Journal of Artificial Intelligence Research 37 (2010) 141-188 Submitted 10/09; published 02/10

From Frequency to Meaning: Vector Space Models of Semantics

Peter D. Turney PETER.TURNEY@NRC-CNRC.GC.CA
National Research Council Canada
Ottawa, Ontario, Canada, K1A 0R6

Patrick Pantel ME@PATRICKPANTEL.COM
Yahoo! Labs
Sunnyvale, CA, 94089, USA

Abstract

Computers understand very little of the meaning of human language. This profoundly limits our ability to give instructions to computers, the ability of computers to explain their actions to us, and the ability of computers to analyse and process text. Vector space models (VSMs) of semantics are beginning to address these limits. This paper surveys the use of VSMs for semantic processing of text. We organize the literature on VSMs according to the structure of the matrix in a VSM. There are currently three broad classes of VSMs, based on term–document, word–context, and pair–pattern matrices, yielding three classes of applications. We survey a broad range of applications in these three categories and we take a detailed look at a specific open source project in each category. Our goal in this survey is to show the breadth of applications of VSMs for semantics, to provide a new perspective on VSMs for those who are already familiar with the area, and to provide pointers into the literature for those who are less familiar with the field.

1. Introduction

One of the biggest obstacles to making full use of the power of computers is that they currently understand very little of the meaning of human language. Recent progress in search engine technology is only scratching the surface of human language, and yet the impact on society and the economy is already immense. This hints at the transformative impact that deeper semantic technologies will have. Vector space models (VSMs), surveyed in this paper, are likely to be a part of these new semantic technologies.

In this paper, we use the term *semantics* in a general sense, as the meaning of a word, a phrase, a sentence, or any text in human language, and the study of such meaning. We are not concerned with narrower senses of *semantics*, such as the *semantic web* or approaches to semantics based on formal logic. We present a survey of VSMs and their relation with the distributional hypothesis as an approach to representing some aspects of natural language semantics.

The VSM was developed for the SMART information retrieval system (Salton, 1971) by Gerard Salton and his colleagues (Salton, Wong, & Yang, 1975). SMART pioneered many of the concepts that are used in modern search engines (Manning, Raghavan, & Schütze, 2008). The idea of the VSM is to represent each document in a collection as a point in a space (a vector in a vector space). Points that are close together in this space are semantically similar and points that are far apart are semantically distant. The user's query is represented as a point in the same space as the documents (the query is a *pseudo-document*). The documents are sorted in order of increasing distance (decreasing semantic similarity) from the query and then presented to the user.

The success of the VSM for information retrieval has inspired researchers to extend the VSM to other semantic tasks in natural language processing, with impressive results. For instance, Rapp (2003) used a vector-based representation of word meaning to achieve a score of 92.5% on multiple-choice synonym questions from the Test of English as a Foreign Language (TOEFL), whereas the average human score was 64.5%.[1] Turney (2006) used a vector-based representation of semantic relations to attain a score of 56% on multiple-choice analogy questions from the SAT college entrance test, compared to an average human score of 57%.[2]

In this survey, we have organized past work with VSMs according to the type of matrix involved: term–document, word–context, and pair–pattern. We believe that the choice of a particular matrix type is more fundamental than other choices, such as the particular linguistic processing or mathematical processing. Although these three matrix types cover most of the work, there is no reason to believe that these three types exhaust the possibilities. We expect future work will introduce new types of matrices and higher-order tensors.[3]

1.1 Motivation for Vector Space Models of Semantics

VSMs have several attractive properties. VSMs extract knowledge automatically from a given corpus, thus they require much less labour than other approaches to semantics, such as hand-coded knowledge bases and ontologies. For example, the main resource used in Rapp's (2003) VSM system for measuring word similarity is the British National Corpus (BNC),[4] whereas the main resource used in non-VSM systems for measuring word similarity (Hirst & St-Onge, 1998; Leacock & Chodrow, 1998; Jarmasz & Szpakowicz, 2003) is a lexicon, such as WordNet[5] or Roget's Thesaurus. Gathering a corpus for a new language is generally much easier than building a lexicon, and building a lexicon often involves also gathering a corpus, such as SemCor for WordNet (Miller, Leacock, Tengi, & Bunker, 1993).

VSMs perform well on tasks that involve measuring the similarity of meaning between words, phrases, and documents. Most search engines use VSMs to measure the similarity between a query and a document (Manning et al., 2008). The leading algorithms for measuring semantic relatedness use VSMs (Pantel & Lin, 2002a; Rapp, 2003; Turney, Littman, Bigham, & Shnayder, 2003). The leading algorithms for measuring the similarity of semantic relations also use VSMs (Lin & Pantel, 2001; Turney, 2006; Nakov & Hearst, 2008). (Section 2.4 discusses the differences between these types of similarity.)

We find VSMs especially interesting due to their relation with the distributional hypothesis and related hypotheses (see Section 2.7). The distributional hypothesis is that

1. Regarding the average score of 64.5% on the TOEFL questions, Landauer and Dumais (1997) note that, "Although we do not know how such a performance would compare, for example, with U.S. school children of a particular age, we have been told that the average score is adequate for admission to many universities."
2. This is the average score for highschool students in their senior year, applying to US universities. For more discussion of this score, see Section 6.3 in Turney's (2006) paper.
3. A vector is a first-order tensor and a matrix is a second-order tensor. See Section 2.5.
4. See http://www.natcorp.ox.ac.uk/.
5. See http://wordnet.princeton.edu/.

words that occur in similar contexts tend to have similar meanings (Wittgenstein, 1953; Harris, 1954; Weaver, 1955; Firth, 1957; Deerwester, Dumais, Landauer, Furnas, & Harshman, 1990). Efforts to apply this abstract hypothesis to concrete algorithms for measuring the similarity of meaning often lead to vectors, matrices, and higher-order tensors. This intimate connection between the distributional hypothesis and VSMs is a strong motivation for taking a close look at VSMs.

Not all uses of vectors and matrices count as vector space models. For the purposes of this survey, we take it as a defining property of VSMs that the values of the elements in a VSM must be derived from event frequencies, such as the number of times that a given word appears in a given context (see Section 2.6). For example, often a lexicon or a knowledge base may be viewed as a graph, and a graph may be represented using an adjacency matrix, but this does not imply that a lexicon is a VSM, because, in general, the values of the elements in an adjacency matrix are not derived from event frequencies. This emphasis on event frequencies brings unity to the variety of VSMs and explicitly connects them to the distributional hypothesis; furthermore, it avoids triviality by excluding many possible matrix representations.

1.2 Vectors in AI and Cognitive Science

Vectors are common in AI and cognitive science; they were common before the VSM was introduced by Salton et al. (1975). The novelty of the VSM was to use frequencies in a corpus of text as a clue for discovering semantic information.

In machine learning, a typical problem is to learn to classify or cluster a set of items (i.e., examples, cases, individuals, entities) represented as *feature vectors* (Mitchell, 1997; Witten & Frank, 2005). In general, the features are not derived from event frequencies, although this is possible (see Section 4.6). For example, a machine learning algorithm can be applied to classifying or clustering documents (Sebastiani, 2002).

Collaborative filtering and recommender systems also use vectors (Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994; Breese, Heckerman, & Kadie, 1998; Linden, Smith, & York, 2003). In a typical recommender system, we have a *person-item* matrix, in which the rows correspond to people (customers, consumers), the columns correspond to items (products, purchases), and the value of an element is the rating (poor, fair, excellent) that the person has given to the item. Many of the mathematical techniques that work well with term–document matrices (see Section 4) also work well with person-item matrices, but ratings are not derived from event frequencies.

In cognitive science, prototype theory often makes use of vectors. The basic idea of prototype theory is that some members of a category are more central than others (Rosch & Lloyd, 1978; Lakoff, 1987). For example, *robin* is a central (prototypical) member of the category *bird*, whereas *penguin* is more peripheral. Concepts have varying degrees of membership in categories (*graded* categorization). A natural way to formalize this is to represent concepts as vectors and categories as sets of vectors (Nosofsky, 1986; Smith, Osherson, Rips, & Keane, 1988). However, these vectors are usually based on numerical scores that are elicited by questioning human subjects; they are not based on event frequencies.

Another area of psychology that makes extensive use of vectors is psychometrics, which studies the measurement of psychological abilities and traits. The usual instrument for measurement is a test or questionnaire, such as a personality test. The results of a test are typically represented as a *subject-item* matrix, in which the rows represent the subjects (people) in an experiment and the columns represent the items (questions) in the test (questionnaire). The value of an element in the matrix is the answer that the corresponding subject gave for the corresponding item. Many techniques for vector analysis, such as factor analysis (Spearman, 1904), were pioneered in psychometrics.

In cognitive science, Latent Semantic Analysis (LSA) (Deerwester et al., 1990; Landauer & Dumais, 1997), Hyperspace Analogue to Language (HAL) (Lund, Burgess, & Atchley, 1995; Lund & Burgess, 1996), and related research (Landauer, McNamara, Dennis, & Kintsch, 2007) is entirely within the scope of VSMs, as defined above, since this research uses vector space models in which the values of the elements are derived from event frequencies, such as the number of times that a given word appears in a given context. Cognitive scientists have argued that there are empirical and theoretical reasons for believing that VSMs, such as LSA and HAL, are plausible models of some aspects of human cognition (Landauer et al., 2007). In AI, computational linguistics, and information retrieval, such plausibility is not essential, but it may be seen as a sign that VSMs are a promising area for further research.

1.3 Motivation for This Survey

This paper is a survey of vector space models of semantics. There is currently no comprehensive, up-to-date survey of this field. As we show in the survey, vector space models are a highly successful approach to semantics, with a wide range of potential and actual applications. There has been much recent growth in research in this area.

This paper should be of interest to all AI researchers who work with natural language, especially researchers who are interested in semantics. The survey will serve as a general introduction to this area and it will provide a framework – a unified perspective – for organizing the diverse literature on the topic. It should encourage new research in the area, by pointing out open problems and areas for further exploration.

This survey makes the following contributions:

New framework: We provide a new framework for organizing the literature: term–document, word–context, and pair–pattern matrices (see Section 2). This framework shows the importance of the structure of the matrix (the choice of rows and columns) in determining the potential applications and may inspire researchers to explore new structures (different kinds of rows and columns, or higher-order tensors instead of matrices).

New developments: We draw attention to pair–pattern matrices. The use of pair–pattern matrices is relatively new and deserves more study. These matrices address some criticisms that have been directed at word–context matrices, regarding lack of sensitivity to word order.

Breadth of approaches and applications: There is no existing survey that shows the breadth of potential and actual applications of VSMs for semantics. Existing summaries omit pair–pattern matrices (Landauer et al., 2007).

Focus on NLP and CL: Our focus in this survey is on systems that perform practical tasks in natural language processing and computational linguistics. Existing overviews focus on cognitive psychology (Landauer et al., 2007).

Success stories: We draw attention to the fact that VSMs are arguably the most successful approach to semantics, so far.

1.4 Intended Readership

Our goal in writing this paper has been to survey the state of the art in vector space models of semantics, to introduce the topic to those who are new to the area, and to give a new perspective to those who are already familiar with the area.

We assume our reader has a basic understanding of vectors, matrices, and linear algebra, such as one might acquire from an introductory undergraduate course in linear algebra, or from a text book (Golub & Van Loan, 1996). The basic concepts of vectors and matrices are more important here than the mathematical details. Widdows (2004) gives a gentle introduction to vectors from the perspective of semantics.

We also assume our reader has some familiarity with computational linguistics or information retrieval. Manning et al. (2008) provide a good introduction to information retrieval. For computational linguistics, we recommend Manning and Schütze's (1999) text.

