CHAPTER

15 Vector Semantics

"You shall know a word by the company it keeps!"
Firth (1957)

The asphalt that Los Angeles is famous for occurs mainly on its freeways. But in the middle of the city is another patch of asphalt, the La Brea tar pits, and this asphalt preserves millions of fossil bones from the last of the Ice Ages of the Pleistocene Epoch. One of these fossils is the *Smilodon*, or sabre-toothed tiger, instantly recognizable by its long canines. Five million years ago or so, a completely different sabre-tooth tiger called *Thylacosmilus* lived in Argentina and other parts of South America. Thylacosmilus was a marsupial whereas Smilodon was a placental mammal, but Thylacosmilus had the same long upper canines and, like Smilodon, had a protective bone flange on the lower jaw. The similarity of these two mammals is one of many examples of parallel or convergent evolution, in which particular contexts or environments lead to the evolution of very similar structures in different species (Gould, 1980).

The role of context is also important in the similarity of a less biological kind of organism: the word. Words that occur in *similar contexts* tend to have *similar meanings*. This insight was perhaps first formulated by Harris (1954) who pointed out that "oculist and eye-doctor . . . occur in almost the same environments" and more generally that "If A and B have almost identical environments. . . we say that they are synonyms." But the most famous statement of the principle comes a few years later from the linguist J. R. Firth (1957), who phrased it as "You shall know a word by the company it keeps!".

The meaning of a word is thus related to the distribution of words around it. Imagine you had never seen the word *tesgüino*, but I gave you the following 4 sentences (an example modified by Lin (1998) from (Nida, 1975, page 167)):

(15.1) A bottle of *tesgüino* is on the table.
Everybody likes *tesgüino*.
Tesgüino makes you drunk.
We make *tesgüino* out of corn.

You can figure out from these sentences that *tesgüino* means a fermented alcoholic drink like beer, made from corn. We can capture this same intuition automatically by just counting words in the context of *tesgüino*; we'll tend to see words like *bottle* and *drunk*. The fact that these words and other similar context words also occur around the word *beer* or *liquor* or *tequila* can help us discover the similarity between these words and *tesgüino*. We can even look at more sophisticated features of the context, syntactic features like 'occurs before *drunk*' or 'occurs after *bottle*' or 'is the direct object of *likes*'.

In this chapter we introduce such **distributional** methods, in which the meaning of a word is computed from the distribution of words around it. These words are generally represented as a *vector* or array of numbers related in some way to counts, and so these methods are often called *vector semantics*.

In this chapter we introduce a simple method in which the meaning of a word is simply defined by how often it occurs near other words. We will see that this method results in very long (technically 'high dimensional') vectors that are sparse, i.e. contain mostly zeros (since most words simply never occur in the context of others). In the following chapter we'll expand on this simple idea by introducing three ways of constructing short, *dense* vectors that have useful semantic properties.

The shared intuition of vector space models of semantics is to model a word by *embedding* it into a vector space. For this reason the representation of a word as a vector is often called an **embedding**. By contrast, in many traditional NLP applications, a word is represented as an index in a vocabulary list, or as a string of letters. (Consider the old philosophy joke:

embedding

> Q: What's the meaning of life?
> A: LIFE'

drawing on the philosophical tradition of representing concepts by words with small capital letters.) As we'll see, vector models of meaning offer a method of representing a word that is much more fine-grained than a simple atom like LIFE, and hence may help in drawing rich inferences about word meaning.

Vector models of meaning have been used in NLP for over 50 years. They are commonly used as features to represent words in applications from named entity extraction to parsing to semantic role labeling to relation extraction. Vector models are also the most common way to compute *semantic similarity*, the similarity between two words, two sentences, or two documents, an important tool in practical applications like question answering, summarization, or automatic essay grading.

15.1 Words and Vectors

Vector or distributional models of meaning are generally based on a **co-occurrence matrix**, a way of representing how often words co-occur. Let's begin by looking at one such co-occurrence matrix, a term-document matrix.

15.1.1 Vectors and documents

term-document matrix

In a **term-document matrix**, each row represents a word in the vocabulary and each column represents a document from some collection. Fig. 15.1 shows a small selection from a term-document matrix showing the occurrence of four words in four plays by Shakespeare. Each cell in this matrix represents the number of times a particular word (defined by the row) occurs in a particular document (defined by the column). Thus *clown* appeared 117 times in *Twelfth Night*.

vector space model

The term-document matrix of Fig. 15.1 was first defined as part of the **vector space model** of information retrieval (Salton, 1971). In this model, a document is represented as a count vector, a column in Fig. 15.2.

vector

To review some basic linear algebra, a **vector** is, at heart, just a list or array of numbers. So *As You Like It* is represented as the list [1,2,37,5] and *Julius Caesar* is represented as the list [8,12,1,0]. A **vector space** is a collection of vectors, char-

vector space

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	5	117	0	0

Figure 15.1 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

dimension

acterized by their **dimension**. The ordering of the numbers in a vector space is not arbitrary; each position indicates a meaningful dimension on which the documents can vary. Thus the first dimension for both these vectors corresponds to the number of times the word *battle* occurs, and we can compare each dimension, noting for example that the vectors for *As You Like It* and *Twelfth Night* have the same value 1 for the first dimension.

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	5	117	0	0

Figure 15.2 The term-document matrix for four words in four Shakespeare plays. The red boxes show that each document is represented as a column vector of length four.

We can think of the vector for a document as identifying a point in $|V|$-dimensional space; thus the documents in Fig. 15.2 are points in 4-dimensional space. Since 4-dimensional spaces are hard to draw in textbooks, Fig. 15.3 shows a visualization in two dimensions; we've arbitrarily chosen the dimensions corresponding to the words *battle* and *fool*.

