the sequences under consideration are quite short (less than 1% of the NPs in our corpus have more than 3 modifiers, and the longest has 6; see Table 7). So exhaustive search is practical for our problem; if we were to apply MSA to longer sequences, we would need to prune heavily.³

4 Evaluation

We trained and tested on the same corpus used by Mitchell (2009), including identical 10-fold cross-validation splits. The corpus consists of all NPs extracted from the Penn Treebank, the Brown corpus, and the Switchboard corpus (Marcus et al., 1999; Kucera and Francis, 1967; Godfrey et al., 1992). The corpus is heavily biased toward WSJ text (74%), with approximately 13% of the NPs from each of the other corpora.

We evaluated our system using several related but distinct metrics, and on both modifier pairs and full NPs.

We define:

- \mathbb{T} = The set of unique orderings found in the test corpus
- \mathbb{P} = The set of unique orderings predicted by the system

Type Precision $(|\mathbb{P} \cap \mathbb{T}|/|\mathbb{P}|)$ measures the probability that a predicted ordering is 'reasonable' (where 'reasonable' is defined as orderings which are found in the test corpus).

³The same issue arises when evaluating candidate orderings; see Section 4.

	Token Accuracy	Type Precision	Type Recall	Type F-measure
Mitchell	N/A	90.3 % (2.2)	67.2% (3.4)	77.1%
ML MSA	85.5% (1.0)	84.6%~(1.1)	84.7%~(1.1)	84.7%
Perceptron MSA	88.9 % (0.7)	88.2%~(0.8)	88.1% (0.8)	88.2 %

Table 4: Results on the combined WSJ, Switchboard, and Brown corpus; averages and standard deviations over a 10-fold cross validation. Winning scores are in bold.

Type Recall $(|\mathbb{P} \cap \mathbb{T}|/|\mathbb{T}|)$ measures the percentage of 'reasonable' orderings which the system recreates.

Note that these two metrics differ only in notation from those used by Mitchell (2009).

We also define a third metric, **Token Accu**racy, which measures accuracy on each individual ordering in the test corpus, rather than on unique orderings. This penalizes producing orderings which are legal, but uncommon. For example, if $\{a, b\}$ occurs eight times in the test corpus as $\langle a, b \rangle$ and two times as $\langle b, a \rangle$, we will be limited to a maximum accuracy of 80% (presuming our system correctly predicts the more common ordering). However, even though suggesting $\langle b, a \rangle$ is not strictly incorrect, we generally prefer to reward a system that produces more common orderings, an attribute not emphasized by type-based metrics. Our test corpus does not contain many ambiguous pairings, so our theoretical maximum token accuracy is 99.8%.

We define:

$$o_{1..N}$$
 = All modifier orderings in the test data

 $pred(o_i) =$ The predicted ordering for modifiers in o_i

$$a_{i} = \begin{cases} 1 & \text{if } pred(o_{i}) = o_{i}, \\ 0 & \text{otherwise} \end{cases}$$

Token Accuracy =
$$\sum_{i=0}^{N} \frac{a_{i}}{N}$$

4.1 Pairwise Ordering

Most earlier work has focused on ordering pairs of modifiers. The results in Table 4 are directly comparable to those found in Mitchell (2009). Mitchell's earlier approach does not generate a prediction when the system has insufficient evidence, and allows generation of multiple predictions given conflicting evidence. In theory, generating multiple predictions could improve recall, but in practice her system appears biased toward under-predicting, favoring precision. Our approach, in contrast, forces prediction of a single ordering for each test instance, occasionally costing some precision (in particular in cross-domain trials; see Table 5), but consistently balancing recall and precision.

Our measurement of Token Accuracy is comparable to the accuracy measure reported in Shaw and Hatzivassiloglou (1999) and Malouf (2000) (although we evaluate on a different corpus). Their approaches produce a single ordering for each test instance evaluated, so for each incorrectly ordered modifier pair, there is a corresponding modifier pair in the test data that was not predicted.

Shaw and Hatzivassiloglou found financial text particularly difficult to order, and reported that their performance dropped by 19% when they included nouns as well as adjectives. Malouf's system surpasses theirs, achieving an accuracy of 91.9%. However, his corpus was derived from the BNC—he did not attempt to order financial text— and he ordered only adjectives as modifiers. In contrast, our test corpus consists mainly of WSJ text, and we test on all forms of prenominal modifiers. We believe this to be a considerably more difficult task, so our peak performance of 88.9% would appear to be—at worst—quite competitive.