If our reader is familiar with linear algebra and computational linguistics, this survey should present no barriers to understanding. Beyond this background, it is not necessary to be familiar with VSMs as they are used in information retrieval, natural language processing, and computational linguistics. However, if the reader would like to do some further background reading, we recommend Landauer et al.'s (2007) collection.

1.5 Highlights and Outline

This article is structured as follows. Section 2 explains our framework for organizing the literature on VSMs according to the type of matrix involved: term–document, word–context, and pair–pattern. In this section, we present an overview of VSMs, without getting into the details of how a matrix can be generated from a corpus of raw text.

After the high-level framework is in place, Sections 3 and 4 examine the steps involved in generating a matrix. Section 3 discusses linguistic processing and Section 4 reviews mathematical processing. This is the order in which a corpus would be processed in most VSM systems (first linguistic processing, then mathematical processing).

When VSMs are used for semantics, the input to the model is usually plain text. Some VSMs work directly with the raw text, but most first apply some linguistic processing to the text, such as stemming, part-of-speech tagging, word sense tagging, or parsing. Section 3 looks at some of these linguistic tools for semantic VSMs.

In a simple VSM, such as a simple term–document VSM, the value of an element in a document vector is the number of times that the corresponding word occurs in the given document, but most VSMs apply some mathematical processing to the raw frequency values. Section 4 presents the main mathematical operations: weighting the elements, smoothing the matrix, and comparing the vectors. This section also describes optimization strategies for comparing the vectors, such as distributed sparse matrix multiplication and randomized techniques.

By the end of Section 4, the reader will have a general view of the concepts involved in vector space models of semantics. We then take a detailed look at three VSM systems in Section 5. As a representative of term–document VSMs, we present the Lucene information retrieval library.[6] For word–context VSMs, we explore the Semantic Vectors package, which builds on Lucene.[7] As the representative of pair–pattern VSMs, we review the Latent Relational Analysis module in the S-Space package, which also builds on Lucene.[8] The source code for all three of these systems is available under open source licensing.

We turn to a broad survey of applications for semantic VSMs in Section 6. This section also serves as a short historical view of research with semantic VSMs, beginning with information retrieval in Section 6.1. Our purpose here is to give the reader an idea of the breadth of applications for VSMs and also to provide pointers into the literature, if the reader wishes to examine any of these applications in detail.

In a *term–document matrix*, rows correspond to terms and columns correspond to documents (Section 6.1). A document provides a context for understanding the term. If we generalize the idea of documents to chunks of text of arbitrary size (phrases, sentences, paragraphs, chapters, books, collections), the result is the *word–context* matrix, which includes the term–document matrix as a special case. Section 6.2 discusses applications for word–context matrices. Section 6.3 considers *pair–pattern* matrices, in which the rows correspond to pairs of terms and the columns correspond to the patterns in which the pairs occur.

In Section 7, we discuss alternatives to VSMs for semantics. Section 8 considers the future of VSMs, raising some questions about their power and their limitations. We conclude in Section 9.

2. Vector Space Models of Semantics

The theme that unites the various forms of VSMs that we discuss in this paper can be stated as the *statistical semantics hypothesis*: statistical patterns of human word usage can be used to figure out what people mean.[9] This general hypothesis underlies several more specific hypotheses, such as the *bag of words hypothesis*, the *distributional hypothesis*, the *extended distributional hypothesis*, and the *latent relation hypothesis*, discussed below.

2.1 Similarity of Documents: The Term–Document Matrix

In this paper, we use the following notational conventions: Matrices are denoted by bold capital letters, $\mathbf{A}$. Vectors are denoted by bold lowercase letters, $\mathbf{b}$. Scalars are represented by lowercase italic letters, c.

If we have a large collection of documents, and hence a large number of document vectors, it is convenient to organize the vectors into a matrix. The row vectors of the matrix correspond to terms (usually terms are words, but we will discuss some other possibilities)

6. See http://lucene.apache.org/java/docs/.
7. See http://code.google.com/p/semanticvectors/.
8. See http://code.google.com/p/airhead-research/wiki/LatentRelationalAnalysis.
9. This phrase was taken from the Faculty Profile of George Furnas at the University of Michigan, http://www.si.umich.edu/people/faculty-detail.htm?sid=41. The full quote is, "Statistical Semantics – Studies of how the statistical patterns of human word usage can be used to figure out what people mean, at least to a level sufficient for information access." The term *statistical semantics* appeared in the work of Furnas, Landauer, Gomez, and Dumais (1983), but it was not defined there.

and the column vectors correspond to documents (web pages, for example). This kind of matrix is called a *term–document* matrix.

In mathematics, a *bag* (also called a *multiset*) is like a set, except that duplicates are allowed. For example, $\{a, a, b, c, c, c\}$ is a bag containing a, b, and c. Order does not matter in bags and sets; the bags $\{a, a, b, c, c, c\}$ and $\{c, a, c, b, a, c\}$ are equivalent. We can represent the bag $\{a, a, b, c, c, c\}$ with the vector $\mathbf{x} = \langle 2, 1, 3\rangle$, by stipulating that the first element of $\mathbf{x}$ is the frequency of a in the bag, the second element is the frequency of b in the bag, and the third element is the frequency of c. A set of bags can be represented as a matrix $\mathbf{X}$, in which each column $\mathbf{x}_{:j}$ corresponds to a bag, each row $\mathbf{x}_{i:}$ corresponds to a unique member, and an element x_{ij} is the frequency of the i-th member in the j-th bag.

In a term–document matrix, a document vector represents the corresponding document as a bag of words. In information retrieval, the *bag of words hypothesis* is that we can estimate the relevance of documents to a query by representing the documents and the query as bags of words. That is, the frequencies of words in a document tend to indicate the relevance of the document to a query. The bag of words hypothesis is the basis for applying the VSM to information retrieval (Salton et al., 1975). The hypothesis expresses the belief that a column vector in a term–document matrix captures (to some degree) an aspect of the meaning of the corresponding document; what the document is *about*.

Let $\mathbf{X}$ be a term–document matrix. Suppose our document collection contains n documents and m unique terms. The matrix $\mathbf{X}$ will then have m rows (one row for each unique term in the vocabulary) and n columns (one column for each document). Let w_i be the i-th term in the vocabulary and let d_j be the j-th document in the collection. The i-th row in $\mathbf{X}$ is the row vector $\mathbf{x}_{i:}$ and the j-th column in $\mathbf{X}$ is the column vector $\mathbf{x}_{:j}$. The row vector $\mathbf{x}_{i:}$ contains n elements, one element for each document, and the column vector $\mathbf{x}_{:j}$ contains m elements, one element for each term. Suppose $\mathbf{X}$ is a simple matrix of frequencies. The element x_{ij} in $\mathbf{X}$ is the frequency of the i-th term w_i in the j-th document d_j.

In general, the value of most of the elements in $\mathbf{X}$ will be zero (the matrix is *sparse*), since most documents will use only a small fraction of the whole vocabulary. If we randomly choose a term w_i and a document d_j, it's likely that w_i does not occur anywhere in d_j, and therefore x_{ij} equals 0.

The pattern of numbers in $\mathbf{x}_{i:}$ is a kind of *signature* of the i-th term w_i; likewise, the pattern of numbers in $\mathbf{x}_{:j}$ is a signature of the j-th document d_j. That is, the pattern of numbers tells us, to some degree, what the term or document is *about*.

The vector $\mathbf{x}_{:j}$ may seem to be a rather crude representation of the document d_j. It tells us how frequently the words appear in the document, but the sequential order of the words is lost. The vector does not attempt to capture the structure in the phrases, sentences, paragraphs, and chapters of the document. However, in spite of this crudeness, search engines work surprisingly well; vectors seem to capture an important aspect of semantics.

The VSM of Salton et al. (1975) was arguably the first practical, useful algorithm for extracting semantic information from word usage. An intuitive justification for the term–document matrix is that the topic of a document will probabilistically influence the author's choice of words when writing the document.[10] If two documents have similar topics, then the two corresponding column vectors will tend to have similar patterns of numbers.

10. Newer generative models, such as Latent Dirichlet Allocation (LDA) (Blei, Ng, & Jordan, 2003), directly model this intuition. See Sections 4.3 and 7.

2.2 Similarity of Words: The Word–Context Matrix

Salton et al. (1975) focused on measuring document similarity, treating a query to a search engine as a pseudo-document. The relevance of a document to a query is given by the similarity of their vectors. Deerwester et al. (1990) observed that we can shift the focus to measuring word similarity, instead of document similarity, by looking at row vectors in the term–document matrix, instead of column vectors.

Deerwester et al. (1990) were inspired by the term–document matrix of Salton et al. (1975), but a document is not necessarily the optimal length of text for measuring word similarity. In general, we may have a *word–context* matrix, in which the context is given by words, phrases, sentences, paragraphs, chapters, documents, or more exotic possibilities, such as sequences of characters or patterns.

The *distributional hypothesis* in linguistics is that words that occur in similar contexts tend to have similar meanings (Harris, 1954). This hypothesis is the justification for applying the VSM to measuring word similarity. A word may be represented by a vector in which the elements are derived from the occurrences of the word in various contexts, such as windows of words (Lund & Burgess, 1996), grammatical dependencies (Lin, 1998; Padó & Lapata, 2007), and richer contexts consisting of dependency links and selectional preferences on the argument positions (Erk & Padó, 2008); see Sahlgren's (2006) thesis for a comprehensive study of various contexts. Similar row vectors in the word–context matrix indicate similar word meanings.

The idea that word usage can reveal semantics was implicit in some of the things that Wittgenstein (1953) said about language-games and family resemblance. Wittgenstein was primarily interested in the physical activities that form the context of word usage (e.g., the word *brick*, spoken in the context of the physical activity of building a house), but the main context for a word is often other words.[11]

Weaver (1955) argued that word sense disambiguation for machine translation should be based on the co-occurrence frequency of the context words near a given target word (the word that we want to disambiguate). Firth (1957, p. 11) said, "You shall know a word by the company it keeps." Deerwester et al. (1990) showed how the intuitions of Wittgenstein (1953), Harris (1954), Weaver, and Firth could be used in a practical algorithm.

2.3 Similarity of Relations: The Pair–Pattern Matrix

In a *pair–pattern* matrix, row vectors correspond to pairs of words, such as *mason* : *stone* and *carpenter* : *wood*, and column vectors correspond to the patterns in which the pairs co-occur, such as "X cuts Y" and "X works with Y". Lin and Pantel (2001) introduced the pair–pattern matrix for the purpose of measuring the semantic similarity of patterns; that is, the similarity of column vectors. Given a pattern such as "X solves Y", their algorithm was able to find similar patterns, such as "Y is solved by X", "Y is resolved in X", and "X resolves Y".

Lin and Pantel (2001) proposed the *extended distributional hypothesis*, that patterns that co-occur with similar pairs tend to have similar meanings. The patterns "X solves Y"

11. Wittgenstein's intuition might be better captured by a matrix that combines words with other modalities, such as images (Monay & Gatica-Perez, 2003). If the values of the elements are derived from event frequencies, we would include this as a VSM approach to semantics.

and "Y is solved by X" tend to co-occur with similar $X:Y$ pairs, which suggests that these patterns have similar meanings. Pattern similarity can be used to infer that one sentence is a paraphrase of another (Lin & Pantel, 2001).

