Representing Noun Compounds in Semantic Quad-Space

Kiran Vodrahalli

Princeton University

January 6, 2015

1. Introduction

Every problem can be reduced to a translation problem. A bold statement – nevertheless, consider that mathematics is the art of translating abstract ideas into a precise language. Computer science has foundations in describing routines for computers to follow in machine language, and the search for better algorithms is simply translating the computation of a quantity from a less efficient process to a more efficient process. The ability to represent what an algorithm means is well-defined. We have the precise language of mathematics to account for all details.

The language with which we speak to each other, as humans, is however not precise. There are many ambiguities. Natural language interacts with reality, and is unavoidably influenced by many aspects of the universe that humans do not yet understand. Our modes of communication are unavoidably imprecise. The lack of precision is amplified by the fact that sometimes we ourselves do not know what we want to convey.

Yet it is very desirable to have a method for automatically and quickly parsing vast amounts of natural language, and understanding what is meant – to reduce the cascades of word phrases and punctuation into concrete, irreducible facts and inferences. Aside from being interesting for its own sake, natural language understanding has multitudes of applications to real world problems.

If computers were able to understand unstructured language, we would be able to build ontology networks of journal articles in the biomedical sciences, and make scientific knowledge more accessible to both researchers and laymen. We would be able to parse the news for the purpose of understanding the impact of world events on the financial markets; we would be able to overcome language barriers with automatic language translation. We would be well on our way to understanding the human mind.

Thus there is both academic and practical desire to solve the problem of representing meaning with numbers.

1.1. Philosophy of Meaning

How can we address the problem of representing concretely the abstract notion of meaning that humans seem to possess? There are two relevant hypotheses that pervade the natural language processing community.

The first is the distributional hypothesis, first posited by Harris in 1954. In short, word meaning is a function of the distribution of a word in all its contexts **[Harris1954]**.

Second is the geometric metaphor of meaning. Born from a hypothesis that humans attach a notion of distance to similarity between concepts, the geometric metaphor of meaning posits that meanings are locations in semantic space, and semantic similarities can be represented by geometric distances **[Sahlgren2006]**.

Together, these two notions enable the concept of a semantic vector space, which we will define shortly.

1.2. Definitions

Now we take an interlude for a few definitions that will be useful throughout the paper.

Corpus:

A corpus is a body of texts, usually in a single language. They can center around a certain mode of language (for instance, news writing, or blog posts, or television sitcom speech), or they can be inclusive of a whole variety of modes of language.

Semantic Vector Space:

A semantic vector space is a set of words mapped to a set of vectors in $\mathbb{R}^n$. Typically, it is represented as a matrix, where the rows are the individual words or word phrases, and the columns are contexts. This matrix is known as a **word-context** matrix. A context is any sort of pattern associated with the row word. Suppose the row word is "boat". One column context may be the seven words between "boat" and the appearance of the word "water". Another could be simply the word "water" appearing within a radius of 3 words from an instance of the row word "boat". The contexts can be very complicated or very simple. The values contained within a word-context matrix are frequencies. For instance, for the row word "boat" and the column context "water appearing within a three-word radius of the row word", the value in that row and column value of the word-context matrix would be the number of times the word "water" appears within a three-word radius of "boat".

A word-context matrix cannot be described without a corpus. In order to calculate the frequencies of the row words showing up in the various column contexts, we search the corpus for all the instances in which the row-column combination happen. Depending on the contexts, word-context matrices can often be very sparse, necessitating the application various transformations to the original word-context matrix.

Similarity of Words:

From a human's perspective, what does similar mean? In order to evaluate the computational model against real human semantics, we need some ground truth approximation. In this section, we describe and justify what a notion of similarity should be.

Similarity comparisons for a pair of single words depends on the types of word. If both words are nouns, what we may mean by 'similar' is something along the lines of a domain comparison. For instance, we might say that 'car' and 'bus' are similar, since they

are both vehicular modes of transport that both travel by land.

For adjectives (again, comparing a single adjective to another single adjective), the situation may be a bit different. Here we may be more concerned with the aspect the adjectives qualify. For instance, 'ugly' and 'beautiful' both qualify subjective judgements of aesthetic attractiveness. Synonyms and antonyms may both be considered similar.

The list goes on. In this paper, we seek to have a qualitative understanding of the human understanding of similarity between noun compounds, and then convert the qualitiative into the quantitative.

Noun Compound:

A noun compound is a compound word with two components, both of which are nouns themselves: the modifier and the head. For instance, the word "paper cup" is a noun compound with head "cup" and modifier "paper". The head is the base word. It is almost always true that a "paper cup" is a "cup". It is not necessarily true that a "paper cup" is "paper". For instance, consider the noun compound "heart attack". While a "heart attack" is an "attack", it is not a "heart". The modifier qualifies the sense in which the head word is taken to mean. In some cases, a noun compound is an idiom (for instance, a "fire sale" is not a sale of fire, but rather a sale where prices are extremely discounted).

1.3. Paper Objective

The goal of this paper is to provide a semantic vector space model for noun compounds. That is, to quantify the meaning of noun compounds and their interactions with other words and noun compounds. The metric of our success will be comparison to a human judgement of similarity between two noun compounds. Our first step must be to provide a quantitative representation for noun compounds. The second step must be to define an unambiguous method of calculating a similarity score between two such representations. We will have succeeded if together, the two parts of our model are able to reproduce human scores of similarity on pairs of noun compounds.

1.4. Motivation

Why do we focus on representing noun compounds specifically in this paper? First of all, the similarity of word phrases (of length n) is an necessary object to consider in the future study of the quantification of the meaning of language. Noun compounds are a simple case ($n = 2$), yet with sufficient complexities that provide interesting interrelationships between the components of the noun compound.

Noun compounds themselves are interesting in the literature for practical purposes. First of all, standard techniques do not work on them; they cannot be simply ignored in natural language processing (NLP) routines. They are important in machine translation, information retrieval, and information extraction, particularly in medicine, where there are many noun compounds (an example: human colon cancer cell line, which has many possible parsings [**Nakov2013**]). For a computer doctor, knowing how to comprehend these medical-term noun compounds would be a necessary skill.

1.5. The Relation of Similarity to Meaning

As we have already noted, a primary goal of computational linguists is to create a computational model of meaning. As 'meaning' is an imprecise term **[Sahlgren2006]**, how can we possibly check whether our models are sensical?

One approach, specific to vector-based models, is to cluster vectors and check whether the vectors in the clusters all belong to a certain ontological class. For instance, hierarchies like that of WordNet **[Fellbaum1998]** may be used. Or, if we note that all the words close by are related to the topic of government, we might also say they are similar.

Therefore, similarity is a useful concept in ascertaining whether or not the computational models we build actually map to real-world notions of meaning. Previously we defined a human notion of similarity with respect to words. In this paper, we are concerned with noun compounds, and so take this opportunity to extend our notion of the similarity between word phrases to noun compounds. We pause here to give a brief description of what is meant, and promise to go into more detail in following sections of the paper.

The clustering approach described above is one way of defining what we mean when we say two words are similar. The essence of the clustering definition of similarity boils down to a notion of geometric distance: When we cluster vectors, the vectors are nearby under some metric. Typically, the metric used is that of cosine distance (the dot product of the norms of the vectors).

Before, we were dealing with a single word – we had a vector space representing each word by a vector. What happens when we increase the number of words to two? To n? We need an approach that generalizes word similarities to phrase similarities. In order to arrive at a sensible definition, we must first consider what humans think of as similar sentences.

Computationally, the extension to phrase similarity can take two paths. First, vector compositions. Second, similarity compositions. We will be dealing with noun compounds in this paper, so we are concerned with the case $n = 2$, with the added special property that both words are nouns.

The vector composition approach generally tries to reduce the n vectors into either one vector, or a matrix of some form. Then the idea is to compare the resultant vectors/matrices.

In contrast, the similarity composition approach advocated in **[Turney2012]** tries to first come up with a similarity structure function between two phrases. That is, a function that takes in a set of distance comparisons between the words from first and second phrases, and then acts on them to produce a similarity measure. A similarity structure function allows for more variety in the methods of computing similarity than the vector composition approach when it results in a single vector. The vector composition into matrix approach may be equally expressive; however, the issue is that as n increases, the size and complexity of the representation become prohibitively large. For $n = 2$, there is not too much of problem.

In this paper we compare the dual-space model from **[Fyshe2013]** that uses vector

composition to a new quad-space model that uses similarity function composition.

1.6. Outline

We provide a brief outline of the remainder of the paper.

