Using Semantic Similarity to Determine Adjective Noun Combinations

Ido Milstein
04730545

May 7, 2000

1 Introduction

In this Paper I describe two disjoint attempts to use semantic similarities between words, to improve probability estimations of adjective noun combinations. The first approach, which is rather indirect, uses PCA (principal component analysis) to smooth the joint probability $P(Noun, Adj)$, and uses these principal components to approximate $P(Noun|Adj)$. The second approach, modifies the probabilities directly by using an idea that draws from nearest neighbors approaches. The idea is to transform the similarities into a probability measure, and use that measure to find the expected probability $P(Noun|Adj)$. Both approaches will be discussed in a greater detail later.

2 Implementation

I implemented the project mainly in MATLAB. I started by writing a C++ code, that transforms the data from a list of adjectives and nouns, to a list of adjective numbers and noun numbers. I then used MATLAB to read this file, find $P(Noun, Adj)$ from it, and used scripts that I wrote for the two approaches I mentioned, to manipulate the data. The code is fairly simple, and regularly commented. More details can be found in the README file, and in the source files them selves. Note that the probabilities written are as small as 10^{-10}, but none of them is zero.

3 PCA

3.1 Method

My first idea was to try to use Principal Component Analysis to find eigenadjectives, that would be defined by the nouns that usually go with them, and use these eigenadjectives to smooth the probabilities by taking only the first hundreds of vectors. This approach had one great problem, which is that when we drop some of the eigenvectors we get negative

numbers. This may not be a problem in the general case, but when we deal with probabilities, thinking of a way to reassign these negative probabilities becomes a major problem. I tried using linear transformations, that worked poorly, and ended up using a small threshold to cut off the negative values, and then normalized the matrix to sum to 1. While this had nice results (7.25 cross entropy), I noticed that the improvement was mainly determined by the threshold that I put, and not the by amount of principal components that I took. So this was practically like choosing a λ for the normal smoothing function.

3.2 Results

Other than the cross entropy results which I have mentioned earlier, I think that it is interesting to note that many of the eigenadjectives that I have looked at, didn't seem to have anything to do with any prototypical adjective that I could think of. Some of them, however, did seem like natural prototypes, like the one on Table 1, that shows the nouns that form the second eigenadjective (at least those that had the highest magnitudes).

Noun	Magnitude
executive	0.92511
director	0.037206
minister	0.012917
officer	0.3404
manager	0.011283
file	0.014605
counsel	0.051464
official	0.020817
concern	0.01171
wife	0.012913
judge	0.046179
prosecutor	0.011132
lawyer	0.0132
adviser	0.018013
justice	0.043704

Table 1: *The table shows the **nouns** that have the greatest magnitude for the second eigenadjective. The **adjectives** that had the highest projection over this eigenadjective are:'former', 'financial', 'chief', 'senior', 'top' and 'executive'. This eigenadjective is something close to the natural idea of words that describe a person in a high position.*

Graph 1 Shows the drop of the singular values of the joined probability matrix. Taking the first 200 vectors gave a cross entropy of 7.69 (with the above threshold). Cross entropy actually only got better when I took more and more vectors, with the peak of 7.23 for the case where I took all 600 vectors. Note that there are 600 vectors because that's the minimum

between the number of nouns and the number of adjectives, and so is the maximum possible rank of the probability matrix.

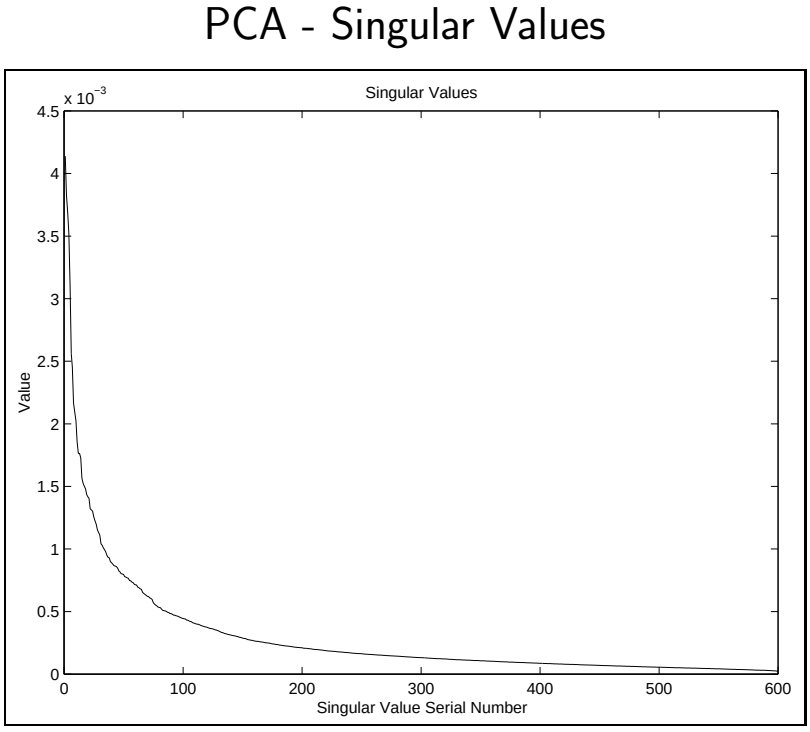

Figure 1: *The Magnitude of the singular values. Even though it seems like the first 150 values capture almost everything, taking less than 300 eigenvectors gave results worse than the baseline.*