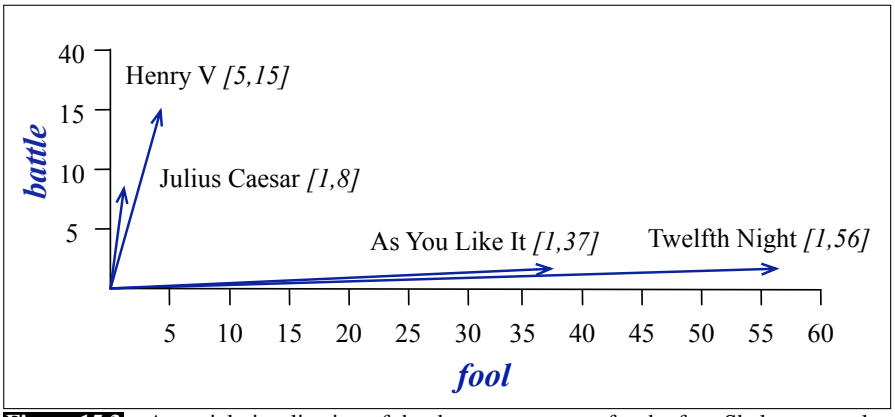

Figure 15.3 A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

Term-document matrices were originally defined as a means of finding similar documents for the task of document **information retrieval**. Two documents that are similar will tend to have similar words, and if two documents have similar words their column vectors will tend to be similar. The vectors for the comedies *As You like It* [1,2,37,5] and *Twelfth Night* [1,2,58,117] look a lot more like each other (more fools and clowns than soldiers and battles) than they do like *Julius Caesar* [8,12,1,0] or *Henry V* [15,36,5,0]. We can see the intuition with the raw numbers; in the

first dimension (battle) the comedies have low numbers and the others have high numbers, and we can see it visually in Fig. 15.3; we'll see very shortly how to quantify this intuition more formally.

A real term-document matrix, of course, wouldn't just have 4 rows and columns, let alone 2. More generally, the term-document matrix X has $|V|$ rows (one for each word type in the vocabulary) and D columns (one for each document in the collection); as we'll see, vocabulary sizes are generally at least in the tens of thousands, and the number of documents can be enormous (think about all the pages on the web).

information retrieval

Information retrieval (IR) is the task of finding the document d from the D documents in some collection that best matches a query q. For IR we'll therefore also represent a query by a vector, also of length $|V|$, and we'll need a way to compare two vectors to find how similar they are. (Doing IR will also require efficient ways to store and manipulate these vectors, which is accomplished by making use of the convenient fact that these vectors are sparse, i.e., mostly zeros). Later in the chapter we'll introduce some of the components of this vector comparison process: the tf-idf term weighting, and the cosine similarity metric.

15.1.2 Words as vectors

We've seen that documents can be represented as vectors in a vector space. But vector semantics can also be used to represent the meaning of *words*, by associating each word with a vector.

row vector

The word vector is now a **row vector** rather than a column vector, and hence the dimensions of the vector are different. The four dimensions of the vector for *fool*, [37,58,1,5], correspond to the four Shakespeare plays. The same four dimensions are used to form the vectors for the other 3 words: *clown*, [5, 117, 0, 0]; *battle*, [1,1,8,15]; and *soldier* [2,2,12,36]. Each entry in the vector thus represents the counts of the word's occurrence in the document corresponding to that dimension.

For documents, we saw that similar documents had similar vectors, because similar documents tend to have similar words. This same principle applies to words: similar words have similar vectors because they tend to occur in similar documents. The term-document matrix thus lets us represent the meaning of a word by the documents it tends to occur in.

However, it is most common to use a different kind of context for the dimensions of a word's vector representation. Rather than the term-document matrix we use the

term-term matrix
word-word matrix

term-term matrix, more commonly called the **word-word matrix** or the **term-context matrix**, in which the columns are labeled by words rather than documents. This matrix is thus of dimensionality $|V| \times |V|$ and each cell records the number of times the row (target) word and the column (context) word co-occur in some context in some training corpus. The context could be the document, in which case the cell represents the number of times the two words appear in the same document. It is most common, however, to use smaller contexts, generally a window around the word, for example of 4 words to the left and 4 words to the right, in which case the cell represents the number of times (in some training corpus) the column word occurs in such a ± 4 word window around the row word.

For example here are 7-word windows surrounding four sample words from the Brown corpus (just one example of each word):

sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first **pineapple** and another fruit whose taste she likened
well suited to programming on the digital **computer**. In finding the optimal R-stage policy from
for the purpose of gathering data and **information** necessary for the study authorized in the

For each word we collect the counts (from the windows around each occurrence) of the occurrences of context words. Fig. 15.4 shows a selection from the word-word co-occurrence matrix computed from the Brown corpus for these four words.

	aardvark	...	computer	data	pinch	result	sugar	...
apricot	0	...	0	0	1	0	1	
pineapple	0	...	0	0	1	0	1	
digital	0	...	2	1	0	1	0	
information	0	...	1	6	0	4	0	

Figure 15.4 Co-occurrence vectors for four words, computed from the Brown corpus, showing only six of the dimensions (hand-picked for pedagogical purposes). The vector for the word *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

Fig. 15.5 shows a spatial visualization. Note in Fig. 15.4 that the two words *apricot* and *pineapple* are more similar (both *pinch* and *sugar* tend to occur in their window) while *digital* and *information* are more similar.

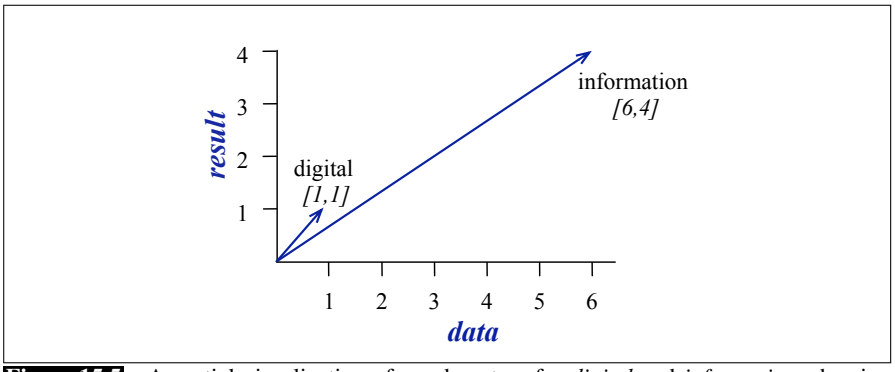

Figure 15.5 A spatial visualization of word vectors for *digital* and *information*, showing just two of the dimensions, corresponding to the words *data* and *result*.