In **Building the Model**, we provide the methodology and reasoning behind the construction of the vector space model for noun compounds that we implement in this paper. First we briefly describe the space we are attempting to represent, and then plunge into the components of its representation. For each of the four distinct spaces, we describe how it is constructed and the associated rationale. We describe how to construct the final version of each of the spaces after producing each word-context matrix. Then we discuss functions that take as input the representations of two noun compounds and output a similarity measure. This is done in two parts: first we describe the notion of a similarity structure function, and secondly we describe the building blocks of the structure function, the similarity composition functions which compose similarities. Finally we describe the similarity functions we actually used.

We specify our testing scheme in **Model Evaluation**. First we explain the overall testing scheme for the vector space model we built, and how we chose the 99 noun compounds we tested. Then we describe how we acquired the responses from Mechanical Turk: the instructions given to the Turkers, the test data (noun compound pairs), and the actual generation of the responses. After that, we describe the way by which we parsed the data and turned it into a yes/no similarity scores for each noun compound pair. Then we describe which vector spaces we built from the **[Fyshe2013]** paper, and how we went about doing so. At the end, we explain both methodologies we used for scoring the vector space models.

The **Results** section first presents both sets of results, and explains them. We then list the errors each VSM model made.

We then finish the paper by discussing plans for **Future Work**, and providing some **Acknowledgements**.

Following the end of the paper are the three appendices: **Appendix A: Word-Context Matrix Processing Steps**, **Appendix B: Describing the Corpora**, and **Appendix C: Implementation Details and Future Usage**. We finish with the **Citations**.

2. Building the Model

We build two versions of our quad-space model, one based on the Corpus of Contemporary American English **[COCA]**, and the other based on the Corpus of Global Web-based English **[GloWbE]**.

Note that the two models are the same in structure; their only differing feature is the corpus used to create them. In the tests, we will refer to the former as the COCA model, and the latter as the GloWbE model.

See **Appendix B: Describing the Corpora** for a more detailed discussion of the corpuses.

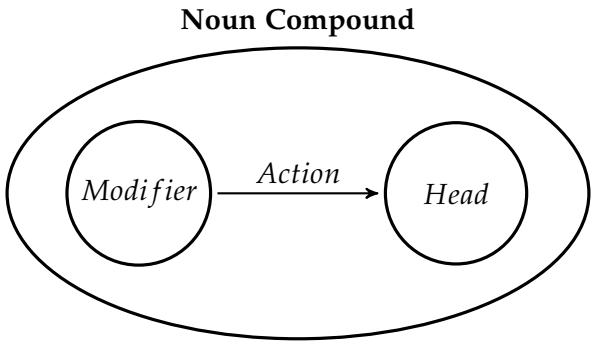


Figure 1: *Components of the Noun Compound Model*

Using the COCA corpus, we picked 99 noun compounds to find representations for which we would test, which can be seen in **9.2**. We defer the discussion of how we chose them and why to the **Model Evaluation** section. There were a total of 106 distinct words in these noun compounds. Let $\mathcal{NC}$ describe the space of noun compounds, and $\mathcal{W}$ describe the space of distinct words.

2.1. The Quad-Space Model: An Overview

The component model of a noun compound is described in **Figure 1**. Our goal is to come up with a representation of the noun compound that accurately reflects these components. Then we need to describe a function that is able to output a similarity score for two noun compounds. Our approach is guided by the composition of similarities as opposed to vectors. The similarity function should take as input two representations of two noun compounds, and use the distances between various component pairs as inputs to a similarity structure function, which then composes the input similarities to output an overall similarity score.

Our representation for a single noun-compound will be a 5-tuple of vectors, the components of which live in four distinct vector spaces. We name the four component spaces (giving rise to the quad-space name in the title). They are the domain space $\mathcal{D}$, the action space $\mathcal{A}$, the qualifier-head space $\mathrm{Q}_{\mathcal{H}}$, and the qualifier-modifier space $\mathrm{Q}_{\mathcal{M}}$. We will provide a full definition in the following sections.

Our 5-tuple can be represented as follows: $(v_1, v_2, v_3, v_4, v_5)$. We have $v_1 \in \mathcal{D}$, $v_2 \in \mathcal{D}$, $v_3 \in \mathcal{A}$, $v_4 \in \mathrm{Q}_{\mathcal{H}}$, $v_5 \in \mathrm{Q}_{\mathcal{M}}$. v_1 is supposed to be a domain representation of the full noun compound (represented by all contained inside the oval in **Figure 1**). v_2 is a domain representation of only the head component of the noun compound. v_3 is a partial representation of the action component. v_4 is a partial representation of both the action component and the modifier component. v_5 partially describes the modifier component.

Now that we have this model for a noun compound in our minds, let us describe the individual spaces in more detail, and also justify the reasoning behind the quad-space approach.

2.2. Domain Space $\mathcal{D}$

We take the idea of domain space from **[Turney2012]**, and modify it slightly. In Turney's paper, the domain space is part of his dual-space model, and is supposed to represent the subject matter or domain of a word. For this paper, the rows of domain space are single-word nouns or noun compounds ($\mathcal{NC} \cup \mathcal{W}$), and the columns are the set of all nouns closest to the row-word in a window with some radius r. In our construction, we set $r = 4$ (this means our window is four words on both sides of the row-word). Labeling each word from 1 to 9, we seek to find the closest left noun with index i such that $5 - i > 0$ is maximal. For each window, we seek to also find the closest right noun with index j, such that $j - 5 > 0$ is minimal. The left and right nouns get added to the set of words in the columns, and we count frequencies.

The intuition here is that we want a representation of the subject matter of the noun compound, and often, we get a notion of subject matter from other nouns. We restrict the window size to be relatively small so that we stay within bounds and avoid picking up too much noise. We again restrict when we only pick the closest left and right words, to avoid picking up too much noise.

There were two resulting $\mathcal{D}$ spaces. Both $\mathcal{D}$ spaces had $99 + 106 = 205$ rows.

The COCA $\mathcal{D}$ space had 217438 columns, and the GloWbE $\mathcal{D}$ space had 188535 columns.

In our model for the noun compound, we use $\mathcal{D}$ two times: first to capture a representation of the head word, and secondly to capture the domain-meaning of the whole noun compound.

2.3. Action Space $\mathcal{A}$

In the Dual-Space Turney paper, there is a notion of function space **[Turney2012]**. Turney defines function space as intending to represent the relation between two words by using verb contexts. Verb contexts are patterns of words containing the row-word or row-phrase. A verb context might be 'rowed on X', where X is row-word. For instance, suppose that the row-word is 'water'. Then, it is likely that this value of the matrix has a non-zero frequency count, since rowing often takes place on water. The idea is that we can use verbs to understand the functional power of individual words. In Turney's paper, he goes on to calculate this function space for both words in his setup, and combines the similarities in different ways.

We take a slightly different approach, though we retain the notion of using verbs to characterize function. First we introduce the notion of an action for a noun compound. The action is the way the modifier acts upon the head word in the noun compound (see **Figure 1**). For instance, let us consider the noun compound 'air temperature'. We might **measure** the temperature of the air. There is a verb component to understanding the action of 'air' on 'temperature'. We seek to capture this component by looking at windows of relatively large radius ($r = 14$). Inside this window, every time both words of a noun compound are in the window, we add all of the verbs in the window to the context

columns. The phrase regarding 'air temperature' described above would reasonably occur, and we would add the word 'measure' to the column space.

There were two resulting $\mathcal{A}$ spaces. Both $\mathcal{D}$ spaces had 99 rows (we draw from $\mathcal{NC}$ only).

The COCA $\mathcal{A}$ space had 15566 columns, and the GloWbE $\mathcal{A}$ space had 15777 columns.

We use the $\mathcal{A}$ space one time in our noun compound expression, to capture a portion of the notion of the action of the modifier on the head. We can fill out the notion further by turning to adjectives, which we do now.

2.4. Qualifier-Head Space $Q_{\mathcal{H}}$

Turney mentions towards the end of his paper the notion of a quad-space as a possible route for future exploration in the development of semantic vector spaces **[Turney2012]**. We perform that extension in this paper by turning to adjectives.

The idea of the qualifier-head space is to provide a frame of reference for the modifier word of the noun compound by looking at the head. The modifier sometimes acts like an adjective on the head word of the noun compound. A pertinent example is 'stone bowl'. 'stone' acts like an adjective on bowl (consider 'grey bowl'). Therefore, knowing only the head of the noun compound allows us to gain some knowledge regarding the kinds of adjectives that are used upon it. That is the fundamental idea behind the qualifier-head space. We construct it by considering a half-window of radius $r = 3$ for each individual head word in the noun compounds– that is, we look at the 3 words behind the head word, and search for the closest adjective. Whatever the closest adjective is gets added to the column space.

This space is able to model the modifier to a certain extent (by the substitute words that are adjectives). However, it also contributes to the notion of the action. Verbs are not the only way we can represent the action of the modifier on the head. In fact, treating the modifier like an adjective is a more direct representation of action – a 'stone bowl' is **made of** stone: stone is the **material** (our qualifier) of which the bowl is made, just as 'grey' is the **color** of a 'grey bowl'. Thus, $Q_{\mathcal{H}}$ space can also be said to contribute to the representation of the action.

