Preventing Sexual Unprotected Intercourse

A Study of Prenominal Adjective Ordering

Chris Collette and Ben Newman
CS224N Final Project

June 7, 2006

1 Introduction

In matters linguistical, to paraphrase Chomsky, there is no substitute for consulting the intuitions of a native speaker. This desire grows particularly acute when one actually attempts to *produce* plausible utterances rather than merely making sense of them. One subproblem in the study of production that seems especially mired in the long-accreted intuitions of the native speaker is the problem of *prenominal adjective ordering*. Consider the following noun phrases:

```
teenage mutant ninja turtles
*ninja mutant teenage turtles
```

This particular example demonstrates that one ordering of adjectives may seem substantially more natural than another, but it is hardly the most dramatic example one might choose. In fact, I selected it precisely because the second ordering is not entirely implausible, only more awkward. Consider for a moment the extent to which your own cultural experience (whether or not you are familiar with the half-shelled heroes) colors your judgment of the options. Then pause a moment longer to anticipate the hopeless mess that one must encounter when one attempts to construct statistical models of such intuitions.

Fortunately, the news is not all bad. Statistical methods such as those employed by (Malouf 2000) have proven considerably more effective than the pre-NLP analysis given by (Goyvaerts 1968), which established that adjectives ought to be ordered according to their semantic classes, such that, to a rough approximation, the following relations hold:

$$\begin{aligned} \text{size/length/shape} &\prec \text{old/new/young} \\ &\prec \text{color} \\ &\prec \text{nationality} \\ &\prec \text{style} \\ &\prec \text{gerund} \\ &\prec \text{denominal} \end{aligned}$$

Here $A \prec B$ means that an adjective of semantic class A ought to precede a word of semantic class B. This is a partial ordering, to be sure, and it therefore underspecifies the proper ordering for adjective sequences that contain more than one adjective of the same semantic class. Intuitively, this underspecification is entirely natural: the sentiment darkening Alexander's famously *Ter-*

rible, Horrible, No Good, Very Bad Day[1] could easily withstand a slight rearrangment of the modifiers. Luckily, English prescriptive grammar generally requires that adjectives of equivalent classes be separated by commas. For the purposes of this study, I ignored comma-separated adjectives, since presumably these adjectives have no prescribed ordering.

I further restricted my analysis to the ordering of adjective bigrams, although I will discuss the problem of generalizing the results to sequences of more than two adjectives in the "Future Directions" section. The motivation for this simplification was twofold. First, data on adjective sequences is fantastically sparse. In the 100 million words of the British National Corpus, my extraction process found a scant 404,686 adjective sequences of length two or more, and the overwhelming majority of these pairs occurred only once. Second, there are only two possible orderings for a bigram, and if both are found to be acceptable, then there is no deprecated ordering. This contrasts with tri- and higher-gram ordering, in which the relationships between the adjectives may be more complex; one adjective might always come first, but the order of the remaining adjectives might be unimportant, so there may still be orderings that are strictly disallowed.

The results I intend to demonstrate are almost identical to those demonstrated by (Malouf 2000). I learned a great deal by following in his footsteps, though I regret following so closely. I chose the topic before realizing the extent to which it has already been addressed in the literature, so my original contributions will mostly be limited to the qualitative as opposed to the quantitative. My use of the first-person singular is also no mistake—for reasons beyond our control, this project turned into a one-person effort.[2]

[1] Cf. the Amazon.com page if you missed this book during your childhood.

2 Method

2.1 Corpus Considerations

For this particular application, the gigantic British National Corpus was ideal. The hugeness of the corpus is the best antidote for the inevitable sparsity of the data, and extracting sequences of prenominal adjectives requires nothing more in the way of corporal annotation than part-of-speech tags, which is precisely what the BNC offers. At 100 million words, the tagging was necessarily done by machine[3], which gave rise to anomalies like the following:

```
<w AJ0>pro-North <w AJ0>Korean
<w NN2>residents
```

where "pro-" is rightfully an adverb, and "North Korean" is a two-word adjective. But a subjective inspection confirmed that these anomalies were indeed exceptional and, moreover, random rather than systematic.