Turney et al. (2003) introduced the use of the pair–pattern matrix for measuring the semantic similarity of relations between word pairs; that is, the similarity of row vectors. For example, the pairs *mason*: *stone*, *carpenter*: *wood*, *potter*: *clay*, and *glassblower*: *glass* share the semantic relation *artisan*: *material*. In each case, the first member of the pair is an artisan who makes artifacts from the material that is the second member of the pair. The pairs tend to co-occur in similar patterns, such as "the X used the Y to" and "the X shaped the Y into".

The *latent relation hypothesis* is that pairs of words that co-occur in similar patterns tend to have similar semantic relations (Turney, 2008a). Word pairs with similar row vectors in a pair–pattern matrix tend to have similar semantic relations. This is the inverse of the extended distributional hypothesis, that patterns with similar column vectors in the pair–pattern matrix tend to have similar meanings.

2.4 Similarities

Pair–pattern matrices are suited to measuring the similarity of semantic relations between pairs of words; that is, *relational similarity*. In contrast, word–context matrices are suited to measuring *attributional similarity*. The distinction between attributional and relational similarity has been explored in depth by Gentner (1983).

The attributional similarity between two words a and b, $\mathrm{sim_a}(a, b) \in \Re$, depends on the degree of correspondence between the properties of a and b. The more correspondence there is, the greater their attributional similarity. The relational similarity between two *pairs* of words $a:b$ and $c:d$, $\mathrm{sim_r}(a:b, c:d) \in \Re$, depends on the degree of correspondence between the relations of $a:b$ and $c:d$. The more correspondence there is, the greater their relational similarity. For example, *dog* and *wolf* have a relatively high degree of attributional similarity, whereas *dog*: *bark* and *cat*: *meow* have a relatively high degree of relational similarity (Turney, 2006).

It is tempting to suppose that relational similarity can be reduced to attributional similarity. For example, *mason* and *carpenter* are similar words and *stone* and *wood* are similar words; therefore, perhaps it follows that *mason*: *stone* and *carpenter*: *wood* have similar relations. Perhaps $\mathrm{sim_r}(a:b, c:d)$ can be reduced to $\mathrm{sim_a}(a, c) + \mathrm{sim_a}(b, d)$. However, *mason*, *carpenter*, *potter*, and *glassblower* are similar words (they are all artisans), as are *wood*, *clay*, *stone*, and *glass* (they are all materials used by artisans), but we cannot infer from this that *mason*: *glass* and *carpenter*: *clay* have similar relations. Turney (2006, 2008a) presented experimental evidence that relational similarity does not reduce to attributional similarity.

The term *semantic relatedness* in computational linguistics (Budanitsky & Hirst, 2001) corresponds to attributional similarity in cognitive science (Gentner, 1983). Two words are semantically related if they have any kind of semantic relation (Budanitsky & Hirst, 2001); they are semantically related to the degree that they share attributes (Turney, 2006). Examples are synonyms (*bank* and *trust company*), meronyms (*car* and *wheel*), antonyms (*hot* and *cold*), and words that are functionally related or frequently associated (*pencil* and *paper*). We might not usually think that antonyms are similar, but antonyms have a high degree of attributional similarity (hot and cold are kinds of temperature, black and white are kinds of colour, loud and quiet are kinds of sound). We prefer the term *attributional similarity* to the term *semantic relatedness*, because *attributional similarity* emphasizes the contrast with *relational similarity*, whereas *semantic relatedness* could be confused with *relational similarity*.

In computational linguistics, the term *semantic similarity* is applied to words that share a hypernym (*car* and *bicycle* are semantically similar, because they share the hypernym *vehicle*) (Resnik, 1995). Semantic similarity is a specific type of attributional similarity. We prefer the term *taxonomical similarity* to the term *semantic similarity*, because the term *semantic similarity* is misleading. Intuitively, both attributional and relational similarity involve meaning, so both deserve to be called semantic similarity.

Words are *semantically associated* if they tend to co-occur frequently (e.g., *bee* and *honey*) (Chiarello, Burgess, Richards, & Pollock, 1990). Words may be taxonomically similar and semantically associated (*doctor* and *nurse*), taxonomically similar but not semantically associated (*horse* and *platypus*), semantically associated but not taxonomically similar (*cradle* and *baby*), or neither semantically associated nor taxonomically similar (*calculus* and *candy*).

Schütze and Pedersen (1993) defined two ways that words can be distributed in a corpus of text: If two words tend to be neighbours of each other, then they are *syntagmatic associates*. If two words have similar neighbours, then they are *paradigmatic parallels*. Syntagmatic associates are often different parts of speech, whereas paradigmatic parallels are usually the same part of speech. Syntagmatic associates tend to be semantically associated (*bee* and *honey* are often neighbours); paradigmatic parallels tend to be taxonomically similar (*doctor* and *nurse* have similar neighbours).

2.5 Other Semantic VSMs

The possibilities are not exhausted by term–document, word–context, and pair–pattern matrices. We might want to consider triple–pattern matrices, for measuring the semantic similarity between word triples. Whereas a pair–pattern matrix might have a row *mason* : *stone* and a column "X works with Y", a triple–pattern matrix could have a row *mason* : *stone* : *masonry* and a column "X uses Y to build Z". However, n-tuples of words grow increasingly rare as n increases. For example, phrases that contain *mason*, *stone*, and *masonry* together are less frequent than phrases that contain *mason* and *stone* together. A triple–pattern matrix will be much more sparse than a pair–pattern matrix (ceteris paribus). The quantity of text that we need, in order to have enough numbers to make our matrices useful, grows rapidly as n increases. It may be better to break n-tuples into pairs. For example, $a\!:\!b\!:\!c$ could be decomposed into $a\!:\!b$, $a\!:\!c$, and $b\!:\!c$ (Turney, 2008a). The similarity of two triples, $a\!:\!b\!:\!c$ and $d\!:\!e\!:\!f$, could be estimated by the similarity of their corresponding pairs. A relatively dense pair–pattern matrix could serve as a surrogate for a relatively sparse triple–pattern matrix.

We may also go beyond matrices. The generalization of a matrix is a tensor (Kolda & Bader, 2009; Acar & Yener, 2009). A scalar (a single number) is zeroth-order tensor, a vector is first-order tensor, and a matrix is a second-order tensor. A tensor of order three or higher is called a higher-order tensor. Chew, Bader, Kolda, and Abdelali (2007) use a term–document–language third-order tensor for multilingual information retrieval. Turney (2007) uses a word–word–pattern tensor to measure similarity of words. Van de Cruys (2009) uses a verb–subject–object tensor to learn selectional preferences of verbs.

In Turney's (2007) tensor, for example, rows correspond to words from the TOEFL multiple-choice synonym questions, columns correspond to words from Basic English (Ogden, 1930),[12] and *tubes* correspond to patterns that join rows and columns (hence we have a word–word–pattern third-order tensor). A given word from the TOEFL questions is represented by the corresponding word–pattern matrix *slice* in the tensor. The elements in this slice correspond to all the patterns that relate the given TOEFL word to any word in Basic English. The similarity of two TOEFL words is calculated by comparing the two corresponding matrix slices. The algorithm achieves 83.8% on the TOEFL questions.

2.6 Types and Tokens

A *token* is a single instance of a symbol, whereas a *type* is a general class of tokens (Manning et al., 2008). Consider the following example (from Samuel Beckett):

> Ever tried. Ever failed.
> No matter. Try again.
> Fail again. Fail better.

There are two tokens of the type *Ever*, two tokens of the type *again*, and two tokens of the type *Fail*. Let's say that each line in this example is a document, so we have three documents of two sentences each. We can represent this example with a token–document matrix or a type–document matrix. The token–document matrix has twelve rows, one for each token, and three columns, one for each line (Figure 1). The type–document matrix has nine rows, one for each type, and three columns (Figure 2).

A row vector for a token has binary values: an element is 1 if the given token appears in the given document and 0 otherwise. A row vector for a type has integer values: an element is the frequency of the given type in the given document. These vectors are related, in that a type vector is the sum of the corresponding token vectors. For example, the row vector for the type *Ever* is the sum of the two token vectors for the two tokens of *Ever*.

In applications dealing with polysemy, one approach uses vectors that represent word tokens (Schütze, 1998; Agirre & Edmonds, 2006) and another uses vectors that represent word types (Pantel & Lin, 2002a). Typical word sense disambiguation (WSD) algorithms deal with word tokens (instances of words in specific contexts) rather than word types. We mention both approaches to polysemy in Section 6, due to their similarity and close relationship, although a defining characteristic of the VSM is that it is concerned with frequencies (see Section 1.1), and frequency is a property of types, not tokens.

12. Basic English is a highly reduced subset of English, designed to be easy for people to learn. The words of Basic English are listed at http://ogden.basic-english.org/.

	Ever tried. Ever failed.	No matter. Try again.	Fail again. Fail better.
Ever	1	0	0
tried	1	0	0
Ever	1	0	0
failed	1	0	0
No	0	1	0
matter	0	1	0
Try	0	1	0
again	0	1	0
Fail	0	0	1
again	0	0	1
Fail	0	0	1
better	0	0	1

Figure 1: The token–document matrix. Rows are tokens and columns are documents.

	Ever tried. Ever failed.	No matter. Try again.	Fail again. Fail better.
Ever	2	0	0
tried	1	0	0
failed	1	0	0
No	0	1	0
matter	0	1	0
Try	0	1	0
again	0	1	1
Fail	0	0	2
better	0	0	1

Figure 2: The type–document matrix. Rows are types and columns are documents.

2.7 Hypotheses

We have mentioned five hypotheses in this section. Here we repeat these hypotheses and then interpret them in terms of vectors. For each hypothesis, we cite work that explicitly states something like the hypothesis or implicitly assumes something like the hypothesis.

Statistical semantics hypothesis: Statistical patterns of human word usage can be used to figure out what people mean (Weaver, 1955; Furnas et al., 1983). – If units of text have similar vectors in a text frequency matrix,[13] then they tend to have similar meanings. (We take this to be a general hypothesis that subsumes the four more specific hypotheses that follow.)

Bag of words hypothesis: The frequencies of words in a document tend to indicate the relevance of the document to a query (Salton et al., 1975). – If documents and pseudo-documents (queries) have similar column vectors in a term–document matrix, then they tend to have similar meanings.

Distributional hypothesis: Words that occur in similar contexts tend to have similar meanings (Harris, 1954; Firth, 1957; Deerwester et al., 1990). – If words have similar row vectors in a word–context matrix, then they tend to have similar meanings.

Extended distributional hypothesis: Patterns that co-occur with similar pairs tend to have similar meanings (Lin & Pantel, 2001). – If patterns have similar column vectors in a pair–pattern matrix, then they tend to express similar semantic relations.

Latent relation hypothesis: Pairs of words that co-occur in similar patterns tend to have similar semantic relations (Turney et al., 2003). – If word pairs have similar row vectors in a pair–pattern matrix, then they tend to have similar semantic relations.

We have not yet explained what it means to say that vectors are similar. We discuss this in Section 4.4.

3. Linguistic Processing for Vector Space Models

We will assume that our raw data is a large corpus of natural language text. Before we generate a term–document, word–context, or pair–pattern matrix, it can be useful to apply some linguistic processing to the raw text. The types of processing that are used can be grouped into three classes. First, we need to *tokenize* the raw text; that is, we need to decide what constitutes a *term* and how to extract terms from raw text. Second, we may want to *normalize* the raw text, to convert superficially different strings of characters to the same form (e.g., *car*, *Car*, *cars*, and *Cars* could all be normalized to *car*). Third, we may want to *annotate* the raw text, to mark identical strings of characters as being different (e.g., *fly* as a verb could be annotated as *fly/VB* and *fly* as a noun could be annotated as *fly/NN*).