We use the $Q_{\mathcal{H}}$ space once in our representation of the noun compound: for a given noun compound, we include the $Q_{\mathcal{H}}$representation of the head word in our 5-tuple.

There were two resulting $Q_{\mathcal{H}}$ spaces. Both $Q_{\mathcal{H}}$ spaces had 106 rows (we draw from $\mathcal{W}$ only).

The COCA $Q_{\mathcal{H}}$ space had 68216 columns, and the GloWbE $Q_{\mathcal{H}}$ space had 59168 columns.

2.5. Qualifier-Modifier Space $Q_{\mathcal{M}}$

We add a second adjective-based space. The qualifier-modifier space is intended to act as a sort of domain-esque model for the modifier word, but instead of being based on nouns, based on adjectives. As we have established, the modifier can sometimes behave like an

adjective to the head word. Therefore, we can compare it to a bunch of adjectives that are nearby to develop a model of the kinds of adjectives the modifier is similar to. For instance, consider the phrase 'beautiful, luxurious marble bowl upon which were etched gorgeous scenes'. The $Q_{\mathcal{M}}$ model would then capture the fact that modifier noun 'marble' appears to be associated with the adjectives 'beautiful', 'luxurious', and 'gorgeous'. These are qualities of the descriptor. To build this space, we consider windows with size $r = 10$ around each modifier word, and add all adjectives in the range to the column space.

We use the $Q_{\mathcal{M}}$ space once in our representation of the noun compound: for a given noun compound, we include the $Q_{\mathcal{M}}$ representation of the modifier word in our 5-tuple.

There were two resulting $Q_{\mathcal{M}}$ spaces. Both $Q_{\mathcal{M}}$ spaces had 106 rows (we draw from $\mathcal{W}$ only).

The COCA $Q_{\mathcal{M}}$ space had 194522 columns, and the GloWbE $Q_{\mathcal{M}}$ space had 114466 columns.

2.6. Processing the Word-Context Matrices

After deriving the word-context matrix for a particular space, each matrix undergoes two transformations. First we do a ppmi transformation of the matrix, and then we do rank-k SVD on the matrix. We choose k by picking the first k singular values that explain 90% of the square-sum of the singular values, which is a standard choice in dimension reduction. We describe the theory behind and benefits of each of these processes in **Appendix A: Word-Context Matrix Processing Steps**.

We provide a list of the resulting dimensions of each space here. The number of singular values kept after dimension reduction is the number of columns in the vector representation.

COCA $\mathcal{D}$: We kept 105 singular values.

GloWbE $\mathcal{D}$: We kept 113 singular values.

COCA $\mathcal{A}$: We kept 50 singular values.

GloWbE $\mathcal{A}$: We kept 48 singular values.

COCA $Q_{\mathcal{H}}$: We kept 78 singular values.

GloWbE $Q_{\mathcal{H}}$: We kept 80 singular values.

COCA $Q_{\mathcal{M}}$: We kept 79 singular values.

GloWbE $Q_{\mathcal{M}}$: We kept 83 singular values.

Note that the GloWbE spaces often ended up having similar dimensions to the COCA space (and sometimes, the original word-context matrix even had fewer columns!), despite the approximate 4 times increase in size, which suggests that the patterns used to generate the columns may have been sparser in the GloWbE corpus.

2.7. Similarity Structure Functions

Now we have defined our four spaces $\mathcal{D}, \mathcal{A}, Q_{\mathcal{H}}, Q_{\mathcal{M}}$. Recall from before we represent a noun compound with a 5-tuple.

Let us call a given noun compound as the tuple (a, b), where a is the modifier word and b is the head word. Then, our representation of the noun compound is given by

$$\textbf{rep}((a,b)) = \{\mathcal{D}[a,b], \mathcal{D}[b], \mathcal{A}[a,b], \mathrm{Q}_{\mathcal{H}}[b], \mathrm{Q}_{\mathcal{M}}[a]\} \tag{1}$$

For future reference, let $\textbf{proj}(R, i)$ be the projection function that takes in a representation R and an index i and returns the i^{th} term of the representation. Note that $i \in [5]$.

Importantly, this representation of the noun compound is asymmetric: That is, $\textbf{rep}((a,b)) \neq \textbf{rep}((b,a))$ due to the usage of $\mathrm{Q}_{\mathcal{H}}$ and $\mathrm{Q}_{\mathcal{M}}$. Asymmetry is desirable since it is rarely ever true that inverting the head and modifier of a noun compound create a similar noun compound (or even a sensical one). For example, consider 'death wish' and its inversion 'wish death'. A 'death wish' is a 'wish to die', and while 'wish death' is probably much less frequent, could mean something along the lines of 'the death of a wish', which is a different meaning entirely.

Now our goal is to find a function $F : \mathcal{NC} \times \mathcal{NC} \to \mathbb{R}$ that evaluates the similarity of two noun compounds. We will generally refer to the first noun compound in the input to F as (a, b) and the second noun compound in the input to F as (c, d), where a, c are modifiers and b, d are heads.

At this point, there are two approaches we could take: the vector composition approach (taken in **[Fyshe2013]**) or the similarity composition approach (taken in **[Turney2012]**). We choose to take the similarity composition approach. What we mean by this approach is that effectively, F is transformed into a function of type $f : \mathbb{R}^k \to \mathbb{R}$. The input to f is k real values. The k real values are derived by computing distances from components in $\textbf{rep}((a,b))$ to components in $\textbf{rep}((c,d))$. We make this explicit in the following equation:

$$F((a,b),(c,d)) = f(l_1, ..., l_k) \tag{2}$$

where l_i, $i \in [k]$ are the distance, or length values. Two questions now arise: First, how do we compute length? Second, which lengths do we choose as the input to f?

The answer to the first question is cosine distance. It is the most widely used measure of distance in the literature because it is agnostic to magnitude **[Turney2010]**. We are only interested in vectors that are nearby each other angle-wise, or clusters. We make a slight modification of the standard definition: typically, the range is $[-1, 1]$. We modify the function so that if the output is ≤ 0, the output is 0, thus changing the range to $[0, 1]$. We implement this cutoff so that we never have negative terms in our distances. This feature is desirable because negative similarities do not really make sense: either two noun compounds are similar, or they are not. Similar and dissimilar things do not cancel each other out algebraically the way opposites do. In fact, opposites may be considered similar in the geometric clustering! Therefore, we consider all negative similarity values to mean dissimilar, and set them to 0. The distance function is therefore defined as

$$\textbf{dist}(\mathbf{v}_1, \mathbf{v}_2) = \begin{cases} \cos(\mathbf{v}_1, \mathbf{v}_2) = \dfrac{\mathbf{v}_1}{||\mathbf{v}_1||} \cdot \dfrac{\mathbf{v}_2}{||\mathbf{v}_2||} & \text{if } \cos(\mathbf{v}_1, \mathbf{v}_2) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

We will use this measure throughout the rest of the paper: Every time we refer to a distance or length between components of noun compound representations, we will mean the cosine distance. Therefore, note that all lengths and distances are non-negative and in the range $[0, 1]$.

The second question is a bit trickier, and is subject to more creativity. We want to only use the l_i that capture meaningful comparisons between the two noun compound representations. In some cases that might involve cross-terms, as in **[Turney2012]**. However in this paper, we use no cross-terms and only do component-to-component distances. That is, we have precisely 5 distances.

$$l_i = \textbf{dist}(\textbf{proj}(\textbf{rep}(a, b), i), \textbf{proj}(\textbf{rep}(c, d), i)) \text{ for all } i \in [5] \tag{4}$$

Now we know the inputs to f, we can define it. Henceforth f shall be known as the similarity structure function, for it tells us how to put together the component similarities to derive the total similarity between two noun compounds.

It is desirable that the structure function follow some rules. We draw from Turney's rules for his structure functions, and add some modifications **[Turney2012]**.

The rules that f must satisfy are given by

$$f((a, b), (c, d)) \neq f((b, a), (d, c)) \tag{5}$$

$$f((a, b), (c, d)) = f((c, d), (a, b)) \tag{6}$$

$$f((a, b), (c, d)) \neq f((a, b), (d, c)) \tag{7}$$

$$f((a, b), (c, d)) \neq f((a, d), (c, b)) \tag{8}$$

For the most part, these are identical to those of Turney's with the exception of the first equation. Turney's paper does not focus on noun compounds when dealing with pairs of words. As we saw before, it is not in general true that inverting a noun compound produces a noun compound with a similar set of relations. Therefore, we are more strict. We have already accounted for **Equation 5** when we added our two qualifier spaces: they ensure that the order of terms of the noun compound matter. Similarly, the qualifier spaces solves **Equation 7**. **Equation 8** simply says that we cannot simply swap head words – this rule is inherent in the whole representation of the noun compounds. We must therefore be only careful to follow **Equation 6**, which leaves the noun compounds alone and focuses on the structure of the function itself.