Note that $|V|$, the length of the vector, is generally the size of the vocabulary, usually between 10,000 and 50,000 words (using the most frequent words in the training corpus; keeping words after about the most frequent 50,000 or so is generally not helpful). But of course since most of these numbers are zero these are **sparse** vector representations, and there are efficient algorithms for storing and computing with sparse matrices.

The size of the window used to collect counts can vary based on the goals of the representation, but is generally between 1 and 8 words on each side of the target word (for a total context of 3-17 words). In general, the shorter the window, the more syntactic the representations, since the information is coming from immediately nearby words; the longer the window, the more semantic the relations.

first-order co-occurrence

second-order co-occurrence

We have been talking loosely about similarity, but it's often useful to distinguish two kinds of similarity or association between words (Schütze and Pedersen, 1993). Two words have **first-order co-occurrence** (sometimes called **syntagmatic association**) if they are typically nearby each other. Thus *wrote* is a first-order associate of *book* or *poem*. Two words have **second-order co-occurrence** (sometimes called

paradigmatic association) if they have similar neighbors. Thus *wrote* is a second-order associate of words like *said* or *remarked*.

Now that we have some intuitions, let's move on to examine the details of computing a vector representation for a word. We'll begin with one of the most commonly used vector representations: PPMI or positive pointwise mutual information.

15.2 Weighing terms: Pointwise Mutual Information (PMI)

The co-occurrence matrix in Fig. 15.4 represented each cell by the raw frequency of the co-occurrence of two words. It turns out, however, that simple frequency isn't the best measure of association between words. One problem is that raw frequency is very skewed and not very discriminative. If we want to know what kinds of contexts are shared by *apricot* and *pineapple* but not by *digital* and *information*, we're not going to get good discrimination from words like *the*, *it*, or *they*, which occur frequently with all sorts of words and aren't informative about any particular word.

Instead we'd like context words that are particularly informative about the target word. The best weighting or measure of association between words should tell us how much more often than chance the two words co-occur.

Pointwise mutual information is just such a measure. It was proposed by Church and Hanks (1989) and (Church and Hanks, 1990), based on the notion of mutual information. The **mutual information** between two random variables X and Y is

mutual information

$$I(X,Y) = \sum_x \sum_y P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)} \tag{15.2}$$

pointwise mutual information

The **pointwise mutual information** (Fano, 1961)[1] is a measure of how often two events x and y occur, compared with what we would expect if they were independent:

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)} \tag{15.3}$$

We can apply this intuition to co-occurrence vectors by defining the pointwise mutual information association between a target word w and a context word c as

$$\text{PMI}(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)} \tag{15.4}$$

The numerator tells us how often we observed the two words together (assuming we compute probability by using the MLE). The denominator tells us how often we would **expect** the two words to co-occur assuming they each occurred independently, so their probabilities could just be multiplied. Thus, the ratio gives us an estimate of how much more the target and feature co-occur than we expect by chance.

PMI values range from negative to positive infinity. But negative PMI values (which imply things are co-occurring *less often* than we would expect by chance) tend to be unreliable unless our corpora are enormous. To distinguish whether two words whose individual probability is each 10^{-6} occur together more often than

[1] Fano actually used the phrase *mutual information* to refer to what we now call *pointwise mutual information* and the phrase *expectation of the mutual information* for what we now call *mutual information*; the term *mutual information* is still often used to mean *pointwise mutual information*.

chance, we would need to be certain that the probability of the two occurring together is significantly different than 10^{-12}, and this kind of granularity would require an enormous corpus. Furthermore it's not clear whether it's even possible to evaluate such scores of 'unrelatedness' with human judgments. For this reason it is more common to use Positive PMI (called **PPMI**) which replaces all negative PMI values with zero (Church and Hanks 1989, Dagan et al. 1993, Niwa and Nitta 1994)[2]:

PPMI

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0) \tag{15.5}$$

More formally, let's assume we have a co-occurrence matrix F with W rows (words) and C columns (contexts), where f_{ij} gives the number of times word w_i occurs in context c_j. This can be turned into a PPMI matrix where $ppmi_{ij}$ gives the PPMI value of word w_i with context c_j as follows:

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \quad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \quad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \tag{15.6}$$

$$\text{PPMI}_{ij} = \max(\log_2 \frac{P_{ij}}{P_{i*}P_{*j}}, 0) \tag{15.7}$$

Thus for example we could compute PPMI(w=information,c=data), assuming we pretended that Fig. 15.4 encompassed all the relevant word contexts/dimensions, as follows:

$$P(\text{w=information,c=data}) = \frac{6}{19} = .316$$
$$P(\text{w=information}) = \frac{11}{19} = .579$$
$$P(\text{c=data}) = \frac{7}{19} = .368$$
$$\text{ppmi(information,data)} = \log 2(.316/(.368 * .579)) = .568$$

Fig. 15.6 shows the joint probabilities computed from the counts in Fig. 15.4, and Fig. 15.7 shows the PPMI values.

	p(w,context)					p(w)
	computer	**data**	**pinch**	**result**	**sugar**	**p(w)**
apricot	0	0	0.05	0	0.05	0.11
pineapple	0	0	0.05	0	0.05	0.11
digital	0.11	0.05	0	0.05	0	0.21
information	0.05	.32	0	0.21	0	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

Figure 15.6 Replacing the counts in Fig. 15.4 with joint probabilities, showing the marginals around the outside.