Grefenstette (1994) presents a good study of linguistic processing for word–context VSMs. He uses a similar three-step decomposition of linguistic processing: tokenization, surface syntactic analysis, and syntactic attribute extraction.

13. By *text frequency matrix*, we mean any matrix or higher-order tensor in which the values of the elements are derived from the frequencies of pieces of text in the context of other pieces of text in some collection of text. A text frequency matrix is intended to be a general structure, which includes term–document, word–context, and pair–pattern matrices as special cases.

3.1 Tokenization

Tokenization of English seems simple at first glance: words are separated by spaces. This assumption is approximately true for English, and it may work sufficiently well for a basic VSM, but a more advanced VSM requires a more sophisticated approach to tokenization.

An accurate English tokenizer must know how to handle punctuation (e.g., *don't*, *Jane's*, *and/or*), hyphenation (e.g., *state-of-the-art* versus *state of the art*), and recognize multi-word terms (e.g., *Barack Obama* and *ice hockey*) (Manning et al., 2008). We may also wish to ignore *stop words*, high-frequency words with relatively low information content, such as function words (e.g., *of*, *the*, *and*) and pronouns (e.g., *them*, *who*, *that*). A popular list of stop words is the set of 571 common words included in the source code for the SMART system (Salton, 1971).[14]

In some languages (e.g., Chinese), words are not separated by spaces. A basic VSM can break the text into character unigrams or bigrams. A more sophisticated approach is to match the input text against entries in a lexicon, but the matching often does not determine a unique tokenization (Sproat & Emerson, 2003). Furthermore, native speakers often disagree about the correct segmentation. Highly accurate tokenization is a challenging task for most human languages.

3.2 Normalization

The motivation for normalization is the observation that many different strings of characters often convey essentially identical meanings. Given that we want to get at the meaning that underlies the words, it seems reasonable to normalize superficial variations by converting them to the same form. The most common types of normalization are case folding (converting all words to lower case) and stemming (reducing inflected words to their stem or root form).

Case folding is easy in English, but can be problematic in some languages. In French, accents are optional for uppercase, and it may be difficult to restore missing accents when converting the words to lowercase. Some words cannot be distinguished without accents; for example, *PECHE* could be either *pêche* (meaning *fishing* or *peach*) or *péché* (meaning *sin*). Even in English, case folding can cause problems, because case sometimes has semantic significance. For example, *SMART* is an information retrieval system, whereas *smart* is a common adjective; *Bush* may be a surname, whereas *bush* is a kind of plant.

Morphology is the study of the internal structure of words. Often a word is composed of a stem (root) with added affixes (inflections), such as plural forms and past tenses (e.g., *trapped* is composed of the stem *trap* and the affix *-ed*). Stemming, a kind of morphological analysis, is the process of reducing inflected words to their stems. In English, affixes are simpler and more regular than in many other languages, and stemming algorithms based on heuristics (rules of thumb) work relatively well (Lovins, 1968; Porter, 1980; Minnen, Carroll, & Pearce, 2001). In an *agglutinative* language (e.g., Inuktitut), many concepts are combined into a single word, using various prefixes, infixes, and suffixes, and morphological analysis is complicated. A single word in an agglutinative language may correspond to a sentence of half a dozen words in English (Johnson & Martin, 2003).

14. The source code is available at ftp://ftp.cs.cornell.edu/pub/smart/.

The performance of an information retrieval system is often measured by precision and recall (Manning et al., 2008). The *precision* of a system is an estimate of the conditional probability that a document is truly relevant to a query, if the system says it is relevant. The *recall* of a system is an estimate of the conditional probability that the system will say that a document is relevant to a query, if it truly is relevant.

In general, normalization increases recall and reduces precision (Kraaij & Pohlmann, 1996). This is natural, given the nature of normalization. When we remove superficial variations that we believe are irrelevant to meaning, we make it easier to recognize similarities; we find more similar things, and so recall increases. But sometimes these superficial variations have semantic significance; ignoring the variations causes errors, and so precision decreases. Normalization can also have a positive effect on precision in cases where variant tokens are infrequent and smoothing the variations gives more reliable statistics.

If we have a small corpus, we may not be able to afford to be overly selective, and it may be best to aggressively normalize the text, to increase recall. If we have a very large corpus, precision may be more important, and we might not want any normalization. Hull (1996) gives a good analysis of normalization for information retrieval.

3.3 Annotation

Annotation is the inverse of normalization. Just as different strings of characters may have the same meaning, it also happens that identical strings of characters may have different meanings, depending on the context. Common forms of annotation include part-of-speech tagging (marking words according to their parts of speech), word sense tagging (marking ambiguous words according to their intended meanings), and parsing (analyzing the grammatical structure of sentences and marking the words in the sentences according to their grammatical roles) (Manning & Schütze, 1999).

Since annotation is the inverse of normalization, we expect it to decrease recall and increase precision. For example, by tagging *program* as a noun or a verb, we may be able to selectively search for documents that are about the act of computer programming (verb) instead of documents that discuss particular computer programs (noun); hence we can increase precision. However, a document about computer programs (noun) may have something useful to say about the act of computer programming (verb), even if the document never uses the verb form of *program*; hence we may decrease recall.

Large gains in IR performance have recently been reported as a result of query annotation with syntactic and semantic information. Syntactic annotation includes query segmentation (Tan & Peng, 2008) and part of speech tagging (Barr, Jones, & Regelson, 2008). Examples of semantic annotation are disambiguating abbreviations in queries (Wei, Peng, & Dumoulin, 2008) and finding query keyword associations (Lavrenko & Croft, 2001; Cao, Nie, & Bai, 2005).

Annotation is also useful for measuring the semantic similarity of words and concepts (word–context matrices). For example, Pantel and Lin (2002a) presented an algorithm that can discover word senses by clustering row vectors in a word–context matrix, using contextual information derived from parsing.

4. Mathematical Processing for Vector Space Models

After the text has been tokenized and (optionally) normalized and annotated, the first step is to generate a matrix of frequencies. Second, we may want to adjust the weights of the elements in the matrix, because common words will have high frequencies, yet they are less informative than rare words. Third, we may want to smooth the matrix, to reduce the amount of random noise and to fill in some of the zero elements in a sparse matrix. Fourth, there are many different ways to measure the similarity of two vectors.

Lowe (2001) gives a good summary of mathematical processing for word–context VSMs. He decomposes VSM construction into a similar four-step process: calculate the frequencies, transform the raw frequency counts, smooth the space (dimensionality reduction), then calculate the similarities.

4.1 Building the Frequency Matrix

An element in a frequency matrix corresponds to an *event*: a certain item (term, word, word pair) occurred in a certain situation (document, context, pattern) a certain number of times (frequency). At an abstract level, building a frequency matrix is a simple matter of counting events. In practice, it can be complicated when the corpus is large.

A typical approach to building a frequency matrix involves two steps. First, scan sequentially through the corpus, recording events and their frequencies in a hash table, a database, or a search engine index. Second, use the resulting data structure to generate the frequency matrix, with a sparse matrix representation (Gilbert, Moler, & Schreiber, 1992).

4.2 Weighting the Elements

The idea of weighting is to give more weight to surprising events and less weight to expected events. The hypothesis is that surprising events, if shared by two vectors, are more discriminative of the similarity between the vectors than less surprising events. For example, in measuring the semantic similarity between the words *mouse* and *rat*, the contexts *dissect* and *exterminate* are more discriminative of their similarity than the contexts *have* and *like*. In information theory, a surprising event has higher information content than an expected event (Shannon, 1948). The most popular way to formalize this idea for term–document matrices is the *tf-idf* (term frequency $\times$ inverse document frequency) family of weighting functions (Spärck Jones, 1972). An element gets a high weight when the corresponding term is frequent in the corresponding document (i.e., tf is high), but the term is rare in other documents in the corpus (i.e., df is low, and thus idf is high). Salton and Buckley (1988) defined a large family of tf-idf weighting functions and evaluated them on information retrieval tasks, demonstrating that tf-idf weighting can yield significant improvements over raw frequency.

Another kind of weighting, often combined with tf-idf weighting, is length normalization (Singhal, Salton, Mitra, & Buckley, 1996). In information retrieval, if document length is ignored, search engines tend to have a bias in favour of longer documents. Length normalization corrects for this bias.

Term weighting may also be used to correct for correlated terms. For example, the terms *hostage* and *hostages* tend to be correlated, yet we may not want to normalize them to the same term (as in Section 3.2), because they have slightly different meanings. As an alternative to normalizing them, we may reduce their weights when they co-occur in a document (Church, 1995).

Feature selection may be viewed as a form of weighting, in which some terms get a weight of zero and hence can be removed from the matrix. Forman (2003) provides a good study of feature selection methods for text classification.

An alternative to tf-idf is Pointwise Mutual Information (PMI) (Church & Hanks, 1989; Turney, 2001), which works well for both word–context matrices (Pantel & Lin, 2002a) and term–document matrices (Pantel & Lin, 2002b). A variation of PMI is Positive PMI (PPMI), in which all PMI values that are less than zero are replaced with zero (Niwa & Nitta, 1994). Bullinaria and Levy (2007) demonstrated that PPMI performs better than a wide variety of other weighting approaches when measuring semantic similarity with word–context matrices. Turney (2008a) applied PPMI to pair–pattern matrices. We will give the formal definition of PPMI here, as an example of an effective weighting function.

Let $\mathbf{F}$ be a word–context frequency matrix with n_r rows and n_c columns. The i-th row in $\mathbf{F}$ is the row vector $\mathbf{f}_{i:}$ and the j-th column in $\mathbf{F}$ is the column vector $\mathbf{f}_{:j}$. The row $\mathbf{f}_{i:}$ corresponds to a word w_i and the column $\mathbf{f}_{:j}$ corresponds to a context c_j. The value of the element f_{ij} is the number of times that w_i occurs in the context c_j. Let $\mathbf{X}$ be the matrix that results when PPMI is applied to $\mathbf{F}$. The new matrix $\mathbf{X}$ has the same number of rows and columns as the raw frequency matrix $\mathbf{F}$. The value of an element x_{ij} in $\mathbf{X}$ is defined as follows:

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} f_{ij}} \tag{1}$$

$$p_{i*} = \frac{\sum_{j=1}^{n_c} f_{ij}}{\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} f_{ij}} \tag{2}$$

$$p_{*j} = \frac{\sum_{i=1}^{n_r} f_{ij}}{\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} f_{ij}} \tag{3}$$

$$\mathrm{pmi}_{ij} = \log \left(\frac{p_{ij}}{p_{i*} p_{*j}} \right) \tag{4}$$

$$x_{ij} = \begin{cases} \mathrm{pmi}_{ij} & \text{if } \mathrm{pmi}_{ij} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

In this definition, p_{ij} is the estimated probability that the word w_i occurs in the context c_j, p_{i*} is the estimated probability of the word w_i, and p_{*j} is the estimated probability of the context c_j. If w_i and c_j are statistically independent, then $p_{i*} p_{*j} = p_{ij}$ (by the definition of independence), and thus pmi_{ij} is zero (since $\log(1) = 0$). The product $p_{i*} p_{*j}$ is what we would expect for p_{ij} if w_i occurs in c_j by pure random chance. On the other hand, if there is an interesting semantic relation between w_i and c_j, then we should expect p_{ij} to be larger than it would be if w_i and c_j were indepedent; hence we should find that $p_{ij} > p_{i*} p_{*j}$, and thus pmi_{ij} is positive. This follows from the distributional hypothesis (see Section 2). If the word w_i is unrelated to the context c_j, we may find that pmi_{ij} is negative. PPMI is designed to give a high value to x_{ij} when there is an interesting semantic relation between

w_i and c_j; otherwise, x_{ij} should have a value of zero, indicating that the occurrence of w_i in c_j is uninformative.