A few of the more systematic errors had solutions, while others did not.

[2] And that person's name is Ben.

[3] Using an ensemble of taggers, including Claws4, as described in the documentation: `/afs/ir/data/linguistic-data/BNC-world/docs/urg.pdf`.

2.1.1 Capitalization

I chose not to uncapitalize sentence-initial words, since there was no reliable way of knowing whether the words ought to have been capitalized anyway. In fact, I elected not to modify capitalization in any way, because it proved a useful heuristic for identifying semantic classes like nationality (obviously imperfectly, but every little vote helps).

2.1.2 Non-ASCII characters

The BNC escapes non-ASCII characters using character sequences like `–` (for – as opposed to -) and `é` (for é). I noticed the long dash first and replaced it with a simple hyphen, but once I understood the full scope of the escaped character set, I decided against trying to find ASCII equivalents for the rest of the characters. With any luck, I thought, these sequences would carry enough semantic information to be useful in the classification task.

2.1.3 Artificial Frequency of Rare Sequences Within Single Articles

Within a single article[4], an adjective sequence that is rare in general may appear several times. Although it is likely that the sequence itself will be consistently (and correctly) ordered, one of the heuristics I will describe later considers only how often a word appears first in any sequence, so this artificial frequency threatened to undermine that measurement. Nationalities were particularly bursty, as demonstrated by this excerpt from and article on Irish politics:

```
<w AJ0>Irish <w AJ0>Labour
<w AJ0>Irish <w AJ0>high
<w AJ0>Irish <w AJ0>subordinate
<w AJ0>Progressive <w AJ0>Democratic
<w AJ0>Irish <w AJ0>general
<w AJ0>young <w AJ0>political
<w AJ0>Irish <w AJ0>socialist
<w AJ0>loyalist <w AJ0>popular
<w AJ0>overall <w AJ0>divisional
<w AJ0>high <w AJ0>moral
<w AJ0>Roman <w AJ0>catholic
<w AJ0>religious <w AJ0>congregational
<w AJ0>famous <w AJ0>spiritual
<w AJ0>main <w AJ0>real
<w AJ0>Irish <w AJ0>popular
```

While the word "Irish" should crop up fairly regularly in a British corpus, instances such as this one are unrealistically dense. In order to combat the worst of this problem, I employed the UNIX command `uniq -i` in order to eliminate adjacent adjective sequences that were identical up to capitalization. This reduced the total number of sequences from 415731 to 404686, removing 11045 duplicate sequences (2.66% of the total). This correction may account for the fact that my results were slightly worse than Malouf's, since there would have been fewer exact matches between testing and training data.

2.1.4 Multi-word Adjectives

I took care to avoid the assumption that all adjectives would be delimited

[4] By "article," I mean any of the individual works included in the BNC, though not all are actually articles.

by spaces (first observed counterexample: `à la carte`). The POS tags were more reliable as delimiters, so I split the bigrams whole using the tags rather than removing the tags and then splitting on the spaces. Here's the `sed` command I used to split on the tags (and replace them with commas):

```
sed 's/ <w ...>/,/g' < file | \
sed 's/^<w ...>//'
```

I verified before using this command that there were not already any commas in the extracted sequences. I mention this in case any future students are tempted to make the wrong assumption here.

2.2 Bigram Extraction

On the intuition that anything written in C/C++ will be faster than the alternative, I elected to compile a token scanner for extracting sequences of adjectives using the scanner generator `flex`. I have since learned that beating `grep` for speed is not so easy, but I stuck with `flex` because it supports the composition of regular expressions in a way that is substantially more readable than any equivalent `grep` command. Have a look at the file `prenom.l` for the specification of the scanner. I should note here the reason why the size of my dataset was larger than Malouf's. Whereas he only extracted sequences of words tagged with `AJ0`, `AJ1`, and `AJ2` (positive, comparative, and superlative adjectives), I additionally extracted sequences including cardinal (`CRD`) and ordinal (`ORD`) numbers, and determiners (`DPS`, `DT0`, and `DTQ`).[5] This made the dataset sparser, but, I felt, more realistic.