PMI has the problem of being biased toward infrequent events; very rare words tend to have very high PMI values. One way to reduce this bias toward low frequency

[2] Positive PMI also cleanly solves the problem of what to do with zero counts, using 0 to replace the $-\infty$ from $\log(0)$.

	computer	data	pinch	result	sugar
apricot	0	0	2.25	0	2.25
pineapple	0	0	2.25	0	2.25
digital	1.66	0	0	0	0
information	0	0.57	0	0.47	0

Figure 15.7 The PPMI matrix showing the association between words and context words, computed from the counts in Fig. 15.4 again showing five dimensions. Note that the 0 ppmi values are ones that had a negative pmi; for example pmi(*information,computer*) = $\log 2(.05/(.16*.58)) = -0.618$, meaning that *information* and *computer* co-occur in this mini-corpus slightly less often than we would expect by chance, and with ppmi we replace negative values by zero. Many of the zero ppmi values had a pmi of $-\infty$, like pmi(*apricot,computer*) = $\log 2(0/(0.16*0.11)) = \log 2(0) = -\infty$.

events is to slightly change the computation for $P(c)$, using a different function $P_\alpha(c)$ that raises contexts to the power of α (Levy et al., 2015):

$$\text{PPMI}_\alpha(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P_\alpha(c)}, 0) \tag{15.8}$$

$$P_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha} \tag{15.9}$$

Levy et al. (2015) found that a setting of $\alpha = 0.75$ improved performance of embeddings on a wide range of tasks (drawing on a similar weighting used for skip-grams (Mikolov et al., 2013b) and GloVe (Pennington et al., 2014)). This works because raising the probability to $\alpha = 0.75$ increases the probability assigned to rare contexts, and hence lowers their PMI ($P_\alpha(c) > P(c)$ when c is rare).

Another possible solution is Laplace smoothing: Before computing PMI, a small constant k (values of 0.1-3 are common) is added to each of the counts, shrinking (discounting) all the non-zero values. The larger the k, the more the non-zero counts are discounted.

	computer	data	pinch	result	sugar
apricot	2	2	3	2	3
pineapple	2	2	3	2	3
digital	4	3	2	3	2
information	3	8	2	6	2

Figure 15.8 Laplace (add-2) smoothing of the counts in Fig. 15.4.

	computer	data	pinch	result	sugar
apricot	0	0	0.56	0	0.56
pineapple	0	0	0.56	0	0.56
digital	0.62	0	0	0	0
information	0	0.58	0	0.37	0

Figure 15.9 The Add-2 Laplace smoothed PPMI matrix from the add-2 smoothing counts in Fig. 15.8.

15.2.1 Alternatives to PPMI for measuring association

While PPMI is quite popular, it is by no means the only measure of association between two words (or between a word and some other feature). Other common

measures of association come from information retrieval (tf-idf, Dice) or from hypothesis testing (the t-test, the likelihood-ratio test). In this section we briefly summarize one of each of these types of measures.

tf-idf
term frequency

Let's first consider the standard weighting scheme for term-document matrices in information retrieval, called **tf-idf**. Tf-idf (this is a hyphen, not a minus sign) is the product of two factors. The first is the **term frequency** (Luhn, 1957): simply the frequency of the word in the document, although we may also use functions of this frequency like the log frequency.

inverse document frequency
IDF

The second factor is used to give a higher weight to words that occur only in a few documents. Terms that are limited to a few documents are useful for discriminating those documents from the rest of the collection; terms that occur frequently across the entire collection aren't as helpful. The **inverse document frequency** or **IDF** term weight (Sparck Jones, 1972) is one way of assigning higher weights to these more discriminative words. IDF is defined using the fraction N/df_i, where N is the total number of documents in the collection, and df_i is the number of documents in which term i occurs. The fewer documents in which a term occurs, the higher this weight. The lowest weight of 1 is assigned to terms that occur in all the documents. Because of the large number of documents in many collections, this measure is usually squashed with a log function. The resulting definition for inverse document frequency (IDF) is thus

$$\text{idf}_i = \log\left(\frac{N}{df_i}\right) \tag{15.10}$$

tf-idf

Combining term frequency with IDF results in a scheme known as **tf-idf** weighting of the value for word i in document j, w_{ij}:

$$w_{ij} = \text{tf}_{ij}\text{idf}_i \tag{15.11}$$

Tf-idf thus prefers words that are frequent in the current document j but rare overall in the collection.

The tf-idf weighting is by far the dominant way of weighting co-occurrence matrices in information retrieval, but also plays a role in many other aspects of natural language processing including summarization.

Tf-idf, however, is not generally used as a component in measures of word similarity; for that PPMI and significance-testing metrics like t-test and likelihood-ratio are more common. The **t-test** statistic, like PMI, can be used to measure how much more frequent the association is than chance. The t-test statistic computes the difference between observed and expected means, normalized by the variance. The higher the value of t, the greater the likelihood that we can reject the null hypothesis that the observed and expected means are the same.

t-test

$$t = \frac{\bar{x} - \mu}{\sqrt{\frac{s^2}{N}}} \tag{15.12}$$

When applied to association between words, the null hypothesis is that the two words are independent, and hence $P(a,b) = P(a)P(b)$ correctly models the relationship between the two words. We want to know how different the actual MLE probability $P(a,b)$ is from this null hypothesis value, normalized by the variance. The variance s^2 can be approximated by the expected probability $P(a)P(b)$ (see Manning and Schütze (1999)). Ignoring N (since it is constant), the resulting t-test association measure is thus (Curran, 2003):

$$\text{t-test}(a,b) = \frac{P(a,b) - P(a)P(b)}{\sqrt{P(a)P(b)}} \tag{15.13}$$

See the Historical Notes section for a summary of various other weighting factors for distributional models of meaning.