A well-known problem of PMI is that it is biased towards infrequent events. Consider the case where w_i and c_j are statistically dependent (i.e., they have maximum association). Then $p_{ij} = p_{i*} = p_{*j}$. Hence (4) becomes $\log(1/p_{i*})$ and PMI increases as the probability of word w_i decreases. Several discounting factors have been proposed to alleviate this problem. An example follows (Pantel & Lin, 2002a):

$$\delta_{ij} = \frac{f_{ij}}{f_{ij}+1} \cdot \frac{\min\left(\sum_{k=1}^{n_r} f_{kj}, \sum_{k=1}^{n_c} f_{ik}\right)}{\min\left(\sum_{k=1}^{n_r} f_{kj}, \sum_{k=1}^{n_c} f_{ik}\right)+1} \tag{6}$$

$$\text{newpmi}_{ij} = \delta_{ij} \cdot \text{pmi}_{ij} \tag{7}$$

Another way to deal with infrequent events is Laplace smoothing of the probability estimates, p_{ij}, p_{i*}, and p_{*j} (Turney & Littman, 2003). A constant positive value is added to the raw frequencies before calculating the probabilities; each f_{ij} is replaced with $f_{ij} + k$, for some $k > 0$. The larger the constant, the greater the smoothing effect. Laplace smoothing pushes the pmi_{ij} values towards zero. The magnitude of the push (the difference between pmi_{ij} with and without Laplace smoothing) depends on the raw frequency f_{ij}. If the frequency is large, the push is small; if the frequency is small, the push is large. Thus Laplace smoothing reduces the bias of PMI towards infrequent events.

4.3 Smoothing the Matrix

The simplest way to improve information retrieval performance is to limit the number of vector components. Keeping only components representing the most frequently occurring content words is such a way; however, common words, such as *the* and *have*, carry little semantic discrimination power. Simple component smoothing heuristics, based on the properties of the weighting schemes presented in Section 4.2, have been shown to both maintain semantic discrimination power and improve the performance of similarity computations.

Computing the similarity between all pairs of vectors, described in Section 4.4, is a computationally intensive task. However, only vectors that share a non-zero coordinate must be compared (i.e., two vectors that do not share a coordinate are dissimilar). Very frequent context words, such as the word *the*, unfortunately result in most vectors matching a non-zero coordinate. Such words are precisely the contexts that have little semantic discrimination power. Consider the pointwise mutual information weighting described in Section 4.2. Highly weighted dimensions co-occur frequently with only very few words and are by definition highly discriminating contexts (i.e., they have very high association with the words with which they co-occur). By keeping only the context-word dimensions with a PMI above a conservative threshold and setting the others to zero, Lin (1998) showed that the number of comparisons needed to compare vectors greatly decreases while losing little precision in the similarity score between the top-200 most similar words of every word. While smoothing the matrix, one computes a reverse index on the non-zero coordinates. Then, to compare the similarity between a word's context vector and all other words' context vectors, only those vectors found to match a non-zero component in the reverse index must be compared. Section 4.5 proposes further optimizations along these lines.

Deerwester et al. (1990) found an elegant way to improve similarity measurements with a mathematical operation on the term–document matrix, $\mathbf{X}$, based on linear algebra. The operation is truncated Singular Value Decomposition (SVD), also called thin SVD. Deerwester et al. briefly mentioned that truncated SVD can be applied to both document similarity and word similarity, but their focus was document similarity. Landauer and Dumais (1997) applied truncated SVD to word similarity, achieving human-level scores on multiple-choice synonym questions from the Test of English as a Foreign Language (TOEFL). Truncated SVD applied to document similarity is called Latent Semantic Indexing (LSI), but it is called Latent Semantic Analysis (LSA) when applied to word similarity.

There are several ways of thinking about how truncated SVD works. We will first present the math behind truncated SVD and then describe four ways of looking at it: latent meaning, noise reduction, high-order co-occurrence, and sparsity reduction.

SVD decomposes $\mathbf{X}$ into the product of three matrices $\mathbf{U\Sigma V}^\mathsf{T}$, where $\mathbf{U}$ and $\mathbf{V}$ are in column orthonormal form (i.e., the columns are orthogonal and have unit length, $\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{V}^\mathsf{T}\mathbf{V} = \mathbf{I}$) and $\mathbf{\Sigma}$ is a diagonal matrix of singular values (Golub & Van Loan, 1996). If $\mathbf{X}$ is of rank r, then $\mathbf{\Sigma}$ is also of rank r. Let $\mathbf{\Sigma}_k$, where $k < r$, be the diagonal matrix formed from the top k singular values, and let $\mathbf{U}_k$ and $\mathbf{V}_k$ be the matrices produced by selecting the corresponding columns from $\mathbf{U}$ and $\mathbf{V}$. The matrix $\mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\mathsf{T}$ is the matrix of rank k that best approximates the original matrix $\mathbf{X}$, in the sense that it minimizes the approximation errors. That is, $\hat{\mathbf{X}} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\mathsf{T}$ minimizes $\|\hat{\mathbf{X}} - \mathbf{X}\|_F$ over all matrices $\hat{\mathbf{X}}$ of rank k, where $\| \ldots \|_F$ denotes the Frobenius norm (Golub & Van Loan, 1996).

Latent meaning: Deerwester et al. (1990) and Landauer and Dumais (1997) describe truncated SVD as a method for discovering latent meaning. Suppose we have a word–context matrix $\mathbf{X}$. The truncated SVD, $\hat{\mathbf{X}} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\mathsf{T}$, creates a low-dimensional linear mapping between row space (words) and column space (contexts). This low-dimensional mapping captures the latent (hidden) meaning in the words and the contexts. Limiting the number of latent dimensions ($k < r$) forces a greater correspondence between words and contexts. This forced correspondence between words and contexts improves the similarity measurement.

Noise reduction: Rapp (2003) describes truncated SVD as a noise reduction technique. We may think of the matrix $\hat{\mathbf{X}} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\mathsf{T}$ as a smoothed version of the original matrix $\mathbf{X}$. The matrix $\mathbf{U}_k$ maps the row space (the space spanned by the rows of $\mathbf{X}$) into a smaller k-dimensional space and the matrix $\mathbf{V}_k$ maps the column space (the space spanned by the columns of $\mathbf{X}$) into the same k-dimensional space. The diagonal matrix $\mathbf{\Sigma}_k$ specifies the weights in this reduced k-dimensional space. The singular values in $\mathbf{\Sigma}$ are ranked in descending order of the amount of variation in $\mathbf{X}$ that they fit. If we think of the matrix $\mathbf{X}$ as being composed of a mixture of signal and noise, with more signal than noise, then $\mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\mathsf{T}$ mostly captures the variation in $\mathbf{X}$ that is due to the signal, whereas the remaining vectors in $\mathbf{U\Sigma V}^\mathsf{T}$ are mostly fitting the variation in $\mathbf{X}$ that is due to the noise.

High-order co-occurrence: Landauer and Dumais (1997) also describe truncated SVD as a method for discovering high-order co-occurrence. Direct co-occurrence (first-order co-occurrence) is when two words appear in identical contexts. Indirect co-occurrence (high-order co-occurrence) is when two words appear in *similar* contexts. Similarity of contexts may be defined recursively in terms of lower-order co-occurrence. Lemaire and Denhière (2006) demonstrate that truncated SVD can discover high-order co-occurrence.

Sparsity reduction: In general, the matrix $\mathbf{X}$ is very sparse (mostly zeroes), but the truncated SVD, $\hat{\mathbf{X}} = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^{\mathsf{T}}$, is dense. Sparsity may be viewed as a problem of insufficient data: with more text, the matrix $\mathbf{X}$ would have fewer zeroes, and the VSM would perform better on the chosen task. From this perspective, truncated SVD is a way of simulating the missing text, compensating for the lack of data (Vozalis & Margaritis, 2003).

These different ways of viewing truncated SVD are compatible with each other; it is possible for all of these perspectives to be correct. Future work is likely to provide more views of SVD and perhaps a unifying view.

A good C implementation of SVD for large sparse matrices is Rohde's SVDLIBC.[15] Another approach is Brand's (2006) incremental truncated SVD algorithm.[16] Yet another approach is Gorrell's (2006) Hebbian algorithm for incremental truncated SVD. Brand's and Gorrell's algorithms both introduce interesting new ways of handling missing values, instead of treating them as zero values.

For higher-order tensors, there are operations that are analogous to truncated SVD, such as parallel factor analysis (PARAFAC) (Harshman, 1970), canonical decomposition (CANDECOMP) (Carroll & Chang, 1970) (equivalent to PARAFAC but discovered independently), and Tucker decomposition (Tucker, 1966). For an overview of tensor decompositions, see the surveys of Kolda and Bader (2009) or Acar and Yener (2009). Turney (2007) gives an empirical evaluation of how well four different Tucker decomposition algorithms scale up for large sparse third-order tensors. A low-RAM algorithm, Multislice Projection, for large sparse tensors is presented and evaluated.[17]

Since the work of Deerwester et al. (1990), subsequent research has discovered many alternative matrix smoothing processes, such as Nonnegative Matrix Factorization (NMF) (Lee & Seung, 1999), Probabilistic Latent Semantic Indexing (PLSI) (Hofmann, 1999), Iterative Scaling (IS) (Ando, 2000), Kernel Principal Components Analysis (KPCA) (Scholkopf, Smola, & Muller, 1997), Latent Dirichlet Allocation (LDA) (Blei et al., 2003), and Discrete Component Analysis (DCA) (Buntine & Jakulin, 2006).

The four perspectives on truncated SVD, presented above, apply equally well to all of these more recent matrix smoothing algorithms. These newer smoothing algorithms tend to be more computationally intensive than truncated SVD, but they attempt to model word frequencies better than SVD. Truncated SVD implicitly assumes that the elements in $\mathbf{X}$ have a Gaussian distribution. Minimizing the the Frobenius norm $\|\hat{\mathbf{X}} - \mathbf{X}\|_F$ will minimize the noise, if the noise has a Gaussian distribution. However, it is known that word frequencies do not have Gaussian distributions. More recent algorithms are based on more realistic models of the distribution for word frequencies.[18]

4.4 Comparing the Vectors

The most popular way to measure the similarity of two frequency vectors (raw or weighted) is to take their cosine. Let $\mathbf{x}$ and $\mathbf{y}$ be two vectors, each with n elements.

15. SVDLIBC is available at http://tedlab.mit.edu/~dr/svdlibc/.
16. MATLAB source code is available at http://web.mit.edu/~wingated/www/resources.html.
17. MATLAB source code is available at http://www.apperceptual.com/multislice/.
18. In our experience, pmi_{ij} appears to be approximately Gaussian, which may explain why PMI works well with truncated SVD, but then PPMI is puzzling, because it is less Gaussian than PMI, yet it apparently yields better semantic models than PMI.

$$\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle \tag{8}$$

$$\mathbf{y} = \langle y_1, y_2, \ldots, y_n \rangle \tag{9}$$

The cosine of the angle θ between $\mathbf{x}$ and $\mathbf{y}$ can be calculated as follows:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \cdot \sum_{i=1}^{n} y_i^2}} \tag{10}$$

$$= \frac{\mathbf{x} \cdot \mathbf{y}}{\sqrt{\mathbf{x} \cdot \mathbf{x}} \cdot \sqrt{\mathbf{y} \cdot \mathbf{y}}} \tag{11}$$

$$= \frac{\mathbf{x}}{\|\mathbf{x}\|} \cdot \frac{\mathbf{y}}{\|\mathbf{y}\|} \tag{12}$$

In other words, the cosine of the angle between two vectors is the inner product of the vectors, after they have been normalized to unit length. If $\mathbf{x}$ and $\mathbf{y}$ are frequency vectors for words, a frequent word will have a long vector and a rare word will have a short vector, yet the words might be synonyms. Cosine captures the idea that the length of the vectors is irrelevant; the important thing is the angle between the vectors.