The very first adjective-adjective-noun sequence in the BNC was the phrase "unprotected sexual intercourse." The title of the paper reflects the minimal goal of determining the proper ordering of the adjectives "unprotected" and "sexual." I am pleased to say that we were successful in this regard, and that we managed to remain (mostly) STD-free throughout the project.[6]

I divided the higher-order N-grams from the bigrams using the following `grep` command:

```
grep '\(<w ...>[^<]*\)\{3,\}' \
file-of-all-sequences
```

In order to then separate the bigrams, I simply added the `-v` switch to the above grep command. Note that changing the 3 to a 2 would not have worked, since higher-order N-grams necessarily contain bigrams. There were 384932 bigrams and 19754 N-grams of higher order (4.88% of the total), which of course were overwhelmingly trigrams. The relative size of these two data sets should give some indication as to why I was reluctant to work with higher-order N-grams; 19754 sequences in 100 million words is simply not enough data to do any meaningful training in isolation. Virtually all of the higher-order N-grams were unique, after the correction described above.

[5] A complete list of tags for the BNC can be found here.

[6] STD is an acronym for "surreptitiously transposed determiners," of course!

2.3 Memory-based Learning

Following (Malouf 2000), I decided to use the Tilburg Memory-Based Learner (TiMBL) to predict the ordering of adjective bigrams using an array of features that I generated (more on that later). The crucial insight which allows adjective ordering to be construed as a memory-based classification task is this. Given a pair of adjectives sorted by some arbitrary rule, such as alphabetical order, one can say that the pair is either correctly ordered (in the semantic sense) or incorrectly ordered. This suggests two possible classifications, Y for the already-ordered case and N for the reversed case. The alphabetic ordering of the adjectives makes it much easier to use a machine learning tool that knows very little about the problem domain, since it ensures that individual features correspond to one another across all training and testing instances.

NB: This classification paradigm is total: there is no need to mention precision versus recall, because every test instance gets classified. Even dead ties are broken randomly.

2.3.1 Morphological Features

But if adjective ordering is truly a matter of identifying semantic classes, how does one go about generating features that reflect these classes? I experimented with a few strategies, but in the end the most useful strategy was the one suggested by (Malouf 2000). Simply take the last eight characters of each adjective as sixteen individual features. This makes available to the memory-learner much of the semantic information encoded by morphology. To this I added two features indicating whether the first letter of each adjective was capitalized or not, as this attribute conveys some additional semantic information (and, for adjectives longer than 8 characters, would not otherwise be singled out). This addition improved the test-set accuracy by 0.14%, which is statistically significant given the size of the data set (though the calculation is not worth including here). This approach yielded a test-set classification accuracy of 89.47%, which is slightly better than the corresponding result in Malouf's paper (though arguably not statistically significantly so—whoa, check out that sequence of adverbs!).

2.3.2 Less-Than-Useful Brute-Force Features

Having done the Maximum Entropy portion of the second CS224N assignment, in which we were required to classify proper nouns, I remembered that it was quite useful to compile semi-exhaustive lists of first names (using the census) and movie titles (using the resources of imdb.com) in order to allow features which solved a portion of the classification problem by brute force.

Unfortunately, no such brute force approach proved very useful in this case. My partner compiled a list of some 100 words relating to age, all of which seemed as though they might fit into the phrase "___ green leather couch" (think "new," "teenage," "geriatric," etc.), but either the list was not exhaustive enough or the information was not genuinely useful, because the addition of `+` and `-` features signifying membership in the seman-

tic class new/old/young actually diminished the accuracy of the classifier by a miniscule amount, probably due to overtraining. The critical difference between this task and that of the second assignment was that the semantic classification was not the end goal in this case. While the list of age words might certainly improved the accuracy of that particular feature, it appears that the feature itself was not useful in the ordering classification. I tried a similar tactic for identifying professions used as adjectives (e.g., "ninja turtles"), but this approach was even more obviously doomed, because the BNC tagging did not reliably identify such adjective-nouns.[7]

2.3.3 Positional Probabilities as Features

(Malouf 2000) makes a startling suggestion: one might be able to model the probability that a given adjective pair is properly ordered using only the blind probability that each adjective appears first in a given pair, conditioned on the number of times it appears in any pair.