4 Semantic Similarity

A common problem in machine learning and NLP, is how to get good estimates of probabilities, given a limited data set. When dealing with words this problem is specifically great, because even when we restrict ourselves to adjective-noun pairs, we get many of them, and a great deal of these combinations appear very rarely. So, what we may hope to do, is to use models that use our knowledge about language to form structural dependencies between words. In our case of adjective-noun pairs, we try to incorporate the knowledge that similar adjectives appear usually with similar nouns. However, this entails two problems. First - what exactly is similarity? and how can we measure it? Second assuming that we have found such a measure, how can we use it to improve our probability estimations?

4.1 Measuring Similarities

None of these questions has a clear cut answer. It seems that if we want to measure for example a similarity between adjectives in the context of the nouns that follow them, a good

way may be to see how similar the probability distributions that they create over the set of nouns are. So, this means that we are measuring some distance between $P(Noun|Adj_1)$ and $P(Noun|Adj_2)$. But this helps us only a little because now we need to define how similar two probability distributions are. Since these are defined on different probability spaces and so do not encode the same thing, I decided to use Manhattan distances, rather than entropy based ones. This means that probabilities that overlap on 50% of their space get a score of 1, and probabilities that don't overlap at all get a distance of 2. Notice that these scores are symmetric, so

$$Score(Adj_1, Adj_2) = Score(Adj_2, Adj_1) = \sum_{nouns} \|P(Noun|Adj_1) - P(Noun|Adj_2)\|$$

4.2 Using Similarities to Improve Probability Estimations

The way I tried to use this measure of similarities was to transform it into a probability measure, and use that measure to find the expected $P(Noun|Adj)$. First I needed to get a new scale for the similarities, that would give a higher probability to similar adjectives than to non-similar ones. I used the following function:

$$P^*(Adj_1, Adj_2) = C^{2-Score(Adj_1,Adj_2)} - 1$$

This means that adjectives with non overlapping probabilities get a zero, and those that have 50% overlap get a probability that is $\frac{1}{C}$ of the probability for an adjective with itself. The selection of the constant C is described a little later. This probability distribution is then marginalized to get $P(Adj_1|Adj_2)$. I then define

$$P(Noun|Adj) <-- E_{P^*}[P(Noun|Adj)] = \sum_{adj_i} P(Noun|Adj_i) * P(Adj_i|Adj)$$

The intuitive way to look at it, is that the adjective that we observe may 'behave' like several other adjectives, and this 'behavior' adjective is represented by Adj_i. So we get that the nouns are independent of the adjective given that we know what adjective it 'behaves' like.

4.3 Results

The Similarity matrices that I got seemed quite good. Many words that came out close seemed like obvious choices, but this seemed less so for rare words. So for example 'lower' and 'higher' had a high score, and so did ('few', 'several'), ('younger', 'older') and others. But I also got 'mountainous' and 'darkened', and couples like ('cold', 'persian'), ('25th', 'silver') and others.

My experiments on the adj16-20 data showed that $C = 5$ gave the best results (6.85 cross entropy), but this was more or less the same for C in the range 4-10. Higher Cs (20-100) gave better results on adjectives that had a low initial cross entropy ,like 'alive' and 'blond' (because they 'smooth' less), but lower Cs yielded a greater overall improvement. One may try to find good ways to distinguish between adjectives that need high Cs and those that need low Cs, and give different Cs to different groups, but I did not do that.

I did try to use the same technique to estimate $P(Noun|Noun)$, and found out that the best constants C for that, were in the range 20-30. These yielded a cross entropy of 7.00. I tried to use both noun similarities and adjective similarities at the same time, with different constants, but the results were not better than those for adjective similarities alone. Probably because of over smoothing.

Another idea that I had was as follows. Since I use $P(Noun|Adj)$ to estimate the similarities, and use the similarities to improve our measurements of $P(Noun|Adj)$, I could try to repeat this process to improve the results. Experimenting with this proved that I was wrong. It seemed to have over smoothed the probabilities, and so results were even worse than baseline.

5 Conclusions

Even though PCA seems like a quick and promising technique, I was quite surprised to see how poorly it worked (as I wrote, the best results I achieved with it where when I took all the eigenvectors).

On hind-site, I see that my second idea was very much like having a very short HMM, with 2 observed variables, and 2 hidden variables. The 2 observed variables being the adjective and the noun, and the 2 hidden ones where the adjective 'behavior' and noun 'behavior'. I guess EM would have been a better thing to use to learn the probabilities of the hidden units, than what I tried. What I did is actually to use heuristics to estimate $P(hidden-adjective-meaning|observed-adjective)$ (and the same for the nouns). I got some nice results (6.85 cross entropy), but I would guess that learning the probabilities the right way, with EM, would have resulted in even better results, and would have probably enabled the use of both adjective to adjective similarity, and noun to noun similarity at the same time.

References

[1] Christopher D. Manning and Hinrich Schutze, *Foundations of Statistical Natural Language Processing (229-261), 1999.*