The cosine ranges from -1 when the vectors point in opposite directions (θ is 180 degrees) to $+1$ when they point in the same direction (θ is 0 degrees). When the vectors are orthogonal (θ is 90 degrees), the cosine is zero. With raw frequency vectors, which necessarily cannot have negative elements, the cosine cannot be negative, but weighting and smoothing often introduce negative elements. PPMI weighting does not yield negative elements, but truncated SVD can generate negative elements, even when the input matrix has no negative values.

A measure of distance between vectors can easily be converted to a measure of similarity by inversion (13) or subtraction (14).

$$\mathrm{sim}(\mathbf{x}, \mathbf{y}) = 1/\mathrm{dist}(\mathbf{x}, \mathbf{y}) \tag{13}$$

$$\mathrm{sim}(\mathbf{x}, \mathbf{y}) = 1 - \mathrm{dist}(\mathbf{x}, \mathbf{y}) \tag{14}$$

Many similarity measures have been proposed in both IR (Jones & Furnas, 1987) and lexical semantics circles (Lin, 1998; Dagan, Lee, & Pereira, 1999; Lee, 1999; Weeds, Weir, & McCarthy, 2004). It is commonly said in IR that, properly normalized, the difference in retrieval performance using different measures is insignificant (van Rijsbergen, 1979). Often the vectors are normalized in some way (e.g., unit length or unit probability) before applying any similarity measure.

Popular geometric measures of vector distance include Euclidean distance and Manhattan distance. Distance measures from information theory include Hellinger, Bhattacharya, and Kullback-Leibler. Bullinaria and Levy (2007) compared these five distance measures and the cosine similarity measure on four different tasks involving word similarity. Overall, the best measure was cosine. Other popular measures are the Dice and Jaccard coefficients (Manning et al., 2008).

Lee (1999) proposed that, for finding word similarities, measures that focused more on overlapping coordinates and less on the importance of negative features (i.e., coordinates where one word has a nonzero value and the other has a zero value) appear to perform better. In Lee's experiments, the Jaccard, Jensen-Shannon, and L1 measures seemed to perform best. Weeds et al. (2004) studied the linguistic and statistical properties of the similar words returned by various similarity measures and found that the measures can be grouped into three classes:

1. high-frequency sensitive measures (cosine, Jensen-Shannon, α-skew, recall),
2. low-frequency sensitive measures (precision), and
3. similar-frequency sensitive methods (Jaccard, Jaccard+MI, Lin, harmonic mean).

Given a word w_0, if we use a high-frequency sensitive measure to score other words w_i according to their similarity with w_0, higher frequency words will tend to get higher scores than lower frequency words. If we use a low-frequency sensitive measure, there will be a bias towards lower frequency words. Similar-frequency sensitive methods prefer a word w_i that has approximately the same frequency as w_0. In one experiment on determining the compositionality of collocations, high-frequency sensitive measures outperformed the other classes (Weeds et al., 2004). We believe that determining the most appropriate similarity measure is inherently dependent on the similarity task, the sparsity of the statistics, the frequency distribution of the elements being compared, and the smoothing method applied to the matrix.

4.5 Efficient Comparisons

Computing the similarity between all rows (or columns) in a large matrix is a non-trivial problem, with a worst case cubic running time $O(n_r^2 n_c)$, where n_r is the number of rows and n_c is the number of columns (i.e., the dimensionality of the feature space). Optimizations and parallelization are often necessary.

4.5.1 Sparse-Matrix Multiplication

One optimization strategy is a generalized sparse-matrix multiplication approach (Sarawagi & Kirpal, 2004), which is based on the observation that a scalar product of two vectors depends only on the coordinates for which both vectors have nonzero values. Further, we observe that most commonly used similarity measures for vectors $\mathbf{x}$ and $\mathbf{y}$, such as cosine, overlap, and Dice, can be decomposed into three values: one depending only on the nonzero values of $\mathbf{x}$, another depending only on the nonzero values of $\mathbf{y}$, and the third depending on the nonzero coordinates shared both by $\mathbf{x}$ and $\mathbf{y}$. More formally, commonly used similarity scores, $\mathrm{sim}(\mathbf{x}, \mathbf{y})$, can be expressed as follows:

$$\mathrm{sim}(\mathbf{x}, \mathbf{y}) = f_0\left(\textstyle\sum_{i=1}^{n} f_1(x_i, y_i), f_2(\mathbf{x}), f_3(\mathbf{y})\right) \tag{15}$$

For example, the cosine measure, $\cos(\mathbf{x}, \mathbf{y})$, defined in (10), can be expressed in this model as follows:

$$\cos(\mathbf{x}, \mathbf{y}) = f_0\left(\textstyle\sum_{i=1}^{n} f_1(x_i, y_i), f_2(\mathbf{x}), f_3(\mathbf{y})\right) \tag{16}$$

$$f_0(a, b, c) = \frac{a}{b \cdot c} \tag{17}$$

$$f_1(a, b) = a \cdot b \tag{18}$$

$$f_2(\mathbf{a}) = f_3(\mathbf{a}) = \sqrt{\textstyle\sum_{i=1}^{n} a_i^2} \tag{19}$$

Let $\mathbf{X}$ be a matrix for which we want to compute the pairwise similarity, $\text{sim}(\mathbf{x}, \mathbf{y})$, between all rows or all columns $\mathbf{x}$ and $\mathbf{y}$. Efficient computation of the similarity matrix $\mathbf{S}$ can be achieved by leveraging the fact that $\text{sim}(\mathbf{x}, \mathbf{y})$ is determined solely by the nonzero coordinates shared by $\mathbf{x}$ and $\mathbf{y}$ (i.e., $f_1(0, x_i) = f_1(x_i, 0) = 0$ for any x_i) and that most of the vectors are very sparse. In this case, calculating $f_1(x_i, y_i)$ is only required when both vectors have a shared nonzero coordinate, significantly reducing the cost of computation. Determining which vectors share a nonzero coodinate can easily be achieved by first building an inverted index for the coordinates. During indexing, we can also precompute $f_2(\mathbf{x})$ and $f_3(\mathbf{y})$ without changing the algorithm complexity. Then, for each vector $\mathbf{x}$ we retrieve in constant time, from the index, each vector $\mathbf{y}$ that shares a nonzero coordinate with $\mathbf{x}$ and we apply $f_1(x_i, y_i)$ on the shared coordinates i. The computational cost of this algorithm is $\sum_i N_i^2$ where N_i is the number of vectors that have a nonzero i-th coordinate. Its worst case time complexity is $O(ncv)$ where n is the number of vectors to be compared, c is the maximum number of nonzero coordinates of any vector, and v is the number of vectors that have a nonzero i-th coordinate where i is the coordinate which is nonzero for the most vectors. In other words, the algorithm is efficient only when the density of the coordinates is low. In our own experiments of computing the semantic similarity between all pairs of words in a large web crawl, we observed near linear average running time complexity in n.

The computational cost can be reduced further by leveraging the element weighting techniques described in Section 4.2. By setting to zero all coordinates that have a low PPMI, PMI or tf-idf score, the coordinate density is dramatically reduced at the cost of losing little discriminative power. In this vein, Bayardo, Ma, and Srikant (2007) described a strategy that omits the coordinates with the highest number of nonzero values. Their algorithm gives a significant advantage only when we are interested in finding solely the similarity between highly similar vectors.

4.5.2 Distributed Implementation using MapReduce

The algorithm described in Section 4.5.1 assumes that the matrix $\mathbf{X}$ can fit into memory, which for large $\mathbf{X}$ may be impossible. Also, as each element of $\mathbf{X}$ is processed independently, running parallel processes for non-intersecting subsets of $\mathbf{X}$ makes the processing faster. Elsayed, Lin, and Oard (2008) proposed a MapReduce implementation deployed using Hadoop, an open-source software package implementing the MapReduce framework and distributed file system.[19] Hadoop has been shown to scale to several thousands of machines, allowing users to write simple code, and to seamlessly manage the sophisticated parallel execution of the code. Dean and Ghemawat (2008) provide a good primer on MapReduce programming.

19. Hadoop is available for download at http://lucene.apache.org/hadoop/.

The MapReduce model's Map step is used to start $m \times n$ Map tasks in parallel, each caching one m-th part of $\mathbf{X}$ as an inverted index and streaming one n-th part of $\mathbf{X}$ through it. The actual inputs are read by the tasks directly from HDFS (Hadoop Distributed File System). The value of m is determined by the amount of memory dedicated for the inverted index, and n should be determined by trading off the fact that, as n increases, more parallelism can be obtained at the increased cost of building the same inverted index n times.

The similarity algorithm from Section 4.5.1 runs in each task of the Map step of a MapReduce job. The Reduce step groups the output by the rows (or columns) of $\mathbf{X}$.

4.5.3 Randomized Algorithms

Other optimization strategies use randomized techniques to approximate various similarity measures. The aim of randomized algorithms is to improve computational efficiency (memory and time) by projecting high-dimensional vectors into a low-dimensional subspace. Truncated SVD performs such a projection, but SVD can be computationally intensive.[20] The insight of randomized techniques is that high-dimensional vectors can be randomly projected into a low-dimensional subspace with relatively little impact on the final similarity scores. Significant reductions in computational cost have been reported with little average error to computing the true similarity scores, especially in applications such as word similarity where we are interested in only the top-k most similar vectors to each vector (Ravichandran, Pantel, & Hovy, 2005; Gorman & Curran, 2006).

Random Indexing, an approximation technique based on Sparse Distributed Memory (Kanerva, 1993), computes the pairwise similarity between all rows (or vectors) of a matrix with complexity $O(n_r n_c \delta_1)$, where δ_1 is a fixed constant representing the length of the index vectors assigned to each column. The value of δ_1 controls the tradeoff of accuracy versus efficiency. The elements of each index vector are mostly zeros with a small number of randomly assigned +1's and −1's. The cosine measure between two rows $\mathbf{r}_1$ and $\mathbf{r}_2$ is then approximated by computing the cosine between two *fingerprint* vectors, $\mathbf{fingerprint}(\mathbf{r}_1)$ and $\mathbf{fingerprint}(\mathbf{r}_2)$, where $\mathbf{fingerprint}(\mathbf{r})$ is computed by summing the index vectors of each non-unique coordinate of $\mathbf{r}$. *Random Indexing* was shown to perform as well as LSA on a word synonym selection task (Karlgren & Sahlgren, 2001).

Locality sensitive hashing (LSH) (Broder, 1997) is another technique that approximates the similarity matrix with complexity $O(n_r^2 \delta_2)$, where δ_2 is a constant number of random projections, which controls the accuracy versus efficiency tradeoff.[21] LSH is a general class of techniques for defining functions that map vectors (rows or columns) into short signatures or fingerprints, such that two similar vectors are likely to have similar fingerprints. Definitions of LSH functions include the Min-wise independent function, which preserves the Jaccard similarity between vectors (Broder, 1997), and functions that preserve the cosine similarity between vectors (Charikar, 2002). On a word similarity task, Ravichandran et al. (2005) showed that, on average, over 80% of the top-10 similar words of random words are found in the top-10 results using Charikar's functions, and that the average cosine error is 0.016

20. However, there are efficient forms of SVD (Brand, 2006; Gorrell, 2006).
21. LSH stems from work by Rabin (1981), who proposed the use of hash functions from random irreducible polynomials to create short fingerprints of collections of documents. Such techniques are useful for many tasks, such as removing duplicate documents (*deduping*) in a web crawl.