2.8. Similarity Composition Functions

Before we finally describe f, we introduce the notion of similarity composition functions. A similarity composition function g takes as input n lengths $l_1, ..., l_n$ and outputs a unified similarity $l_{composed}$. At first glance, this definition is identical to the previously defined f. We make the distinction as follows: Similarity composition functions will be used as building block tools *inside of* f. In other words, in the structure function, multiple complicated functions are allowed to be used, while for composition functions, we will identify only one function. This setup allows for ease of expression when we describe a structure function in terms of block-like composition functions.

There are two composition functions we use to form our overall similarity function. The first we borrow from **[Turney2012]**: we call it the truncated geometric mean (TGM). We have

$$\mathbf{tgm}(l_1, ..., l_n) = \begin{cases} \left(\prod_{i=1}^{n} l_i\right)^{\frac{1}{n}} & \text{if } \mathrm{l}_i > 0 \text{ for each } i \in [n] \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The idea behind this function is that the combined similarity should be high when component similarities are high, and low when component similarities are low. **[Turney2012]**.

Turney notes that there is no set-in-stone reason that this should be the only similarity composition function used, so we also define our own composition function, called the weighted sigmoid arithmetic mean (WSAM). It is defined in two parts. First, we take the arithmetic weighted mean of the input lengths (let the vector with values l_i be defined as $\mathbf{l}$), with respect to some weight vector $\mathbf{w}$. The values of the weight vector must be all non-negative and must sum to 1. Then, we feed the mean to a sigmoid function with a range $[0, 1]$. The idea is we take the mean to get a different estimate of the average from the geometric mean, and then using some guesses regarding the distribution (i.e., we would think that more noun compound pairs are not similar than similar), set a threshold after which the similarity score will spike and flatten to 1 asympotically. A sigmoid function also encapsulates a lower threshold, below which point the similarity score flattens to 0.

$$\mathbf{wsam}(\mathbf{l}, \mathbf{w}) = \frac{a}{a + c * e^{-b(\mathbf{l} \cdot \mathbf{w})}} \tag{10}$$

for some parameters a, b, c.

The specific sigmoid function was given by the equation the parameter choices $a = 0.00007, b = 20, c = 4$, which was found after playing around with parameters. The idea was that we wanted the threshold for having high similarity to be larger than 0.6. The function is plotted in **Figure 2**.

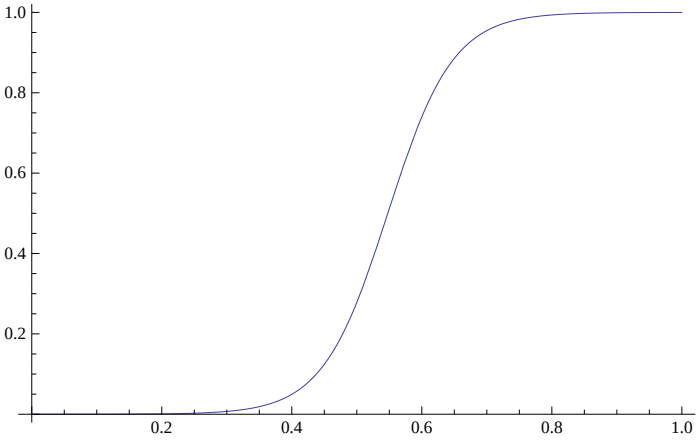


Figure 2: *Plot of the Sigmoid Function*

2.9. Putting Structure and Composition Together

In this section, we describe the similarity functions we test.

We will call the first similarity function **tgm-vanilla**. In this function, we simply return tgm applied to all of the 5 lengths. It is titled vanilla because of how simple it is. Recall that $l_1, ..., l_5$ are the component-wise distances defined for each of the five components that make up the representation of a noun compound.

$$\textbf{tgm-vanilla}((a,b),(c,d)) = \textbf{tgm-vanilla}(l_1, ..., l_5) = \textbf{tgm}(l_1, ..., l_5) \tag{11}$$

The second similarity function we call **weighted-sim**. Based on our understanding of each of the components, we assigned a weight vector for the noun compound structure: $\mathbf{w}_{nc} = \{0.05, 0.2, 0.35, 0.25, 0.15\}$. That is to say, we gave the least importance to the domain of the whole noun compound with a weight of 0.05, assigned a significant amount of weight to components that influence the action of the noun compound (0.35 for $\mathcal{A}$, 0.25 for $Q_{\mathcal{H}}$), and then some weight to domain space for the head 0.2 and a little less weight (0.15) to $Q_{\mathcal{M}}$. The idea was to weight the action the most, and the individual components a fair amount. We assigned the least weight to the domain of the whole structure because the idea is that noun compound similarity depends a lot on the interaction between modifier and head. Thus we simply have

$$\textbf{weighted-sim}((a,b),(c,d)) = \textbf{weighted-sim}(l_1, ..., l_5) = \textbf{wsam}(\{l_1, l_2, l_3, l_4, l_5\}, \mathbf{w}_{nc}) \tag{12}$$

We now introduce the term *linking structure*. A linking structure is a diagram of a comparison of two objects with the same components. Double-sided arrows are drawn between components that are compared. We also label these arrows with the space in which the comparison is happening. $\mathcal{X}\uparrow$ indicates a relation in the space $\mathcal{X}$ where high similarity is desirable.

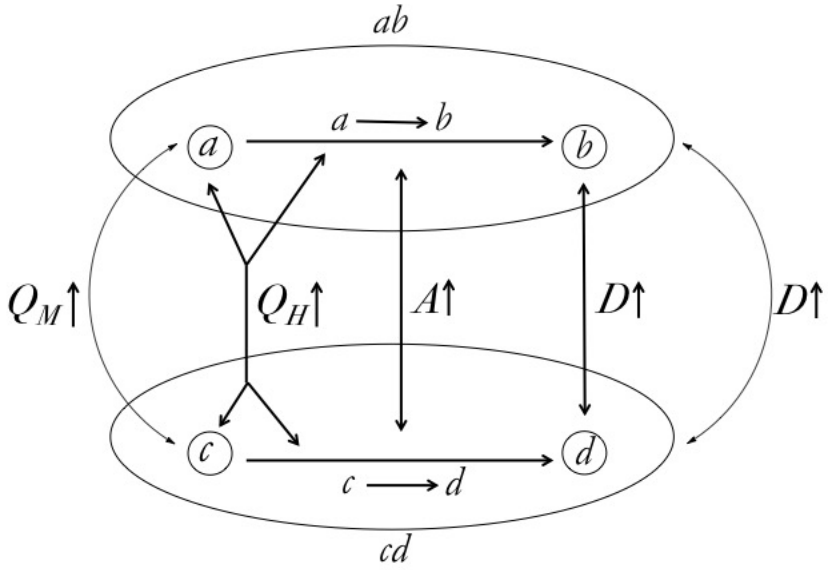


Figure 3: *Linking Structure*

We describe a linking structure when comparing a pair of noun compounds in **Figure 3**.

The last similarity function we test we will call **component-sim**. The idea in this function is to follow the linking structure to build component similarities, which we then combine with some composition function. First we combine $\mathcal{A}$ and $\mathrm{Q}_{\mathcal{H}}$ with **tgm** to form a function block. Then we combine $\mathrm{Q}_{\mathcal{H}}$ and $\mathrm{Q}_{\mathcal{M}}$ with **tgm** to form a modifier block. Then we build a noun compound block by first combining the head representation in $\mathcal{D}$ with the modifier block, and then combining that block with the noun compound domain representation to form a total domain block. We return finally a weighted average of the function block and the total domain block, with weights 0.8 and 0.2 respectively. Note that we re-use components: this is ok! The approach we follow here demonstrates the power of using similarity function composition to build a realistic idea of what is going on. The function is given by

$$\begin{aligned}\textbf{component-sim}((a,b),(c,d)) &= \textbf{component-sim}(l_1,...,l_5)\\ &= 0.8\textbf{tgm}(l_3,l_4)+0.2\textbf{tgm}(d_1,\textbf{tgm}(d_2,\textbf{tgm}(d_4,d_5)))\end{aligned} \tag{13}$$

3. Model Evaluation

To evaluate our semantic space models, we require three things: First, a set of noun compound pairs to evaluate the similarities of. Second, a groundtruth dataset of human-decided similarities between these noun compound pairs. Finally, another set of models from a previous paper so that we can compare our approach to those of other researchers

on the same dataset.