15.3 Measuring similarity: the cosine

To define similarity between two target words v and w, we need a measure for taking two such vectors and giving a measure of vector similarity. By far the most common similarity metric is the **cosine** of the angle between the vectors. In this section we'll motivate and introduce this important measure.

dot product
inner product

The cosine—like most measures for vector similarity used in NLP—is based on the **dot product** operator from linear algebra, also called the **inner product**:

$$\text{dot-product}(\vec{v},\vec{w}) = \vec{v}\cdot\vec{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N \tag{15.14}$$

As we will see, most metrics for similarity between vectors are based on the dot product. The dot product acts as a similarity metric because it will tend to be high just when the two vectors have large values in the same dimensions. Alternatively, vectors that have zeros in different dimensions—orthogonal vectors— will have a dot product of 0, representing their strong dissimilarity.

vector length

This raw dot-product, however, has a problem as a similarity metric: it favors **long** vectors. The **vector length** is defined as

$$|\vec{v}| = \sqrt{\sum_{i=1}^{N} v_i^2} \tag{15.15}$$

The dot product is higher if a vector is longer, with higher values in each dimension. More frequent words have longer vectors, since they tend to co-occur with more words and have higher co-occurrence values with each of them. Raw dot product thus will be higher for frequent words. But this is a problem; we'd like a similarity metric that tells us how similar two words are irregardless of their frequency.

The simplest way to modify the dot product to normalize for the vector length is to divide the dot product by the lengths of each of the two vectors. This **normalized dot product** turns out to be the same as the cosine of the angle between the two vectors, following from the definition of the dot product between two vectors $\vec{a}$ and $\vec{b}$:

$$\begin{aligned} \vec{a}\cdot\vec{b} &= |\vec{a}||\vec{b}|\cos\theta \\ \frac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|} &= \cos\theta \end{aligned} \tag{15.16}$$

cosine

The **cosine** similarity metric between two vectors $\vec{v}$ and $\vec{w}$ thus can be computed as:

$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}} \tag{15.17}$$

unit vector

For some applications we pre-normalize each vector, by dividing it by its length, creating a **unit vector** of length 1. Thus we could compute a unit vector from $\vec{a}$ by dividing it by $|\vec{a}|$. For unit vectors, the dot product is the same as the cosine.

The cosine value ranges from 1 for vectors pointing in the same direction, through 0 for vectors that are orthogonal, to -1 for vectors pointing in opposite directions. But raw frequency or PPMI are non-negative, so the cosine for these vectors ranges from 0-1.

Let's see how the cosine correctly predicts which of the words *apricot* or *digital* is closer in meaning to *information*, just using raw counts from the following simplified table:

	large	data	computer
apricot	2	0	0
digital	0	1	2
information	1	6	1

$$\cos(\text{apricot},\text{information}) = \frac{2+0+0}{\sqrt{4+0+0}\sqrt{1+36+1}} = \frac{2}{2\sqrt{38}} = .16$$
$$\cos(\text{digital},\text{information}) = \frac{0+6+2}{\sqrt{0+1+4}\sqrt{1+36+1}} = \frac{8}{\sqrt{38}\sqrt{5}} = .58 \tag{15.18}$$

The model correctly predicts that *information* is closer to *digital* than it is to *apricot*. Fig. 15.10 shows a visualization.

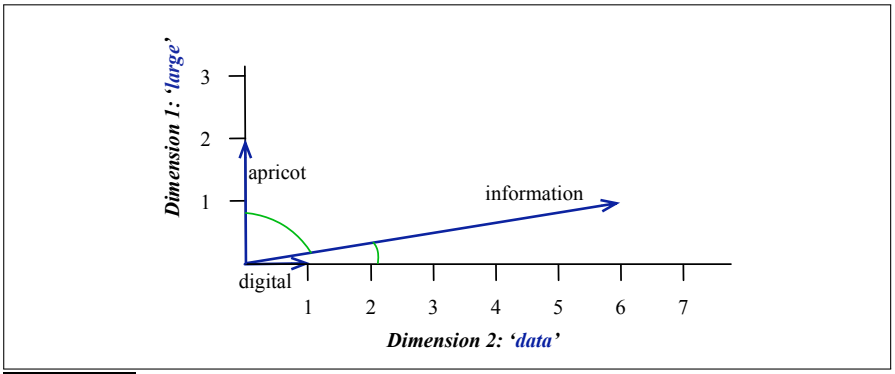

Figure 15.10 A graphical demonstration of the cosine measure of similarity, showing vectors for three words (*apricot*, *digital*, and *information*) in the two dimensional space defined by counts of the words *data* and *large* in the neighborhood. Note that the angle between *digital* and *information* is smaller than the angle between *apricot* and *information*.

Fig. 15.11 uses clustering of vectors as a way to visualize what words are most similar to other ones (Rohde et al., 2006).

15.3.1 Alternative Similarity Metrics

Jaccard

There are alternatives to the cosine metric for measuring similarity. The **Jaccard**

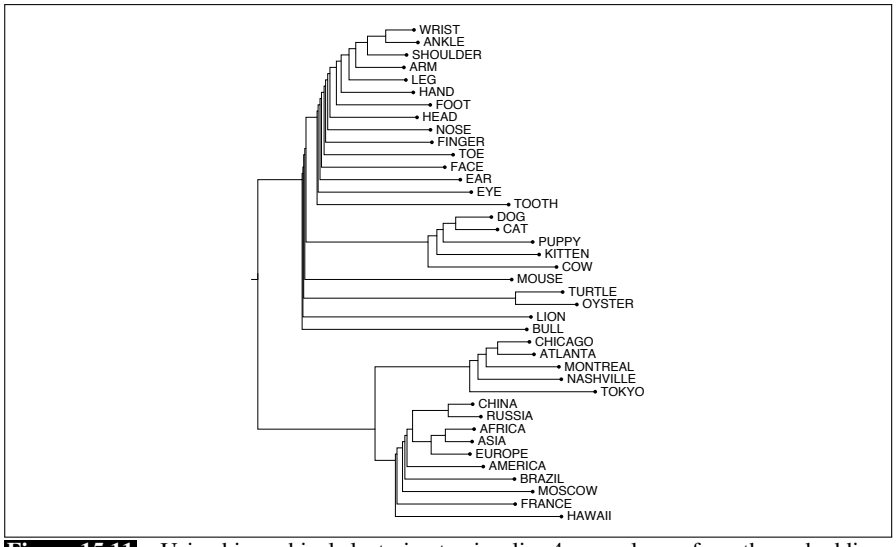

Figure 15.11 Using hierarchical clustering to visualize 4 noun classes from the embeddings produced by Rohde et al. (2006). These embeddings use a window size of ±4, and 14,000 dimensions, with 157 closed-class words removed. Rather than PPMI, these embeddings compute each cell via the positive correlation (the correlation between word pairs, with negative values replaced by zero), followed by a square root. This visualization uses hierarchical clustering, with correlation as the similarity function. From Rohde et al. (2006).