See the functions `getPositionalProbs` and `initPositionalProbs` in the script `totimbl.py` to see exactly how these probabilities were calculated. The basic mathematical insight (due entirely to Malouf) is that, when one makes the unintuitive assumption of independence between adjectives, the probability that two adjectives $\{a, b\}$ should have the order $\langle a, b\rangle$, or

$$P(\langle a, b\rangle \mid \{a, b\}),$$

is simply

$$P(\langle a, x\rangle \mid \{a, x\}) \cdot P(\langle x, b\rangle \mid \{b, x\}),$$

where x is any arbitrary adjective in the training data.[8] Remarkably, when I used only the real-valued features $P(\langle a, x\rangle \mid \{a, b\})$ and $P(\langle b, x\rangle \mid \{b, x\})$, I got a prediction accuracy almost identical to that obtained through the use of morphological features.

You may be confused by this talk of using real-valued features in a classification system that only considers the degree of overlap between sets of features. Real-valued decimal numbers should only very rarely "overlap," if at all! This is one case where the TiMBL classifier really shines out. Instead of blindly comparing the string representation of real-valued features, it recognizes them as decimal numbers, notes the total range of values, discretizes this range into 20 separate intervals, and synthesizes features for each interval.[9] How ridiculously cool is that? As you can imagine, this means that processing real-valued features takes significantly longer than processing simpler features, but since TiMBL is using a trie for features behind the scenes, the increase in processing time is not devastating. This approach yielded a test-set classification accuracy of 89.02%. This result is remarkably consistent with (Malouf 2000).

[7] For what it may be worth, the list was taken from Wikipedia.

[8] In cases where the training data does not provide any evidence about a or b, I simply return 0.5 for the corresponding positional probability. This is in line with the assumption of a flat prior, and also avoids division by zero.

[9] Described in the TiMBL manual on page 21.

2.3.4 Combining Strategies

Supposing that the information conveyed by the morphological features would not perfectly overlap with the information contained by the positional probabilities, I ran another test which simply added the positional probabilities as features to the morphological feature set. The improvement was modest, but it was enough to push the accuracy up to 90.17%, which was certainly a psychological milestone.

3 Discussion of Results

3.1 Theoretical Limit

You might wonder just how high the accuracy could possibly be; after all, this experimental design assumes that the data are actually consistent, that there will not be any sequences of adjectives that are ordered in one way somewhere in the data and another way in another place. Running the classifier on the training data is the best way to find out what proportion of the data is fundamentally inconsistent. This test yielded an accuracy of 98.09%, which might seem surprisingly high. Most of the consistency comes from the sparseness of the data: the fewer instances of a given set of adjectives are present, the smaller the chance of finding counterexamples in the data, especially if one correct ordering is much less common than the other. In all likelihood, accounting for these inconsistencies is out of the practical reach of statistical methods, although a system which scored adjective sequences rather than simply classifying them might be able to recommend all sequences that fell above a certain threshold score.

3.2 Final Tuning

After reading through the manual for TiMBL in more depth, I settled on a combination of settings that best balanced predictive accuracy with processing time. The final command to the `Timbl` executable went something like this:

```
Timbl5/Timbl -mM -k 5 -f train \
-t test
```

The `-mM` setting selected the *Modified Value Difference Metric*, with which "each pair of values of a particular feature is assigned a value difference," so that the similarity metric for comparing testing and training instances now becomes the "pairwise differences... computed for each pair of values in each feature."[10] This seemed reasonable, because the adjacent elements of each pair of morphological features (the letter-grams) have more to do with each other than they have to do with, say, the positional probability features (once discretized). This similarity metric contrasts with the default *Weighted Overlap* metric, which simply weights each feature according to its information gain and then uses the classification of training instance which has the highest weighted overlap in features with the testing instance.

The `-k 5` setting compares a testing instance to its five nearest neighbors in the training data, and takes the classification to be the majority vote of the five neighbor

[10] From the manual, page 13.

classifications. The manual explicitly recommended using this setting in conjunction with the `-mM` setting, since the *MVDM* setting ends up making fewer feature comparisons than the *WO* metric.

