# Learning how Adjectives Affect the Meaning of Phrases in a Vector Space Model

Jayant Krishnamurthy

October 1, 2009

Many NLP applications use a vector space model to represent the meaning of words and short phrases. In this model, every phrase is represented by a vector of co-occurrence counts with a set of contexts. Depending on the exact construction of the model, these contexts can be other words, short phrases, or even known facts about the phrase. The intuition behind the vector space model is that words with similar meanings are used in similar ways, and therefore will appear in similar contexts. We therefore expect similar words to be located near each other in the vector space.

One limitation of vector space models is that they do not represent multi-word phrases particularly well. We intuitively expect the vector for a phrase like "red ball" to be related to the vector for "red" and the vector for "ball," but the vector space model does not explicitly represent this relationship. The goal of this project is to explore the relationship between a phrase's vector and the vectors of its constituent parts. Specifically, I propose to treat adjectives as functions that map noun phrases to other noun phrases. I will then use linear regression to estimate the function computed by several adjectives. Discovering the relationship between the meaning of a phrase and the meanings of its constituent parts will allow computers to predict the meanings of novel word combinations. Additionally, if this approach is extensible to other parts of speech, it can be used to construct semantic representations of arbitrary phrases, which may be useful in a variety of NLP tasks (e.g. language modeling, see [3]).

# 1 Prior Work

Other researchers have proposed several models for composing the meanings of vectors [2, 4]. These works have searched for a canonical model for combining any adjective and noun. That is, these works look for a function of the form $f : (\text{ADJ}, \text{NP}) \rightarrow \text{NP}$. These papers have found that multiplying two word vectors is the best predictor of the word vector for the resulting phrase. This multiplicative model is somewhat analogous to computing the logical AND of the noun's attributes and the adjective's attributes, that is, selecting the attributes that both vectors have in common.

These previous approaches use rather simple models to combine word vectors. I propose a slightly different and more powerful approach: I treat adjectives as functions that map NP vectors to NP vectors. That is, adjectives are functions of the form $\text{ADJ} : \text{NP} \rightarrow \text{NP}$. I then propose to learn the function computed by each adjective using data.

# 2 Learning Adjective Functions

I propose to model adjectives as linear operators on noun phrase vectors. Say that a noun phrase "NP" is represented as an $n$-context column vector $v$. Then an adjective "ADJ" is an $n \times n$ matrix $A$, such that $Av = w$, where $w$ is the vector representation for the combined phrase "ADJ NP".

In this formulation, we can learn the function computed by an adjective using linear regression on pairs of NP vectors. In each pair, the first NP is an arbitrary noun phrase, and the second NP is the same noun phrase with the adjective prepended. Place the first NP of each pair into a matrix $V$ and the second into a matrix $W$. We can then learn $A$ using linear regression, as $AV = W$. Note that each row of $A$ predicts the value of a single row in $W$, and therefore can be learned independently of every other row. The formulation still requires regularization, however, as each row of $A$ has many more parameters than there are training examples.

This problem formulation is more general than previous approaches. It generalizes the multiplicative model, which is represented by a diagonal $A$ matrix. If the $V$ matrix includes an extra row of 1s, the model also generalizes additive models and models that combine multiplication and addition. Considering this property, we would expect this model to predict the resulting NP vector at least as well as previously tested models. In addition,

this model represents more interesting, adjective-specific dependencies. For example, it can represent the fact that the adjective "red" will affect the color properties (or color-dependent contexts) of an object, but nothing else. Finally, it allows us to use data to learn the adjective functions.

For my project, I will learn functions for a large set of adjectives. From preliminary experiments using regularized linear regression on the `NPContext1` data set, I expect to encounter several sparsity-related problems:

1. There are very few NP vector pairs for any given adjective.
2. Every NP vector occurs in very few contexts, meaning that for a given set of NPs, many contexts occur for only one NP.

The primary way I hope to address this problem is by using the `NPContext2` data set, which contains approximately 20 times as many NP vector pairs for each adjective. I can also use dimensionality reduction to reduce sparsity in the observed counts.

Non-literal expressions also seem to be a problem for training the adjective functions. Of the 16 NP vector pairs for the adjective "red" (in `NPContext1`), around half have non-literal meanings (e.g. "red meat" is meat that is red, but it also has other implications). Using the larger data set will reduce the impact of these expressions as (on visual inspection) it contains many more literal uses of each adjective. It may also be possible to perform some preprocessing and select a small set of contexts which should be affected by an adjective. This process could use the differencess between NP vector pairs to filter out contexts with unsystematic variations. Limiting the functions to only relevant contexts may improve the quality of the predictions and also the intuitive interpretability of the model.

I will evaluate the performance of my adjective functions by testing them on unseen test examples and measuring the mean squared error and $R^2$ of their predictions. I will also measure the angle between their predicted phrase vectors and the actual phrase vectors.

# 3 Data Sets

For my project, I will use the provided `NPContext2` matrix and a list of common adjectives, available online (e.g., from `http://www.esldesk.com/vocabulary/adjectives`).

# 4 Applications

Once I work out the form for the adjective prediction functions, I plan on using the functions to perform one or two NLP tasks. The first task, idiom detection, should be a relatively straightforward application of the adjective prediction functions. The second task, paraphrase detection, is more ambitious and requires extending this work to other parts of speech.

## 4.1 Idiom Detection

I propose to use these learned adjective functions to perform a task of idiom detection. Idioms are expressions like "red herring" which do not mean what they literally say. I define the task as follows: given the adjective function for "red", the NP vector for "herring," and the NP vector for "red herring," determine how likely "red herring" is to be an idiom. My basic approach is to predict the NP vector for "red herring" using the adjective function, then compare the predicted vector to the actual vector. If the vectors are different enough, I will label the phrase as an idiom.

## 4.2 Paraphrase Detection

Another possible application involves extending the basic approach of this project to other parts of speech. Given appropriate context vectors, we can learn other functions such as V : NP $\rightarrow$ VP. If it is possible to build up a significant set of these learned functions, I will consider doing a paraphrase detection application.

Paraphrase detection is a task in which a program is given two sentences and must determine if they are paraphrases of each other. Previous work in this area has combined word similarity metrics (e.g. using WordNet) in a somewhat ad-hoc framework to measure sentence similarity [1]. A set of learned part-of-speech functions would provide a principled basis for combining word similarity measures: use the functions to map each sentence to a vector, then use vector similarity to measure sentence similarity.

# 5 Schedule

By Halloween, I will learn reasonably accurate functions for a small set of common adjectives using the `NPContext2` data set. Depending on how much the additional data helps, this may require a significant amount of trial-and-error, as I'll have to investigate dimensionality reduction and various methods for filtering contexts. I will also attempt to learn functions for less common adjectives and investigate how sparse data affects the learned functions.

After Halloween, I will work on the idiom detection application and attempt to extend my work to other parts of speech.

# References


[1] Courtney Corley and Rada Mihalcea. Measuring the semantic similarity of texts. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 13–18. Association for Computational Linguistics, 2005.

[2] Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[3] Jeff Mitchell and Mirella Lapata. Language models based on semantic composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 430–439, Singapore, August 2009. Association for Computational Linguistics.

[4] Dominic Widdows. Semantic vector products: Some initial investigations. In *Second AAAI Symposium on Quantum Interaction*, 2008.