(Jaccard 1908, Jaccard 1912) measure, originally designed for binary vectors, was extended by Grefenstette (1994) to vectors of weighted associations as follows:

$$\text{sim}_{\text{Jaccard}}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N}\min(v_i,w_i)}{\sum_{i=1}^{N}\max(v_i,w_i)} \tag{15.19}$$

The numerator of the Grefenstette/Jaccard function uses the min function, essentially computing the (weighted) number of overlapping features (since if either vector has a zero association value for an attribute, the result will be zero). The denominator can be viewed as a normalizing factor.

Dice

The **Dice** measure, was similarly extended from binary vectors to vectors of weighted associations; one extension from Curran (2003) uses the Jaccard numerator but uses as the denominator normalization factor the total weighted value of non-zero entries in the two vectors.

$$\text{sim}_{\text{Dice}}(\vec{v},\vec{w}) = \frac{2\times\sum_{i=1}^{N}\min(v_i,w_i)}{\sum_{i=1}^{N}(v_i+w_i)} \tag{15.20}$$

Finally, there is a family of information-theoretic distributional similarity measures (Pereira et al. 1993, Dagan et al. 1994, Dagan et al. 1999, Lee 1999). The intuition of these models is that if two vectors, $\vec{v}$ and $\vec{w}$, each express a probability distribution (their values sum to one), then they are are similar to the extent that these probability distributions are similar. The basis of comparing two probability distributions P and Q is the **Kullback-Leibler divergence** or **KL divergence** or **relative entropy** (Kullback and Leibler, 1951):

KL divergence

$$D(P||Q) = \sum_{x} P(x)\log\frac{P(x)}{Q(x)} \tag{15.21}$$

$$\text{PMI}(w,f) = \log_2 \frac{P(w,f)}{P(w)P(f)} \quad (15.4)$$

$$\text{t-test}(w,f) = \frac{P(w,f)-P(w)P(f)}{\sqrt{P(f)P(w)}} \quad (15.13)$$

$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}} \quad (15.17)$$

$$\text{Jaccard}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} \max(v_i,w_i)} \quad (15.19)$$

$$\text{Dice}(\vec{v},\vec{w}) = \frac{2\times\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} (v_i+w_i)} \quad (15.20)$$

$$\text{JS}(\vec{v}||\vec{w}) = D(\vec{v}|\frac{\vec{v}+\vec{w}}{2}) + D(\vec{w}|\frac{\vec{v}+\vec{w}}{2}) \quad (15.23)$$

Figure 15.12 Defining word similarity: measures of association between a target word w and a feature $f = (r, w')$ to another word w', and measures of vector similarity between word co-occurrence vectors $\vec{v}$ and $\vec{w}$.

Unfortunately, the KL-divergence is undefined when $Q(x) = 0$ and $P(x) \neq 0$, which is a problem since these word-distribution vectors are generally quite sparse. One alternative (Lee, 1999) is to use the **Jensen-Shannon divergence**, which represents the divergence of each distribution from the mean of the two and doesn't have this problem with zeros.

Jensen-Shannon divergence

$$JS(P||Q) = D(P|\frac{P+Q}{2}) + D(Q|\frac{P+Q}{2}) \quad (15.22)$$

Rephrased in terms of vectors $\vec{v}$ and $\vec{w}$,

$$\text{sim}_{\text{JS}}(\vec{v}||\vec{w}) = D(\vec{v}|\frac{\vec{v}+\vec{w}}{2}) + D(\vec{w}|\frac{\vec{v}+\vec{w}}{2}) \quad (15.23)$$

Figure 15.12 summarizes the measures of association and of vector similarity that we have designed. See the Historical Notes section for a summary of other vector similarity measures.

15.4 Using syntax to define a word's context

Instead of defining a word's context by nearby words, we could instead define it by the syntactic relations of these neighboring words. This intuition was first suggested by Harris (1968), who pointed out the relation between meaning and syntactic combinatory possibilities:

> The meaning of entities, and the meaning of grammatical relations among them, is related to the restriction of combinations of these entities relative to other entities.

Consider the words *duty* and *responsibility*. The similarity between the meanings of these words is mirrored in their syntactic behavior. Both can be modified by adjectives like *additional, administrative, assumed, collective, congressional, constitutional*, and both can be the direct objects of verbs like *assert, assign, assume, attend to, avoid, become, breach* (Lin and Pantel, 2001).

In other words, we could define the dimensions of our context vector not by the presence of a word in a window, but by the presence of a word in a particular dependency (or other grammatical relation), an idea first worked out by Hindle (1990). Since each word can be in a variety of different dependency relations with other words, we'll need to augment the feature space. Each feature is now a pairing of a word and a relation, so instead of a vector of $|V|$ features, we have a vector of $|V| \times R$ features, where R is the number of possible relations. Figure 15.13 shows a schematic early example of such a vector, taken from Lin (1998), showing one row for the word *cell*. As the value of each attribute we have shown the raw frequency of the feature co-occurring with *cell*.

	subj-of, absorb	*subj-of*, adapt	*subj-of*, behave	...	*pobj-of*, inside	*pobj-of*, into	...	*nmod-of*, abnormality	*nmod-of*, anemia	*nmod-of*, architecture	...	*obj-of*, attack	*obj-of*, call	*obj-of*, come from	*obj-of*, decorate	...	*nmod*, bacteria	*nmod*, body	*nmod*, bone marrow
cell	1	1	1		16	30		3	8	1		6	11	3	2		3	2	2

Figure 15.13 Co-occurrence vector for the word *cell*, from Lin (1998), showing grammatical function (dependency) features. Values for each attribute are frequency counts from a 64-million word corpus, parsed by an early version of MINIPAR.