Table 5 presents an evaluation of crossdomain generalization, splitting the same corpus by genre—Brown, Switchboard, and WSJ. In each trial, we train on two genres and test on

	Training	Testing	Token	Type	Type	Type
	Corpora	Corpus	Accuracy	Precision	Recall	F-measure
	Brown+WSJ	Swbd	N/A	94.2 %	58.2%	72.0%
Mitchell	Swbd+WSJ	Brown	N/A	87.0 %	51.2%	64.5%
	Swbd+Brown	WSJ	N/A	82.4 %	27.2%	40.9%
	Brown+WSJ	Swbd	74.6%	74.7%	75.3%	75.0%
ML MSA	Swbd+WSJ	Brown	75.3%	74.7%	74.9%	74.8%
	Swbd+Brown	WSJ	70.2%	71.6%	71.8%	71.7%
	Brown+WSJ	Swbd	77.2%	78.2%	77.6%	77.9 %
Perceptron MSA	Swbd+WSJ	Brown	76.4 %	76.7%	76.4 %	76.5 %
	Swbd+Brown	WSJ	77.9 %	77.5%	77.3 %	77.4 %

Table 5: Cross-domain generalization.

	Token Accuracy	Token Precision	Token Recall	Token F-measure
Mitchell	N/A	94.4%	78.6%~(1.2)	85.7%
ML MSA	76.9%~(1.6)	76.5%~(1.4)	76.5%~(1.4)	76.50%
Perceptron MSA	86.7 % (0.9)	86.7%~(0.9)	86.7% (0.9)	86.7 %

Table 6: Full NP ordering accuracies; averages and standard deviations over a 10-fold cross validation. To compare directly with Mitchell (2009), we report *token* precision and recall instead of type. Our system always proposes one and only one ordering, so token accuracy, precision, and recall are identical.

the third.⁴ Our results mirror those in the previous trials — forcing a prediction costs some precision (vis-a-vis Mitchell's 2009 system), but our recall is dramatically higher, resulting in more balanced performance overall.

4.2 Full NP Ordering

We now extend our analysis to ordering entire NPs, a task we feel the MSA approach should be particularly suited to, since (unlike pairwise models) it can model positional probabilities over an entire NP. To our knowledge, the only previously reported work on this task is Mitchell's (2009). We train this model on the full NP instead of on modifier pairs; this makes little difference in pairwise accuracy, but improves full-NP ordering considerably.

As seen in Table 6, both MSA models perform quite well, the perceptron-trained MSA again outperforming the maximum likelihood model. However, we were somewhat disappointed in the performance on longer sequences. We expected the MSA to encode enough global information

Modifiers	Frequency	Token	Pairwise
		Accuracy	Accuracy
2	89.1%	89.7%	89.7%
3	10.0%	64.5%	84.4%
4	0.9%	37.2%	80.7%

Table 7: Descriminative model performance on NPs of various lengths, including pairwise measures.

to perform accurate full sequence ordering, but found the accuracy drops off dramatically on NPs with more modifiers. In fact, the accuracy on longer sequences is worse than we would expect by simply extending a pairwise model. For instance, ordering three modifiers requires three pairwise decisions. We predict pairwise orderings with 88% accuracy, so we would expect no worse than $(.88)^3$, or 68% accuracy on such sequences. However, the pairwise accuracy declines on longer NPs, so it underperforms even that theoretical minimum. Sparse training data for longer NPs biases the model strongly toward short sequences and transitivity (which our model does not encode) may become important when ordering several modifiers.

 $^{^4\}mathrm{Note}$ that the WSJ corpus is much larger than the other two, comprising approximately 84% of the total.

5 Ablation Tests

We performed limited ablation testing on the discriminative model, removing features individually and comparing token accuracy (see Table 8). We found that few of the features provided great benefit individually; the overall system performance remains dominated by the word. The word and stem features appear to capture essentially the same information; note that performance does not decline when the word or stem features are ablated, but drops drastically when both are omitted. Performance declines slightly more when ending features are ablated as well as words and stems, so it appears that—as expected—the information captured by ending features overlaps somewhat with lexical identity. The effects of individual features are all small and none are statistically significant.