3.3 Results

The results of our approach are summarized in the following table: The final number is

Table 1: Summary of Accuracies

Feature Type	Classif. Accuracy
Morphological	89.47%
Positional *P*s	89.02%
Combined	90.17%

somewhat less than the corresponding result in (Malouf 2000), which was 91.85%. I attribute the difference to my cleaning up of the BNC data during the bigram extraction phase, as explained earlier. With fewer identical phrases between training and testing data, the testing accuracy did not benefit as much from the fact that the two data sets were drawn from the same corpus. I am not prepared to speculate about how my inclusion of ordinal and cardinal numbers and non-article determiners influenced this final pecentage, though any difference in methodology could have played a role in producing the different final results.

4 Future Directions

I deliberately opted not to explore alternative approaches to adjective ordering, because the literature indicated that they performed more poorly in isolation. I will describe two of these methods for completeness' sake, however.

4.1 Direct Evidence

In this approach, the judgment call about which order to select for an adjective bigram is based entirely on the relative counts of each ordering of that bigram in the training data. The comparison can be qualified by a significance test in order to distinguish, say, a 2-1 count from a 20-10 count, but this ultimately does little to overcome the fundamental sparsity of exact bigrams. Two former CS224N students, Ari Greenberg and Praveen Srinivasan (2003), complain that while precision is admirable (in excess of 90%), recall is gut-wrenchingly poor (less than 50%).[11]

4.2 Transitivity

In this approach, first suggested by (Shaw and Hatzivassiloglou 1999) we treat the training data as a directed graph of words connected by edges that represent that the source adjective precedes the destination adjective. These edges are weighted according to the relative frequencies of the two possible orderings of the adjectives. When faced with a bigram that does not appear in the training data, one can perform two minimum-cost-path searches, one from the first adjective to the second, and the other from the second to the first. Whichever cost is smaller dictates the preferred ordering. While this

[11] Their paper is available here.

approach is the most effective way of generalizing to higher-order N-grams that I came across, the results for bigrams are not particularly good—Malouf achieved a test accuracy of 83.91%. The reason seems to be that, while this approach handles the case where the exact bigram has not been seen, it still fails to produce a meaningful result when neither of the words in the testing instance have been seen before, which is quite a frequent occurrence.

The real death knell for this approach, however, is the computational intensiveness. I wanted to stay away from anything that was $O(n^3)$ in the size of the input, given that the input was in the hundreds of thousands of sequences, most of whose words were unique. I toyed with the idea of devising an admissible heuristic for this sort of search, so that the A* algorithm could be used for an average-case improvement in efficiency, but, given the way the weights are calculated, the only provable underestimate of the total remaining cost would be 0. Dijkstra's Algorithm was the best option.[12]

It is quite possible, however, that this method could be combined with the two methods I ultimately used in order to produce an even more effective ensemble of classifiers. There was not time to explore this possibility in sufficient depth. (Greenberg and Srinivasan 2003) applied this approach with some success to the problem of trigram ordering, by simply filtering out those permutations which contained out-of-order pairwise bigrams.

[12]Though Floyd's all-pairs shortest-path algorithm could be used if one just wanted to calculate all the weights at once.

5 Works Cited

Daelemans, et al. "TiMBL: Tilburg Memory-Based Learner (Reference Guide), version 5.1". [2004]. Available online.

Greenberg, A. and Srinivasan, P. [2003]. "A Survey of Algorithms to Properly Order Adjective Trigrams". Available online.

Goyvaerts, D. L. [1968]. "An Introductory Study on the Ordering of a String of Adjectives in Present-day English". *Philologica Pragensia*, 11: 12-28.

Malouf, R. [2000]. "The order of prenominal adjectives in natural language generation". Available online.

Shaw, J. and Hatzivassiloglou, V. "Ordering among premodifiers". Proceedings of the 37th Annual Meeting of the American Association for Computational Linguistics (ACL'99), pp 135143. University of Maryland, 1999.[13]

6 Notes on the Code

See the README included with our electronic submission.

[13]This citation was taken from (Greenberg and Srinivasan 2003)'s works cited list, but the reference was made by both them and Malouf, so it seemed to deserve separate mention here.