An alternative to augmenting the feature space is to use the dependency paths just as a way to accumulate feature counts, but continue to have just $|V|$ dimensions of words. The value for a context word dimension, instead of counting all instances of that word in the neighborhood of the target word, counts only words in a dependency relationship with the target word. More complex models count only certain kinds of dependencies, or weigh the counts based on the length of the dependency path (Padó and Lapata, 2007). And of course we can use PPMI or other weighting schemes to weight the elements of these vectors rather than raw frequency.

15.5 Evaluating Vector Models

Of course the most important evaluation metric for vector models is extrinsic evaluation on tasks; adding them as features into any NLP task and seeing whether this improves performance.

Nonetheless it is useful to have intrinsic evaluations. The most common metric is to test their performance on **similarity**, and in particular on computing the correlation between an algorithm's word similarity scores and word similarity ratings assigned by humans. The various sets of human judgments are the same as we described in Chapter 17 for thesaurus-based similarity, summarized here for convenience. **WordSim-353** (Finkelstein et al., 2002) is a commonly used set of of ratings from 0 to 10 for 353 noun pairs; for example (*plane*, *car*) had an average score of 5.77. **SimLex-999** (Hill et al., 2015) is a more difficult dataset that quantifies similarity (*cup, mug*) rather than relatedness (*cup, coffee*), and including both concrete and abstract adjective, noun and verb pairs. The **TOEFL dataset** is a set of 80 questions, each consisting of a target word with 4 additional word choices; the task is to choose which is the correct synonym, as in the example: *Levied is closest in mean-*

ing to: imposed, believed, requested, correlated (Landauer and Dumais, 1997). All of these datasets present words without context.

Slightly more realistic are intrinsic similarity tasks that include context. The Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012) offers a richer evaluation scenario, giving human judgments on 2,003 pairs of words in their sentential context, including nouns, verbs, and adjectives. This dataset enables the evaluation of word similarity algorithms that can make use of context words. The *semantic textual similarity* task (Agirre et al. 2012, Agirre et al. 2015) evaluates the performance of sentence-level similarity algorithms, consisting of a set of pairs of sentences, each pair with human-labeled similarity scores.

Another task used for evaluate is an analogy task, where the system has to solve problems of the form *a is to b as c is to d*, given *a, b,* and *c* and having to find *d*. The system is given two words that participate in a relation (for example *Athens* and *Greece*, which participate in the capital relation) and a word like *Oslo* and must find the word *Norway*. Or more syntactically-oriented examples: given *mouse*, *mice*, and *dollar* the system must return *dollars*. Large sets of such tuples have been created (Mikolov et al. 2013a, Mikolov et al. 2013c).

15.6 Summary

- The **term-document** matrix, first created for information retrieval, has rows for each word (**term**) in the vocabulary and a column for each document. The cell specify the count of that term in the document.
- The **word-context** (or **word-word**, or **term-term**) matrix has a row for each (target) word in the vocabulary and a column for each context term in the vocabulary. Each cell indicates the number of times the context term occurs in a window (of a specified size) around the target word in a corpus.
- A common weighting for the Instead of using the raw word word co-occurrence matrix, it is often weighted. A common weighting is **positive pointwise mutual information** or **PPMI**.
- Alternative weightings are **tf-idf**, used for information retrieval task, and significance-based methods like **t-test**.
- PPMI and other versions of the word-word matrix can be viewed as offering high-dimensional vector representations of words that are **sparse** (since most values are 0).
- The cosine of two vectors is a common function used for word similarity.

Bibliographical and Historical Notes

Models of distributional word similarity arose out of research in linguistics and psychology of the 1950s. The idea that meaning was related to distribution of words in context was widespread in linguistic theory of the 1950s; even before the well-known Firth (1957) and Harris (1968) dictums discussed earlier, Joos (1950) stated that

> the linguist's "meaning" of a morpheme... is by definition the set of conditional probabilities of its occurrence in context with all other morphemes.

The related idea that the meaning of a word could be modeled as a point in a Euclidean space and that the similarity of meaning between two words could be modeled as the distance between these points was proposed in psychology by Osgood et al. (1957).

The application of these ideas in a computational framework was first made by Sparck Jones (1986) and became a core principle of information retrieval, whence it came into broader use in language processing.

The idea of defining words by a vector of discrete features has a venerable history in our field, with roots at least as far back Descartes and Leibniz (Wierzbicka 1992, Wierzbicka 1996). By the middle of the 20th century, beginning with the work of Hjelmslev (Hjelmslev, 1969) and fleshed out in early models of generative grammar (Katz and Fodor, 1963), the idea arose of representing meaning with **semantic features**, symbols that represent some sort of primitive meaning. For example words like *hen*, *rooster*, or *chick*, have something in common (they all describe chickens) and something different (their age and sex), representable as:

semantic feature

hen	`+female, +chicken, +adult`
rooster	`-female, +chicken, +adult`
chick	`+chicken, -adult`

The dimensions used by vector models of meaning to define words are only abstractly related to these small fixed number of hand-built dimensions. Nonetheless, there has been some attempt to show that certain dimensions of embedding models do contribute some specific compositional aspect of meaning like these early semantic features.

Turney and Pantel (2010) is an excellent and comprehensive survey of vector semantics.

There are a wide variety of other weightings and methods for word similarity. The largest class of methods not discussed in this chapter are the variants to and details of the **information-theoretic** methods like Jensen-Shannon divergence, KL-divergence and α-skew divergence that we briefly introduced (Pereira et al. 1993, Dagan et al. 1994, Dagan et al. 1999, Lee 1999, Lee 2001). Manning and Schütze (1999, Chapters 5 and 8) give collocation measures and other related similarity measures.