(using δ_2 = 10,000 random projections). Gorman and Curran (2006) provide a detailed comparison of Random Indexing and LSH on a distributional similarity task. On the BNC corpus, LSH outperformed Random Indexing; however, on a larger corpora combining BNC, the Reuters Corpus, and most of the English news holdings of the LDC in 2003, Random Indexing outperformed LSH in both efficiency and accuracy.

4.6 Machine Learning

If the intended application for a VSM is clustering or classification, a similarity measure such as cosine (Section 4.4) may be used. For classification, a nearest-neighbour algorithm can use cosine as a measure of nearness (Dasarathy, 1991). For clustering, a similarity-based clustering algorithm can use cosine as a measure of similarity (Jain, Murty, & Flynn, 1999). However, there are many machine learning algorithms that can work directly with the vectors in a VSM, without requiring an external similarity measure, such as cosine. In effect, such machine learning algorithms implicitly use their own internal approaches to measuring similarity.

Any machine learning algorithm that works with real-valued vectors can use vectors from a VSM (Witten & Frank, 2005). Linguistic processing (Section 3) and mathematical processing (Section 4) may still be necessary, but the machine learning algorithm can handle vector comparison (Sections 4.4 and 4.5).

In addition to unsupervised (clustering) and supervised (classification) machine learning, vectors from a VSM may also be used in semi-supervised learning (Ando & Zhang, 2005; Collobert & Weston, 2008). In general, there is nothing unique to VSMs that would compel a choice of one machine learning algorithm over another, aside from the algorithm's performance on the given task. Therefore we refer our readers to the machine learning literature (Witten & Frank, 2005), since we have no advice that is specific to VSMs.

5. Three Open Source VSM Systems

To illustrate the three types of VSMs discussed in Section 2, this section presents three open source systems, one for each VSM type. We have chosen to present open source systems so that interested readers can obtain the source code to find out more about the systems and to apply the systems in their own projects. All three systems are written in Java and are designed for portability and ease of use.

5.1 The Term–Document Matrix: Lucene

Lucene[22] is an open source full-featured text search engine library supported by the Apache Software Foundation. It is arguably the most ubiquitous implementation of a term–document matrix, powering many search engines such as at CNET, SourceForge, Wikipedia, Disney, AOL and Comcast. Lucene offers efficient storage, indexing, as well as retrieval and ranking functionalities. Although it is primarily used as a term–document matrix, it generalizes to other VSMs.

22. Apache Lucene is available for download at http://lucene.apache.org/.

Content, such as webpages, PDF documents, images, and video, are programmatically decomposed into *fields* and stored in a database. The database implements the term–document matrix, where content corresponds to documents and fields correspond to terms. Fields are stored in the database and indices are computed on the field values. Lucene uses fields as a generalization of content terms, allowing any other *string* or *literal* to index documents. For example, a webpage could be indexed by all the terms it contains, and also by the anchor texts pointing to it, its host name, and the semantic classes in which it is classified (e.g., spam, product review, news, etc.). The webpage can then be retrieved by search terms matching any of these *fields*.

Columns in the term–document matrix consist of all the fields of a particular instance of content (e.g., a webpage). The rows consist of all instances of content in the index. Various statistics such as *frequency* and *tf-idf* are stored in the matrix. The developer defines the fields in a schema and identifies those to be indexed by Lucene. The developer also optionally defines a content ranking function for each indexed field.

Once the index is built, Lucene offers functionalities for retrieving content. Users can issue many query types such as phrase queries, wildcard queries, proximity queries, range queries (e.g., date range queries), and field-restricted queries. Results can be sorted by any field and index updates can occur simultaneously during searching. Lucene's index can be directly loaded into a Tomcat webserver and it offers APIs for common programming languages. Solr,[23] a separate Apache Software Foundation project, is an open source enterprise webserver for searching a Lucene index and presenting search results. It is a full-featured webserver providing functionalities such as XML/HTTP and JSON APIs, hit highlighting, faceted search, caching, and replication.

A simple recipe for creating a web search service, using Nutch, Lucene and Solr, consists of crawling a set of URLs (using Nutch), creating a term–document matrix index (using Lucene), and serving search results (using Solr). Nutch,[24] the Apache Software Foundation open source web search software, offers functionality for crawling the web from a seed set of URLs, for building a link-graph of the web crawl, and for parsing web documents such as HTML pages. A good set of seed URLs for Nutch can be downloaded freely from the Open Directory Project.[25] Crawled pages are HTML-parsed, and they are then indexed by Lucene. The resulting indexed collection is then queried and served through a Solr installation with Tomcat.

For more information on Lucene, we recommend Gospodnetic and Hatcher's (2004) book. Konchady (2008) explains how to integrate Lucene with LingPipe and GATE for sophisticated semantic processing.[26]

23. Apache Solr is available for download at http://lucene.apache.org/solr/.
24. Apache Nutch is available for download at http://lucene.apache.org/nutch/.
25. See http://www.dmoz.org/.
26. Information about LingPipe is available at http://alias-i.com/lingpipe/. The GATE (General Architecture for Text Engineering) home page is at http://gate.ac.uk/.

5.2 The Word–Context Matrix: Semantic Vectors

Semantic Vectors[27] is an open source project implementing the random projection approach to measuring word similarity (see Section 4.5.3). The package uses Lucene to create a term–document matrix, and it then creates vectors from Lucene's term–document matrix, using random projection for dimensionality reduction. The random projection vectors can be used, for example, to measure the semantic similarity of two words or to find the words that are most similar to a given word.

The idea of random projection is to take high-dimensional vectors and randomly project them into a relatively low-dimensional space (Sahlgren, 2005). This can be viewed as a kind of smoothing operation (Section 4.3), but the developers of the Semantic Vectors package emphasize the simplicity and efficiency of random projection (Section 4.5), rather than its smoothing ability. They argue that other matrix smoothing algorithms might smooth better, but none of them perform as well as random indexing, in terms of the computational complexity of building a smooth matrix and incrementally updating the matrix when new data arrives (Widdows & Ferraro, 2008). Their aim is to encourage research and development with semantic vectors by creating a simple and efficient open source package.

The Semantic Vectors package is designed to be convenient to use, portable, and easy to extend and modify. The design of the software incorporates lessons learned from the earlier Stanford Infomap project.[28] Although the default is to generate random projection vectors, the system has a modular design that allows other kinds of word–context matrices to be used instead of random projection matrices.

The package supports two basic functions: building a word–context matrix and searching through the vectors in the matrix. In addition to generating word vectors, the building operation can generate document vectors by calculating weighted sums of the word vectors for the words in each document. The searching operation can be used to search for similar words or to search for documents that are similar to a query. A query can be a single word or several words can be combined, using various mathematical operations on the corresponding vectors. The mathematical operations include vector negation and disjunction, based on quantum logic (Widdows, 2004). Widdows and Ferraro (2008) provide a good summary of the Semantic Vectors software.

5.3 The Pair–Pattern Matrix: Latent Relational Analysis in S-Space

Latent Relational Analysis[29] (LRA) is an open source project implementing the pair–pattern matrix. It is a component of the *S-Space* package, a library of tools for building and comparing different semantic spaces.

LRA takes as input a textual corpus and a set of word pairs. A pair–pattern matrix is built by deriving lexical patterns that link together the word pairs in the corpus. For example, consider the word pair ⟨Korea, Japan⟩ and the following retrieved matching sentences:

27. Semantic Vectors is a software package for measuring word similarity, available under the Simplified BSD License at http://code.google.com/p/semanticvectors/.
28. See http://infomap-nlp.sourceforge.net/.
29. Latent Relational Analysis is part of the *S-Space* package and is distributed under the GNU General Public License version 2. It is available at http://code.google.com/p/airhead-research/. At the time of writing, the LRA module was under development.

- *Korea looks to new Japan prime minister's effect on Korea-Japan relations.*
- *What channel is the Korea vs. Japan football game?*

From these two sentences, LRA extracts two patterns: "X looks to new Y" and "X vs. Y". These patterns become two columns in the pair–pattern matrix, and the word pair ⟨Korea, Japan⟩ becomes a row. Pattern frequencies are counted and then smoothed using SVD (see Section 4.3).

In order to mitigate the sparseness of occurrences of word pairs, a thesaurus such as WordNet is used to expand the seed word pairs to alternatives. For example the pair ⟨Korea, Japan⟩ may be expanded to include ⟨South Korea, Japan⟩, ⟨Republic of Korea, Japan⟩, ⟨Korea, Nippon⟩, ⟨South Korea, Nippon⟩, and ⟨Republic of Korea, Nippon⟩.

LRA uses Lucene (see Section 5.1) as its backend to store the matrix, index it, and serve its contents. For a detailed description of the LRA algorithm, we suggest Turney's (2006) paper.

6. Applications

In this section, we will survey some of the semantic applications for VSMs. We will aim for breadth, rather than depth; readers who want more depth should consult the references. Our goal is to give the reader an impression of the scope and flexibility of VSMs for semantics. The following applications are grouped according to the type of matrix involved: term–document, word–context, or pair–pattern. Note that this section is not exhaustive; there are many more references and applications than we have space to list here.

6.1 Term–Document Matrices

Term–document matrices are most suited to measuring the semantic similarity of documents and queries (see Section 2.1). The usual measure of similarity is the cosine of column vectors in a weighted term–document matrix. There are a variety of applications for measures of document similarity.

Document retrieval: The term–document matrix was first developed for document retrieval (Salton et al., 1975), and there is now a large body of literature on the VSM for document retrieval (Manning et al., 2008), including several journals and conferences devoted to the topic. The core idea is, given a query, rank the documents in order of decreasing cosine of the angles between the query vector and the document vectors (Salton et al., 1975). One variation on the theme is cross-lingual document retrieval, where a query in one language is used to retrieve a document in another language (Landauer & Littman, 1990). An important technical advance was the discovery that smoothing the term–document matrix by truncated SVD can improve precision and recall (Deerwester et al., 1990), although few commercial systems use smoothing, due to the computational expense when the document collection is large and dynamic. Random indexing (Sahlgren, 2005) or incremental SVD (Brand, 2006) may help to address these scaling issues. Another important development in document retrieval has been the addition of collaborative filtering, in the form of PageRank (Brin & Page, 1998).

Document clustering: Given a measure of document similarity, we can cluster the documents into groups, such that similarity tends to be high within a group, but low across groups (Manning et al., 2008). The clusters may be partitional (flat) (Cutting, Karger, Pedersen, & Tukey, 1992; Pantel & Lin, 2002b) or they may have a hierarchical structure (groups of groups) (Zhao & Karypis, 2002); they may be non-overlapping (hard) (Croft, 1977) or overlapping (soft) (Zamir & Etzioni, 1999). Clustering algorithms also differ in how clusters are compared and abstracted. With single-link clustering, the similarity between two clusters is the maximum of the similarities between their members. Complete-link clustering uses the minimum of the similarities and average-link clustering uses the average of the similarities (Manning et al., 2008).