We collect our groundtruth dataset from Amazon Mechanical Turk, a website set up to provide human-survey data. The process that occurs is as follows: Survey tasks are provided to a set of workers, known as Turkers. In this case, the survey tasks are pairs of noun compounds, with a request for a classification (similar, not similar, very similar, or unknown). Each Turker thinks about each task (with his/her own human brain). Amazon gets paid a fee for providing the service, and Turkers are paid for each task they complete at a fixed rate.

Using Mechanical Turk to collect this sort of human-survey data has been compared to in-person approaches to gathering survey data, and been found to measure up **[Schnoebelen2010]**.

We get a comparison vector space model from **[Fyshe2013]**, who provided theirs on the internet for download. The Fyshe VSM contains vectors of single words, so composition functions were needed to generate a Fyshe representation for noun compounds. We followed the instructions of their paper to create the vector compositions used.

We then process the Mechanical Turk data to come up with a human-approved similarity rating for each noun compound pair. We simply iterate over all the noun compound pairs and compare each model's similarity score to the human similarity score, and assess each model based on the percent correct (where correct means 'matches the human score').

3.1. Set of Noun Compound Pairs

First we describe how we chose our noun compounds. We picked 99 distinct noun compounds since the number of noun compound pairs resulting is $\binom{99}{2} = 4851$, a number that was within budget to acquire Mechanical Turk answers for. The specific noun compounds used can be viewed in the **List of Noun Compounds Used**.

The way we chose these 99 noun compounds was based on the words in COCA. We looked at the set of noun compounds with frequencies strictly larger than 100 in COCA, and then sorted them into groups by head and by modifier. We then looked at the heads and modifiers with the highest frequencies, and chose from the batch of the top 50 largest head frequencies, and the top 50 largest modifier frequencies. For each head, we chose around $5-10$ noun compounds based on the number of noun compounds there were for a given head. We filled in the first 50 of 100 noun compounds in this way. Likewise, for the latter 50, we chose $5-10$ noun compounds depending on the number of noun compounds for a given modifier. The noun compounds we chose from each head/modifier were chosen with the intent of having some similar relationships, and some different relationships (across each head/modifier). We also sought to avoid overly generic nouns (like 'system' and 'use'). We then verified that these noun compounds were present in GloWbE and the Fyshe VSM.

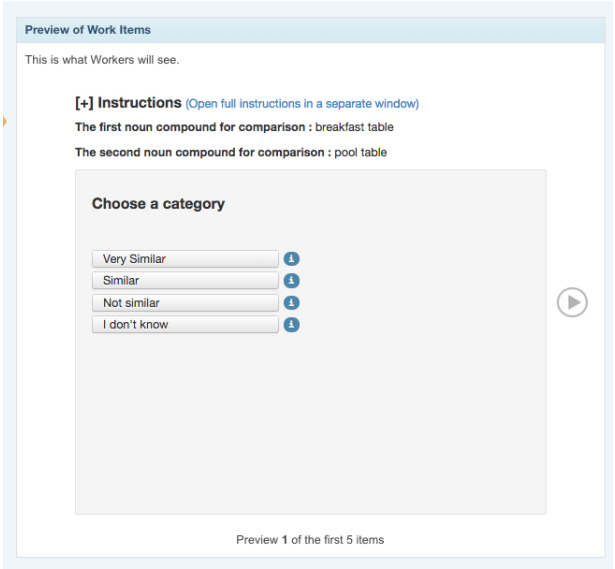


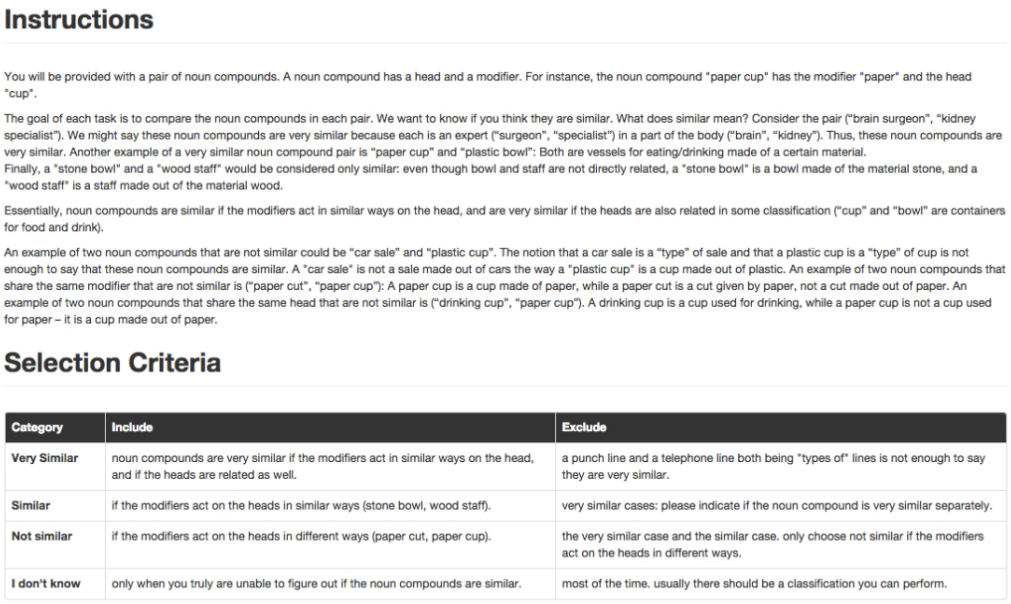
Figure 4: *Sample Mechanical Turk HIT*

3.2. Building the Mechanical Turk Dataset

In order to provide the Turkers with tasks, we had to first define each task. Each task, or HIT in Mechanical Turk lingo, was a single noun compound pair with a single question, "Is the noun compound pair similar, not similar, very similar, or unknown?", presented in multiple choice format. The nature of Mechanical Turk is that it does not allow you to cleanly specify on one page more than one multiple-choice question. In the interest of making the HITs very clear to the Turkers, we left each HIT as containing a single noun compound comparison task. A sample HIT is presented in **Figure 4**.

We also had to provide the Turkers instructions on how to complete each HIT, which primarily constitutes defining what we mean by similarity between noun compounds. We do not want to give a definition of similarity that permits the Turkers to use an algorithm to find if two noun compounds are similar, for that defeats the purpose of using similarity as a measurement of the ability of the computational model to capture a sense of meaning. We want to know what their gut feeling regarding similarity is as well, and we want to avoid influencing them too much. However, it is no good if every Turker decides their own meaning of the notion of similarity between noun compounds. We need some common ground, so we provide a holistic description of what two noun compounds being similar means. We provided the instructions shown in **Figure 5** to the Turkers.

The inherent difficulty in leaving some leeway in what is meant by similar is that we cannot flat out reject the Turkers' responses so easily without knowing their justification. It is also hard to tell then whether or not the Turkers took their job seriously. To mitigate these concerns, we employed master Turkers (which must be earned by consistently getting $> 99\%$ of their work approved) for a slight extra cost. In order to ensure we got different people from perhaps different parts of the world, and not a homogenous Turker



Instructions

You will be provided with a pair of noun compounds. A noun compound has a head and a modifier. For instance, the noun compound "paper cup" has the modifier "paper" and the head "cup".

The goal of each task is to compare the noun compounds in each pair. We want to know if you think they are similar. What does similar mean? Consider the pair ("brain surgeon", "kidney specialist"). We might say these noun compounds are very similar because each is an expert ("surgeon", "specialist") in a part of the body ("brain", "kidney"). Thus, these noun compounds are very similar. Another example of a very similar noun compound pair is "paper cup" and "plastic bowl": Both are vessels for eating/drinking made of a certain material.
Finally, a "stone bowl" and a "wood staff" would be considered only similar: even though bowl and staff are not directly related, a "stone bowl" is a bowl made of the material stone, and a "wood staff" is a staff made out of the material wood.

Essentially, noun compounds are similar if the modifiers act in similar ways on the head, and are very similar if the heads are also related in some classification ("cup" and "bowl" are containers for food and drink).

An example of two noun compounds that are not similar could be "car sale" and "plastic cup". The notion that a car sale is a "type" of sale and that a plastic cup is a "type" of cup is not enough to say that these noun compounds are similar. A "car sale" is not a sale made out of cars the way a "plastic cup" is a cup made out of plastic. An example of two noun compounds that share the same modifier that are not similar is ("paper cut", "paper cup"): A paper cup is a cup made of paper, while a paper cut is a cut given by paper, not a cut made out of paper. An example of two noun compounds that share the same head that are not similar is ("drinking cup", "paper cup"). A drinking cup is a cup used for drinking, while a paper cup is not a cup used for paper – it is a cup made out of paper.