Exercises

Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., Guo, W., Lopez-Gazpio, I., Maritxalar, M., Mihalcea, R., Rigau, G., Uria, L., and Wiebe, J. (2015). 2015 SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *SemEval-15*, pp. 252–263.

Agirre, E., Diab, M., Cer, D., and Gonzalez-Agirre, A. (2012). Semeval-2012 task 6: A pilot on semantic textual similarity. In *SemEval-12*, pp. 385–393.

Church, K. W. and Hanks, P. (1989). Word association norms, mutual information, and lexicography. In *ACL-89*, Vancouver, B.C., pp. 76–83.

Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*, *16*(1), 22–29.

Curran, J. R. (2003). *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh.

Dagan, I., Lee, L., and Pereira, F. C. N. (1999). Similarity-based models of cooccurrence probabilities. *Machine Learning*, *34*(1–3), 43–69.

Dagan, I., Marcus, S., and Markovitch, S. (1993). Contextual word similarity and estimation from sparse data. In *ACL-93*, Columbus, Ohio, pp. 164–171.

Dagan, I., Pereira, F. C. N., and Lee, L. (1994). Similarity-base estimation of word cooccurrence probabilities. In *ACL-94*, Las Cruces, NM, pp. 272–278.

Fano, R. M. (1961). *Transmission of Information: A Statistical Theory of Communications*. MIT Press.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2002). Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, *20*(1), 116—131.

Firth, J. R. (1957). A synopsis of linguistic theory 1930–1955. In *Studies in Linguistic Analysis*. Philological Society. Reprinted in Palmer, F. (ed.) 1968. Selected Papers of J. R. Firth. Longman, Harlow.

Gould, S. J. (1980). *The Panda's Thumb*. Penguin Group.

Grefenstette, G. (1994). *Explorations in Automatic Thesaurus Discovery*. Kluwer, Norwell, MA.

Harris, Z. S. (1954). Distributional structure. *Word*, *10*, 146–162. Reprinted in J. Fodor and J. Katz, *The Structure of Language*, Prentice Hall, 1964 and in Z. S. Harris, *Papers in Structural and Transformational Linguistics*, Reidel, 1970, 775–794.

Harris, Z. S. (1968). *Mathematical Structures of Language*. John Wiley.

Hill, F., Reichart, R., and Korhonen, A. (2015). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. Preprint published on arXiv. arXiv:1408.3456.

Hindle, D. (1990). Noun classification from predicate-argument structures. In *ACL-90*, Pittsburgh, PA, pp. 268–275.

Hjelmslev, L. (1969). *Prologomena to a Theory of Language*. University of Wisconsin Press. Translated by Francis J. Whitfield; original Danish edition 1943.

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *ACL 2012*, pp. 873–882.

Jaccard, P. (1908). Nouvelles recherches sur la distribution florale. *Bulletin de la Société Vaudoise des Sciences Naturelles*, *44*, 223–227.

Jaccard, P. (1912). The distribution of the flora of the alpine zone. *New Phytologist*, *11*, 37–50.

Joos, M. (1950). Description of language design. *JASA*, *22*, 701–708.

Katz, J. J. and Fodor, J. A. (1963). The structure of a semantic theory. *Language*, *39*, 170–210.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, *22*, 79–86.

Landauer, T. K. and Dumais, S. T. (1997). A solution to Plato's problem: The Latent Semantic Analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, *104*, 211–240.

Lee, L. (1999). Measures of distributional similarity. In *ACL-99*, pp. 25–32.

Lee, L. (2001). On the effectiveness of the skew divergence for statistical language analysis. In *Artificial Intelligence and Statistics*, pp. 65–72.

Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *TACL*, *3*, 211–225.

Lin, D. (1998). Automatic retrieval and clustering of similar words. In *COLING/ACL-98*, Montreal, pp. 768–774.

Lin, D. and Pantel, P. (2001). Dirt: discovery of inference rules from text. In *KDD-01*, pp. 323–328.

Luhn, H. P. (1957). A statistical approach to the mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, *1*(4), 309–317.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In *ICLR 2013*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *NIPS 13*, pp. 3111–3119.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *NAACL HLT 2013*, pp. 746–751.

Nida, E. A. (1975). *Componential Analysis of Meaning: An Introduction to Semantic Structures*. Mouton, The Hague.

Niwa, Y. and Nitta, Y. (1994). Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In *ACL-94*, pp. 304–309.

Osgood, C. E., Suci, G. J., and Tannenbaum, P. H. (1957). *The Measurement of Meaning*. University of Illinois Press.

Padó, S. and Lapata, M. (2007). Dependency-based construction of semantic space models. *Computational Linguistics*, *33*(2), 161–199.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP 2014*, pp. 1532–1543.

Pereira, F. C. N., Tishby, N., and Lee, L. (1993). Distributional clustering of English words. In *ACL-93*, Columbus, Ohio, pp. 183–190.

Rohde, D. L. T., Gonnerman, L. M., and Plaut, D. C. (2006). An improved model of semantic similarity based on lexical co-occurrence. *Communications of the ACM*, *8*, 627–633.

Salton, G. (1971). *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice Hall.

Schütze, H. and Pedersen, J. (1993). A vector model for syntagmatic and paradigmatic relatedness. In *Proceedings of the 9th Annual Conference of the UW Centre for the New OED and Text Research*, pp. 104–113.

Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*(1), 11–21.

Sparck Jones, K. (1986). *Synonymy and Semantic Classification*. Edinburgh University Press, Edinburgh. Republication of 1964 PhD Thesis.

Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *JAIR*, *37*(1), 141–188.

Wierzbicka, A. (1992). *Semantics, Culture, and Cognition: University Human Concepts in Culture-Specific Configurations*. Oxford University Press.

Wierzbicka, A. (1996). *Semantics: Primes and Universals*. Oxford University Press.