Feature(s)	Gain/Loss
Word	0.0
\mathbf{Stem}	0.0
Capitalization	-0.1
All-Caps	0.0
Numeric	-0.2
Initial-numeral	0.0
${f Length}$	-0.1
Hyphen	0.0
-al	0.0
-ble	-0.4
-ed	-0.4
-er	0.0
-est	-0.1
-ic	+0.1
-ing	0.0
-ive	-0.1
-ly	0.0
Word and stem	-22.9
Word, stem, and endings	-24.2

Table 8: Ablation test results on the discriminativemodel.

6 Summary and Future Directions

We adapted MSA approaches commonly used in computational biology to linguistic problems and presented two novel methods for training such alignments. We applied these techniques to the problem of ordering prenominal modifiers in noun phrases, and achieved performance competitive with — and in many cases, superior to — the best results previously reported.

In our current work, we have focused on relatively simple features, which should be adaptable to other languages without expensive resources or much linguistic insight. We are interested in exploring richer sources of features for ordering information. We found simple morphological features provided discriminative clues for otherwise ambiguous instances, and believe that richer morphological features might be helpful even in a language as morphologically impoverished as English. Boleda et al. (2005) achieved promising preliminary results using morphology for classifying adjectives in Catalan.

Further, we might be able to capture some of the semantic relationships noted by psychological analyses (Ziff, 1960; Martin, 1969) by labeling words which belong to known semantic classes (e.g., colors, size denominators, etc.). We intend to explore deriving such labels from resources such as WordNet or OntoNotes.

We also plan to continue exploration of MSA training methods. We see considerable room for refinement in generative MSA models; our maximum likelihood training provides a strong starting point for EM optimization, conditional likelihood, or gradient descent methods. We are also considering applying maximum entropy approaches to improving the discriminative model.

Finally (and perhaps most importantly), we expect that our model would benefit from additional training data, and plan to train on a larger, automatically-parsed corpus.

Even in its current form, our approach improves the state-of-the-art, and we believe MSA techniques can be a useful tool for ordering prenominal modifiers in NLP tasks.

7 Acknowledgements

This research was supported in part by NSF Grant #IIS-0811745. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

References

- Regina Barzilay and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing
 Volume 10, pages 164–171, Philadelphia. Association for Computational Linguistics.
- Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL), volume 15, pages 201–31, Edmonton, Canada. Association for Computational Linguistics.
- Gemma Boleda, Toni Badia, and Sabine Schulte im Walde. 2005. Morphology vs. syntax in adjective class acquisition. In Proceedings of the ACL-SIGLEX Workshop on Deep Lexical Acquisition, pages 77–86, Ann Arbor, Michigan, June. Association for Computational Linguistics.
- Humberto Carrillo and David Lipman. 1988. The multiple sequence alignment problem in biology. SIAM Journal on Applied Mathematics, 48(5):1073–1082, October.
- Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings* of the ACL-02 conference on Empirical methods in natural language processing, volume 10, pages 1–8, Philadelphia, July. Association for Computational Linguistics.
- Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. 1999. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, West Nyack, NY, July.
- John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. SWITCHBOARD: telephone speech corpus for research and development. In Acoustics, Speech, and Signal Processing, IEEE International Conference on, volume 1, pages 517– 520, Los Alamitos, CA, USA. IEEE Computer Society.
- Dan Gusfield. 1997. Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. Cambridge University Press, West Nyack, NY, May.
- H. Kucera and W. N Francis. 1967. Computational analysis of present-day American English. Brown University Press, Providence, RI.
- Robert Malouf. 2000. The order of prenominal adjectives in natural language generation. In *Pro*-

ceedings of the 38th Annual Meeting of the Association for Computational Linguistics, pages 85–92, Hong Kong, October. Association for Computational Linguistics.

- Mitchell P Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. *Treebank-3.* Linguistic Data Consortium, Philadelphia.
- J. E. Martin. 1969. Semantic determinants of preferred adjective order. Journal of Verbal Learning & Verbal Behavior. Vol, 8(6):697–704.
- Margaret Mitchell. 2009. Class-Based ordering of prenominal modifiers. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pages 50–57, Athens, Greece, March. Association for Computational Linguistics.
- M.F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130—137.
- James Shaw and Vasileios Hatzivassiloglou. 1999. Ordering among premodifiers. In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, pages 135–143, College Park, Maryland, USA, June. Association for Computational Linguistics.
- Christopher White, Izhak Shafran, and Jean luc Gauvain. 2006. Discriminative classifiers for language recognition. In *Proceedings of the* 2006 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 213–216, Toulouse, France. IEEE.
- Paul Ziff. 1960. Semantic Analysis. Cornell University Press, Ithaca, New York.