Selection Criteria

Category	Include	Exclude
Very Similar	noun compounds are very similar if the modifiers act in similar ways on the head, and if the heads are related as well.	a punch line and a telephone line both being "types of" lines is not enough to say they are very similar.
Similar	if the modifiers act on the heads in similar ways (stone bowl, wood staff).	very similar cases: please indicate if the noun compound is very similar separately.
Not similar	if the modifiers act on the heads in different ways (paper cut, paper cup).	the very similar case and the similar case. only choose not similar if the modifiers act on the heads in different ways.
I don't know	only when you truly are unable to figure out if the noun compounds are similar.	most of the time. usually there should be a classification you can perform.

Figure 5: *Instructions provided to the Turkers*

population, we ran the Mechanical Turk experiments at different times of the day. We also performed sanity checks to ensure that the majority of the responses were not answered 'unknown' as an easy way to make money. We also checked the average time to complete each HIT to ensure we were not getting ridiculously quick clickers. If the Turkers spent more than a few seconds on each HIT, we deemed the HITs acceptable since spending ten seconds or more at least suggests that thought was put into completing each HIT.

We wanted to ensure that for each noun compound pair, at least three distinct Turkers had looked at it and assigned a similarity rating. Thus we needed to upload to Mechanical Turk the same set of HITs at least three times. The average times to complete a HIT were 46 seconds for the first time we ran the dataset on Mechanical Turk, 21 seconds for the second time, and 22 seconds for the third time. We also had to block workers in order to get the minimum of three different Turkers to review the same HIT. Some Turkers who responded to the same HIT more than once sometimes responded with different replies (i.e. the first time the Turker said a certain pair of noun compounds was similar, and the second time the same Turker said the pair was not similar!). We called these Turkers 'bad Turkers' and needed to replace all of their work.

For each of the bad Turkers, we did repeat runs in order to replace their work. The first time we did this, the average time to complete each question was 92 seconds, which means the Turkers spent more time on each question. The second time, it was 39 seconds. The third time, it was 13 seconds. If a bad Turker answered a question only twice, we ran the noun compound pair on Turk two more times. If a bad Turker answered a question three times, we ran the noun compound three more times (to completely replace the Turker's work). There was only one Turker who answered multiple questions three times,

and they were also a bad Turker. For the Turkers who answered the same question twice that were not bad Turkers, we kept their answer and just ran the question another single more time to get a new response so that we would have three distinct responses for each noun compound pair. We also classified 268 noun compound pairs ourselves to get a feel for the task and to fill in gaps so that we would have a minimum of three different human ratings for each noun compound.

However, we retain some confidence in the accuracy of the Mechanical Turk results despite some of the mishaps. When we blocked the Turkers, some of the Turkers emailed asking for reasons why their work was rejected, and provided their detailed reasoning for choosing particular answers. So it appears that at the very least a few of them were very earnest and attempting to do a good job.

After we received at least three classifications from distinct humans, we needed to determine what the final classification was. The number of distinct Turkers who answered per noun compound was between 3 and 6 inclusive. There were only four possible classifications. If all classifications were the same, then we did not have to do any extra work. Otherwise, we looked at the classification that the largest percent of the Turkers answered with. If this value was larger than 50%, then that was the classification used. If not, then if any of the Turkers had answered that the noun compounds were very similar, this distinct was removed and very similar was treated as similar. We recalculated the classification that the largest percent of Turkers answered with. If it was still less than 50%, then we said that we did not know what the human classification of the noun compound was.

3.3. Describing the Mechanical Turk Dataset

We provide the statistics of the Mechanical Turk datset below:

```
Mechanical Turk Test Set Statistics:
How many noun compound pairs for each class?
Not similar = 4380
Unknown = 90
Similar = 376
Very similar = 56

% Not Similar: 89.3%
% Unknown: 1.8%
% Similar: 7.7%
% Very Similar: 1.1%
```

Listing 1: *Mechanical Turk Dataset Statistics*

3.4. Building the Fyshe Vector Space

We downloaded Fyshe's VSM from the website specified in **[Fyshe2013]**. The VSM came in a Matlab file, in which were encoded a matrix with 55000 rows and 2000 columns.

The first 1000 columns specify Document space, and the second 1000 columns specify Dependency space. Document space is created by finding word counts in individual documents drawn from the 50 million document ClueWeb09 data set, and Dependency space represents each word with a vector where each column is a context in which the word appears (for instance, for word w, the word w is the subject of the verb 'eat') **[Fyshe2013]**. The rows are nouns and adjective noun phrases.

One task Fyshe et. al addressed was using the VSM to decide if two adjective-noun phrases were similar, compared against a groundtruth dataset of human assessments, which is very similar to the noun compound similarity task (the only differences are noun compound versus adjective-noun, and the fact that they used a Likert scale of $1-7$ instead of the $-1, 0, 1, 2$ scale we used).

Their best results on this task used vectors built from the first 25 Document columns concatenated with the first 600 Dependency columns to represent individual words. To assess phrase similarity, Fyshe et. al used vector composition instead of similarity composition. Fyshe found that the two best approaches for vector composition were quite simple: a vector sum of the phrase components, and a dilated vector sum of the phrase components. The dilated vector sum is just a vector sum with the adjective vector multiplied by a factor of $\gamma = 16.7$, derived based from previous work.

We implemented these two approaches for comparison purposes.

3.5. Testing Procedure

To evaluate our semantic vector models, we calculated a model similarity score and the Turk classification for each noun compound pair. The model similarity score was in the range $[0, 1]$. If the score was ≤ 0.4, the model classification was designated 'not similar'. If the score was < 0.4 and ≤ 0.6, the model classification was designated 'unknown'. If the score was > 0.6 and ≤ 0.9999, the classification was 'similar'. Finally, if the score was > 0.9999, the classification was 'very similar'. This translation from similarity score to classification was based on the principle that there should be a small window around 0.5 which should be considered too unclear to tell. We made that window symmetric and with radius 0.1. Therefore, for each model and for each noun compound pair, we had a model classification and a Turk classification. If either of the classifications was 'unknown', then we skipped that noun compound pair. Otherwise, we checked if the two classifications were the same. If they were the same, we recorded the model as getting the correct response on that noun compound pair. Otherwise, we recorded the model as getting the incorrect response on that noun compound pair. Then we sorted the models by the percentage of noun compound pairs they solved correctly.

Since there are large number of 'not-similar' ratings, we also examined the how the models perform over the subset of the Turk dataset for which Turkers rated the noun compound pairs as 'similar'. We performed exactly the same steps as before, just with a reduced list of noun compound pairs.

Finally, we examined the errors each model made and analyzed them manually.

4. Results

We discuss the general meanings of the rankings of the vector spaces for the Fyshe, COCA, and GloWbE models.

4.1. Results

Here we have the results for all vector space models.

```
Mechanical Turk Test Results

Ranked by % correct on all noun compound pairs:

---------------------------------
Fyshe Dilation Model:
# Correct: 4252
# Incorrect: 396
# Skipped: 254
---------------------------------
Fyshe Sum Model:
# Correct: 4313
# Incorrect: 389
# Skipped: 200
---------------------------------
COCA Weighted Model:
# Correct: 4384
# Incorrect: 385
# Skipped: 133
---------------------------------
COCA Vanilla Model:
# Correct: 4369
# Incorrect: 385
# Skipped: 148
---------------------------------
GloWbE Weighted Model:
# Correct: 4378
# Incorrect: 389
# Skipped: 135
---------------------------------
GloWbE Component Model:
# Correct: 4358
# Incorrect: 385
# Skipped: 159
---------------------------------
COCA Component Model:
# Correct: 4360
# Incorrect: 384
# Skipped: 158
---------------------------------
GloWbE Vanilla Model:
# Correct: 4350
```

```
# Incorrect: 381
# Skipped: 171
####################################
Summary (sorted by % correct):
----------------------------------
COCA Weighted Model: 89.4328845369%

GloWbE Weighted Model: 89.3104855161%

COCA Vanilla Model: 89.1268869849%

COCA Component Model: 88.9432884537%

GloWbE Component Model: 88.9024887801%

GloWbE Vanilla Model: 88.7392900857%

Fyshe Sum Model: 87.984496124%

Fyshe Dilation Model: 86.7401060792%

Ranked by % correct on noun compound pairs that Mechanical Turk labeled
    'Similar':

----------------------------------
Fyshe Dilation Model:
# Correct: 81
# Incorrect: 337
# Skipped: 14
----------------------------------
Fyshe Sum Model:
# Correct: 125
# Incorrect: 262
# Skipped: 45
----------------------------------
COCA Weighted Model:
# Correct: 85
# Incorrect: 317
# Skipped: 30
----------------------------------
COCA Vanilla Model:
# Correct: 9
# Incorrect: 382
# Skipped: 41
----------------------------------
GloWbE Weighted Model:
# Correct: 82
# Incorrect: 320
# Skipped: 30
----------------------------------
GloWbE Component Model:
# Correct: 52
```

```
# Incorrect: 337
# Skipped: 43
---------------------------------
COCA Component Model:
# Correct: 52
# Incorrect: 337
# Skipped: 43
---------------------------------
GloWbE Vanilla Model:
# Correct: 5
# Incorrect: 377
# Skipped: 50
#####################################
Summary (sorted by % correct):
---------------------------------
Fyshe Sum Model: 28.9351851852%

COCA Weighted Model: 19.6759259259%

GloWbE Weighted Model: 18.9814814815%

Fyshe Dilation Model: 18.75%

GloWbE Component Model: 12.037037037%

COCA Component Model: 12.037037037%

COCA Vanilla Model: 2.08333333333%

GloWbE Vanilla Model: 1.15740740741%
```

Listing 2: *Performance of the VSM Models*

We have two sets of results: results over all the noun compound pairs, and results over only the similar-Turk-ranked noun compound pairs. We denote the former as the comprehensive test set, and the latter as the similar-only testset.

We first note that there is very small variation in the scores in the comprehensive set ($86.7\% - 89.4\%$) compared to the similar-only testset ($1.1\% - 28.9\%$). A possible reason for this result is the large number of noun compounds ranked not-similar. The models all seem to be biased towards ranking not-similar, and thus do well at that job, but not quite as well on the similar-only task.

All our models performed better than the Fyshe VSM models did when we consider the whole noun compound pair testset, with the winner being the COCA Weighted model, followed close behind by the GloWbE Weighted model. The COCA Vanilla model came close behind, followed by the Component Models, with GloWbE Vanilla model performing the worst of our models. However, again, this result may only be an indication of a stronger bias in our models towards rating 'not similar', especially since the model that used the least amount of the structure developed got the best score.

It is also worth noting that the Vanilla Models are more biased towards giving a 'not similar'response, since the way they are constructed requires only one component of the model to have a score of 0 to send the whole similarity score to 0. Since there is no weighting, a very high score in one component may not outperform middle-low scores in all the other components, even if the component that has the high score is an action-space component. It is also interested to note that the COCA versions outperformed their GloWbE counterparts in all cases. The reason for this remains to be seen. However, the fact remains that the performances were all very similar. What we can conclude is that COCA seems to perform at least equally as well as GloWbE on this dataset for the similarity functions used.

Thus we turn our attention to the similar-only testset. The choice of corpus did not seem to affect the performance of our models very much for each choice of type of model. The Fyshe Sum model performed the best on this portion of the corpus with a score of 28.9%. This model was the better of the two in the original Fyshe paper as well. The COCA Weighted and GloWbE Weighted performed the best of our models and succeeded at beating the Fyshe Dilated Sum model, weighing in at 19.6% and 19.0% respectively. The Fyshe Dilated Sum model was close behind at 18.8%. The COCA and GloWbE Component models are next with scores of 12.0% and 12.0% respectively. Finally at the very end, the COCA Vanilla and GloWbE Vanilla models had scores of 2.1% and 1.1% respectively. In the similar-only task, the COCA and GloWbE models for each type of model (Vanilla, Component, Weighted) perform very similarly, with COCA being only slightly better than GloWbE on all models. This result is not surprising due to the fact that COCA and GloWbE had similar vector spaces dimension sizes, though of course the columns may have had different contexts. Since GloWbE was such a large corpus however, it is likely that a fair portion of the column contexts were the same or very similar, especially since the SVD step of dimension reduction culled a lot of the low-frequency contexts. This result suggests that more creative uses of the columnspace need to be made in order to gain more from size-increases in the corpus used.

In the similar-only task, it becomes evident that the approach used for the similarity structure function is crucial in determining whether or not a noun compound pair is similar. The weighted and component models significantly outperformed the vanilla model, suggesting that the machinery built into those two models is doing some work to distinguish similar from not-similar noun compound pairs. It appears that the type of model used, not the corpus is the significant feature in the models we constructed.

5. Future Work

This project has a lot of extensions that could be pursued as future work.

5.1. Improving the Model: Parameter Learning

One thing we could attempt to improve the model is the implementation of parameter tuning. There are multiple parameters we could have fiddled with but did not. The first

is the p parameter used in [**Turney2012**]. The singular value matrix resultant from the SVD is raised to the p^{th} power, and grid search is performed to optimize the value to get the best row vectors for comparison. In this paper we simply set $p = 1$. In that paper k, the number of singular values, is chosen in a similar manner. In this paper we chose the popular adhoc rule of retaining singular values until we have explained at least 90% of the square-sum of singular values.

For the Weighted Model, we could also attempt to optimize the parameters of the sigmoid function, either by using a method along the lines of grid-search, or by potentially utilizing neural nets to learn the best function. We can also extend this approach to learning the weights in the weighted model. The weighted similarity structure function (see **Equation 12**) therefore generalizes itself well to unsupervised learning procedures, which could be an interesting approach to take to develop more similarity functions.

5.2. Improving the Mechanical Turk Test Set

We believe the test set we use could also be improved if there were more funding to test more noun compound pairs. We could also increase the number of distinct Turkers who provide classifications for each noun compound pair. Ideally, we could also get more feedback regarding the task description (see **Figure 5**). Finally, an interesting approach to human survey data collection is provided by Crowdflower, which provides quality control for these kinds of surveys.

5.3. More Experiments

There are also more experiments we can do with our VSMs. We list a few of them below:

1. Build and test vector composition models in addition to similarity composition models with our VSMs.

2. Explore the clustering properties of quad-space vectors.

3. Test our model on other established datasets: particularly, WordSim-353, a standard dataset that includes human similarity judgements on pairs of words would be interesting [**Huang2012**].

4. Construct more of our own different data sets to test: instead of from Mech Turk, use WordNet to generate noun pairs that would be considered similar [**Fellbaum1998**].

5. Compare the VSMs we generated to each other, and see if they rate words the same or differently. Possibly, also implement a majority model that does a majority vote across a few of the models that we created (for instance, pick the most common classification across the Vanilla Model, the Component Model, and the Weighted Model).

6. Acknowledgements

I would like to thank Professor Fellbaum for advising me throughout the past year and for reading the paper, Professor Dvir for being my second reader, the Princeton Math Department for giving me the opportunity to carry out this work as well as providing me

funding for it, and the Princeton Computer Science Department for obtaining access to the corpuses COCA and GloWbE, without which this paper would not have been possible. Lastly, I would like to thank Alice Tao for digitizing the **Linking Structure** diagram.

7. Appendix A: Word-Context Matrix Processing Steps

7.1. Pointwise Positive Mutual Information

ppmi is a smoothing technique applied to a word-context matrix. We apply the **pmi** function to each element of the word-context matrix. If the **pmi** value is ≤ 0, we set the value of the matrix at that location to 0.

We have the definition of the **pmi** function below:

$$\mathbf{pmi}_{ij} = \log\left(\frac{p_{ij}}{p_{i*}p_{*j}}\right) \tag{14}$$

Here, p_{ij} is the estimated probability that word w_i occurs in context c_j, p_{i*} is the probability of the word w_i, and p_{j*} is the probability of the context c_j **[Turney2012]**.

7.2. Singular Value Decomposition and Rank-k Approximation

One of the most important aspects of the algorithm to generate a word space is the dimension reduction step. In this paper, we use Singular Value Decomposition (SVD) to split the vector space matrix into 3 factors, U, D, V. U and V are orthogonal matrices, and D is diagonal with elements known as the singular values. We perform dimension reduction by retaining only k of the singular values and deleting the related columns of U, and then multiplying U by D to obtain the reduced vector space. SVD has the side-effect property of smoothing out the matrix even further. It does not, however, guarantee anything about the positivity or negativity of the values.

8. Appendix B: Describing the Corpora

The COCA corpus was split into 5 different files, and the GloWbE corpus was split into 23 different files. The corpora files used in this paper were of the Word-Lemma-Part of Speech format. Database and Raw Text versions were also provided, but not used in this paper. In the Word-Lemma-Pos format, the parts of speech were tagged by the CLAWS7 tagger using the **UCREL CLAWS7 tagset** tagset. For more information see the FAQ.

8.1. Corpus of Contemporary American English (COCA)

We calculated some noun compound-related statistics of the COCA corpus for general interest purposes **[COCA]**.

```
#=========COCA INFO============#
Some statistics on COCA corpus:
Number of distinct noun compounds: 10032
Number of distinct heads: 3046
Number of distinct modifiers: 2256
Total Number of times a noun compound shows up: 3015413
Approximate fraction of places in corpus that begin with a noun
    compound: 0.00685321137921
50 Most Common Head words and their frequencies:
   room: 42753
   care: 35657
   system: 34649
   members: 24866
   rate: 18315
   program: 17875
   industry: 17283
   programs: 16665
   group: 15743
   rates: 15119
   students: 14887
   education: 14253
   officials: 14153
   market: 14146
   companies: 13528
   groups: 13518
   director: 13409
   use: 13120
   company: 12700
   table: 12374
   door: 12326
   force: 12092
   team: 11863
   station: 11782
   process: 11668
   cancer: 11608
   level: 11158
   game: 10782
   line: 10595
   school: 10573
   control: 10293
   center: 10087
   systems: 9921
   store: 9891
   service: 9863
   plan: 9758
   prices: 9622
   insurance: 9418
   show: 9376
   tax: 9246
   officer: 9222
   reform: 9217
   life: 8801
```

```
   juice: 8497
   enforcement: 8385
   time: 8327
   point: 8236
   conference: 8217
   member: 8173
   services: 8098
50 Most Common Modifier words and their frequencies:
   school: 63347
   health: 62043
   family: 32908
   tax: 28271
   state: 26247
   drug: 24218
   business: 23265
   phone: 22376
   police: 21988
   college: 21958
   government: 21601
   home: 20760
   law: 20424
   community: 19525
   air: 17952
   news: 17866
   security: 17430
   research: 17145
   food: 17048
   heart: 16869
   water: 16504
   oil: 16383
   computer: 15391
   education: 15153
   world: 15028
   back: 15009
   blood: 14890
   care: 14654
   executive: 14457
   music: 14432
   student: 14122
   life: 13621
   living: 12973
   energy: 12958
   art: 12738
   power: 12709
   insurance: 11978
   death: 11834
   child: 11681
   television: 11564
   credit: 10902
   stock: 10897
   interest: 10811
   time: 10681
```

```
   job: 10661
   car: 10647
   trade: 10523
   body: 10496
   cell: 10264
   defense: 10259
```

Listing 3: *COCA Statistics*

8.2. Corpus of Global Web-Based English (GloWbE)

We calculated some noun compound-related statistics of the GloWbE corpus for general interest purposes [**GloWbE**].

```
#========GloWbE INFO=================#
Some statistics on GloWbE corpus:
Number of distinct noun compounds: 8880852
Number of distinct heads: 281268
Number of distinct modifiers: 255147
Total Number of times a noun compound shows up: 65280084
Approximate fraction of places in corpus that begin with a noun
    compound: 0.0343579389474
50 Most Common Head words and their frequencies:
   system: 411983
   state: 363720
   service: 336363
   program: 294855
   group: 251454
   time: 249432
   company: 249155
   rate: 245762
   center: 243544
   member: 234212
   people: 228247
   game: 195068
   york: 191445
   cent: 191020
   team: 186826
   day: 183696
   level: 183516
   area: 180968
   industry: 178364
   site: 176769
   city: 175018
   market: 174124
   zealand: 160964
   project: 158635
   change: 157836
   house: 156601
   management: 154091
```

```
   page: 153384
   card: 150690
   process: 149214
   university: 146909
   government: 145858
   price: 145564
   africa: 144942
   school: 143986
   policy: 141003
   room: 139926
   point: 139061
   line: 137187
   plan: 136943
   number: 128563
   party: 128059
   development: 127997
   post: 127858
   law: 127663
   lanka: 126548
   report: 125876
   issue: 125005
   act: 124153
   station: 122610
Most Common Modifier words and their frequencies:
   new: 437329
   world: 395479
   business: 371368
   us: 337535
   health: 335423
   time: 272285
   south: 258904
   mr: 257921
   government: 248396
   united: 245119
   state: 233444
   family: 225606
   john: 218736
   school: 212160
   million: 205978
   president: 197533
   service: 197281
   home: 197016
   per: 189919
   tax: 187231
   security: 183940
   food: 174182
   media: 171043
   web: 169550
   water: 166168
   development: 154512
   year: 153949
   research: 152993
```

```
city: 150873
day: 150104
north: 147233
sri: 147220
energy: 146720
life: 143689
police: 139986
community: 139526
power: 139454
david: 137920
market: 133621
news: 132406
internet: 131692
air: 127667
west: 127198
dr.-i: 126387
management: 124964
climate: 122367
party: 118667
credit: 117562
quality: 115548
de: 112622
```

Listing 4: *GloWbE Statistics*

9. Appendix C: Implementation Details and Future Usage

9.1. Libraries Used

All the code was written in Python, and Numpy was used significantly throughout the project. We also used the h5py module to build the Fyshe vector space from the provided Matlab file.

9.2. List of Noun Compounds Used

Here we provide the list of 99 noun compounds that were used for model creation and testing.

By accident, we included one noun compound ('life force') twice for a total of 100 noun compounds in the original list (repeated). Because we generated the noun compound pairs from this list, there were in fact $\binom{100}{2} = 4950$ noun compound pairs provided to Turkers, and as a result some of the noun compound pairs were solved more than once. We also had ('life force', 'life force') as a noun compound pair presented to Turkers as a result.

```
breakfast table
pool table
bargaining table
```

```
card table
oak table
assembly line
cruise line
party line
story line
telephone line
punch line
water line
tree line
trend line
goal line
air force
brute force
life force
task force
invasion force
grad students
art students
women students
law students
minority students
bus station
police station
space station
television station
train station
weather station
gas station
subway station
power station
research station
reading room
hotel room
computer room
dorm room
living room
wiggle room
chat room
dance company
finance company
parent company
adult life
animal life
sex life
love life
shelf life
water fountain
water availability
water glass
water heater
water hole
```

```
water lilies
water vapor
water temperature
water rights
water pressure
blood vessel
blood type
blood pressure
blood bank
blood test
blood flow
air temperature
air space
air service
air flow
air travel
air mattress
air fare
heart attack
heart monitor
heart disease
heart pounding
heart rate
death wish
death penalty
death sentence
death toll
death camp
death certificate
death squads
life events
life expectancy
life sciences
life savings
life span
life skills
life support
life style
life jacket
time zone
time interval
time period
time machine
time limit
```

Listing 5: *The 99 Noun Compounds*

9.3. Corpus-Parsing Algorithm

COCA and GloWbE are both very large data sets on the order of 0.5 billion and 2 billion words, so efficiency was important in guiding the algorithm design.

The main idea is to build a generic corpus tracer, which takes in some values (including a function), and at each step of the trace, executes the function with the right parameters.

The functions that were passed to the tracer were designed to calculate frequencies for specific contexts. There were four contexts (one each for $\mathcal{D}, \mathcal{A}, Q_{\mathcal{H}}, Q_{\mathcal{M}}$).

A window of fixed size is maintained as the tracer steps through the corpus in order to allow for exactly one corpus traversal, and in order to save space.

9.4. Potential as a Module

Other students at Princeton University may desire to use the COCA and GloWbE corpuses as well, now that they have been made available to all students. The code used in this project, after a little clean-up, should serve as a starting point for a Python module made available by the University alongside the corpora.

10. Citations

References

[COCA] Davies, M. (2008-). *The Corpus of Contemporary American English: 450 million words, 1990-present*. Available online at **http://corpus.byu.edu/coca/**.

[GloWbE] Davies, M. (2013-). *Corpus of Global Web-Based English: 1.9 billion words from speakers in 20 countries*. Available online at **http://corpus.byu.edu/glowbe/**.

[Fellbaum1998] Fellbaum, C. (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

[Fyshe2013] Fyshe, A., Talukdar, P., Murphy, B., and Mitchell, T. (2013). Documents and Dependencies: an Exploration of Vector Space Models for Semantic Composition. *CONLL*, pp. 84-93.

[Harris1954] Harris, Z. S. (1954). Distributional structure. *Word, 10*(23), pp. 146-162.

[Huang2012] Huang, E. H., Socher, R., Manning, C. D., and Ng, A. (2012). Improving word representations via global context and multiple word prototypes. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, pp. 873-882.

[Nakov2013] Nakov, P. I., and Hearst, M. A. (2013). Semantic interpretation of noun compounds using verbal and other paraphrases. *ACM Transactions on Speech and Language Processing, 10*(3), pp. 1-51.

[Sahlgren2006] Sahlgren, M. (2006). *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces* (Doctoral dissertation). Stockholm University, Department of Lingustics, Computational Linguistics, Stockholm, Sweden.

[Schnoebelen2010] Schnoebelen, T., and Kuperman, V. (2010). Using Amazon Mechanical Turk for linguistic research. *Psihologija, 43*(4), pp. 441-464.

[Turney2010] Turney, P. D. and Pantel, P. (2010). From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research, 37*, pp. 141-188.

[Turney2012] Turney, P. D. (2012). Domain and Function: A Dual-Space Model of Semantic Relations and Compositions. *Journal of Artificial Intelligence Research, 44*, pp. 533-585.

[VanDeCruys2013] Van de Cruys, T., Poibeau, T., and Korhonen, A. (2013). A Tensor-based Factorization Model of Semantic Compositionality. *Proceedings of NAACL-HLT, (June)*, pp. 1142-1151.