Document classification: Given a training set of documents with class labels and a testing set of unlabeled documents, the task of document classification is to learn from the training set how to assign labels to the testing set (Manning et al., 2008). The labels may be the topics of the documents (Sebastiani, 2002), the sentiment of the documents (e.g., positive versus negative product reviews) (Pang, Lee, & Vaithyanathan, 2002; Kim, Pantel, Chklovski, & Pennacchiotti, 2006), spam versus non-spam (Sahami, Dumais, Heckerman, & Horvitz, 1998; Pantel & Lin, 1998), or any other labels that might be inferred from the words in the documents. When we classify documents, we are implying that the documents in a class are similar in some way; thus document classification implies some notion of document similarity, and most machine learning approaches to document classification involve a term–document matrix (Sebastiani, 2002). A measure of document similarity, such as cosine, can be directly applied to document classification by using a nearest-neighbour algorithm (Yang, 1999).

Essay grading: Student essays may be automatically graded by comparing them to one or more high-quality reference essays on the given essay topic (Wolfe, Schreiner, Rehder, Laham, Foltz, Kintsch, & Landauer, 1998; Foltz, Laham, & Landauer, 1999). The student essays and the reference essays can be compared by their cosines in a term–document matrix. The grade that is assigned to a student essay is proportional to its similarity to one of the reference essays; a student essay that is highly similar to a reference essay gets a high grade.

Document segmentation: The task of document segmentation is to partition a document into sections, where each section focuses on a different subtopic of the document (Hearst, 1997; Choi, 2000). We may treat the document as a series of blocks, where a block is a sentence or a paragraph. The problem is to detect a topic shift from one block to the next. Hearst (1997) and Choi (2000) both use the cosine between columns in a word–block frequency matrix to measure the semantic similarity of blocks. A topic shift is signaled by a drop in the cosine between consecutive blocks. The word–block matrix can be viewed as a small term–document matrix, where the corpus is a single document and the documents are blocks.

Question answering: Given a simple question, the task in Question Answering (QA) is to find a short answer to the question by searching in a large corpus. A typical question is, "How many calories are there in a Big Mac?" Most algorithms for QA have four components, question analysis, document retrieval, passage retrieval, and answer extraction (Tellex, Katz, Lin, Fern, & Marton, 2003; Dang, Lin, & Kelly, 2006). Vector-based similarity measurements are often used for both document retrieval and passage retrieval (Tellex et al., 2003).

Call routing: Chu-carroll and Carpenter (1999) present a vector-based system for automatically routing telephone calls, based on the caller's spoken answer to the question, "How may I direct your call?" If the caller's answer is ambiguous, the system automatically generates a question for the caller, derived from the VSM, that prompts the caller for further information.

6.2 Word–Context Matrices

Word–context matrices are most suited to measuring the semantic similarity of words (see Section 2.2). For example, we can measure the similarity of two words by the cosine of the angle between their corresponding row vectors in a word–context matrix. There are many applications for measures of word similarity.

Word similarity: Deerwester et al. (1990) discovered that we can measure word similarity by comparing row vectors in a term–document matrix. Landauer and Dumais (1997) evaluated this approach with 80 multiple-choice synonym questions from the Test of English as a Foreign Language (TOEFL), achieving human-level performance (64.4% correct for the word–context matrix and 64.5% for the average non-English US college applicant). The documents used by Landauer and Dumais had an average length of 151 words, which seems short for a document, but long for the context of a word. Other researchers soon switched to much shorter lengths, which is why we prefer to call these *word–context* matrices, instead of *term–document* matrices. Lund and Burgess (1996) used a context window of ten words. Schütze (1998) used a fifty-word window (±25 words, centered on the target word). Rapp (2003) achieved 92.5% correct on the 80 TOEFL questions, using a four-word context window (±2 words, centered on the target word, after removing stop words). The TOEFL results suggest that performance improves as the context window shrinks. It seems that the immediate context of a word is much more important than the distant context for determining the meaning of the word.

Word clustering: Pereira, Tishby, and Lee (1993) applied soft hierarchical clustering to row-vectors in a word–context matrix. In one experiment, the words were nouns and the contexts were verbs for which the given nouns were direct objects. In another experiment, the words were verbs and the contexts were nouns that were direct objects of the given verbs. Schütze's (1998) seminal word sense discrimination model used hard flat clustering for row-vectors in a word–context matrix, where the context was given by a window of ±25 words, centered on the target word. Pantel and Lin (2002a) applied soft flat clustering to a word–context matrix, where the context was based on parsed text. These algorithms are able to discover different senses of polysemous words, generating different clusters for each sense. In effect, the different clusters correspond to the different concepts that underlie the words.

Word classification: Turney and Littman (2003) used a word–context matrix to classify words as positive (*honest*, *intrepid*) or negative (*disturbing*, *superfluous*). They used the General Inquirer (GI) lexicon (Stone, Dunphy, Smith, & Ogilvie, 1966) to evaluate their algorithms. The GI lexicon includes 11,788 words, labeled with 182 categories related to opinion, affect, and attitude.[30] Turney and Littman hypothesize that all 182 categories can be discriminated with a word–context matrix.

Automatic thesaurus generation: WordNet is a popular tool for research in natural language processing (Fellbaum, 1998), but creating and maintaing such lexical resources

30. The GI lexicon is available at http://www.wjh.harvard.edu/~inquirer/spreadsheet_guide.htm.

is labour intensive, so it is natural to wonder whether the process can be automated to some degree.[31] This task can seen as an instance of word clustering (when the thesaurus is generated from scratch) or classification (when an existing thesaurus is automatically extended), but it is worthwhile to consider the task of automatic thesaurus generation separately from clustering and classification, due to the specific requirements of a thesaurus, such as the particular kind of similarity that is appropriate for a thesaurus (see Section 2.4). Several researchers have used word–context matrices specifically for the task of assisting or automating thesaurus generation (Crouch, 1988; Grefenstette, 1994; Ruge, 1997; Pantel & Lin, 2002a; Curran & Moens, 2002).

Word sense disambiguation: A typical Word Sense Disambiguation (WSD) system (Agirre & Edmonds, 2006; Pedersen, 2006) uses a feature vector representation in which each vector corresponds to a token of a word, not a type (see Section 2.6). However, Leacock, Towell, and Voorhees (1993) used a word–context frequency matrix for WSD, in which each vector corresponds to a type annotated with a sense tag. Yuret and Yatbaz (2009) applied a word–context frequency matrix to unsupervised WSD, achieving results comparable to the performance of supervised WSD systems.

Context-sensitive spelling correction: People frequently confuse certain sets of words, such as *there*, *they're*, and *their*. These confusions cannot be detected by a simple dictionary-based spelling checker; they require context-sensitive spelling correction. A word–context frequency matrix may be used to correct these kinds of spelling errors (Jones & Martin, 1997).

Semantic role labeling: The task of semantic role labeling is to label parts of a sentence according to the roles they play in the sentence, usually in terms of their connection to the main verb of the sentence. Erk (2007) presented a system in which a word–context frequency matrix was used to improve the performance of semantic role labeling. Pennacchiotti, Cao, Basili, Croce, and Roth (2008) show that word–context matrices can reliably predict the semantic frame to which an unknown lexical unit refers, with good levels of accuracy. Such lexical unit induction is important in semantic role labeling, to narrow the candidate set of roles of any observed lexical unit.

Query expansion: Queries submitted to search engines such as Google and Yahoo! often do not directly match the terms in the most relevant documents. To alleviate this problem, the process of query expansion is used for generating new search terms that are consistent with the intent of the original query. VSMs form the basis of query semantics models (Cao, Jiang, Pei, He, Liao, Chen, & Li, 2008). Some methods represent queries by using session contexts, such as query cooccurrences in user sessions (Huang, Chien, & Oyang, 2003; Jones, Rey, Madani, & Greiner, 2006), and others use click contexts, such as the urls that were clicked on as a result of a query (Wen, Nie, & Zhang, 2001).

Textual advertising: In pay-per-click advertising models, prevalent in search engines such as Google and Yahoo!, users pay for keywords, called *bidterms*, which are then used to display their ads when relevant queries are issued by users. The scarcity of data makes ad matching difficult and, in response, several techniques for bidterm expansion using VSMs have been proposed. The word–context matrix consists of rows of bidterms and the columns

31. WordNet is available at http://wordnet.princeton.edu/.

(contexts) consist of advertiser identifiers (Gleich & Zhukov, 2004) or co-bidded bidterms (second order co-occurrences) (Chang, Pantel, Popescu, & Gabrilovich, 2009).

Information extraction: The field of information extraction (IE) includes named entity recognition (NER: recognizing that a chunk of text is the name of an entity, such as a person or a place), relation extraction, event extraction, and fact extraction. Paşca et al. (2006) demonstrate that a word–context frequency matrix can facilitate fact extraction. Vyas and Pantel (2009) propose a semi-supervised model using a word–context matrix for building and iteratively refining arbitrary classes of named entities.

6.3 Pair–Pattern Matrices

Pair–pattern matrices are most suited to measuring the semantic similarity of word pairs and patterns (see Section 2.3). For example, we can measure the similarity of two word pairs by the cosine of the angle between their corresponding row vectors in a pair–pattern matrix. There are many applications for measures of relational similarity.

Relational similarity: Just as we can measure attributional similarity by the cosine of the angle between row vectors in a word–context matrix, we can measure relational similarity by the cosine of the angle between rows in a pair–pattern matrix. This approach to measuring relational similarity was introduced by Turney et al. (2003) and examined in more detail by Turney and Littman (2005). Turney (2006) evaluated this approach to relational similarity with 374 multiple-choice analogy questions from the SAT college entrance test, achieving human-level performance (56% correct for the pair–pattern matrix and 57% correct for the average US college applicant). This is the highest performance so far for an algorithm. The best algorithm based on attributional similarity has an accuracy of only 35% (Turney, 2006). The best non-VSM algorithm achieves 43% (Veale, 2004).

Pattern similarity: Instead of measuring the similarity between row vectors in a pair–pattern matrix, we can measure the similarity between columns; that is, we can measure pattern similarity. Lin and Pantel (2001) constructed a pair–pattern matrix in which the patterns were derived from parsed text. Pattern similarity can be used to infer that one phrase is a paraphrase of another phrase, which is useful for natural language generation, text summarization, information retrieval, and question answering.

Relational clustering: Biçici and Yuret (2006) clustered word pairs by representing them as row vectors in a pair–pattern matrix. Davidov and Rappoport (2008) first clustered contexts (patterns) and then identified representative pairs for each context cluster. They used the representative pairs to automatically generate multiple-choice analogy questions, in the style of SAT analogy questions.

Relational classification: Chklovski and Pantel (2004) used a pair–pattern matrix to classify pairs of verbs into semantic classes. For example, *taint* : *poison* is classified as *strength* (poisoning is stronger than tainting) and *assess* : *review* is classified as *enablement* (assessing is enabled by reviewing). Turney (2005) used a pair–pattern matrix to classify noun compounds into semantic classes. For example, *flu virus* is classified as *cause* (the virus causes the flu), *home town* is classified as *location* (the home is located in the town), and *weather report* is classified as *topic* (the topic of the report is the weather).

Relational search: Cafarella, Banko, and Etzioni (2006) described relational search as the task of searching for entities that satisfy given semantic relations. An example of a query for a relational search engine is "list all X such that X causes cancer". In this example, the relation, *cause*, and one of the terms in the relation, *cancer*, are given by the user, and the task of the search engine is to find terms that satisfy the user's query. The organizers of Task 4 in SemEval 2007 (Girju, Nakov, Nastase, Szpakowicz, Turney, & Yuret, 2007) envisioned a two-step approach to relational search: first a conventional search engine would look for candidate answers, then a relational classification system would filter out incorrect answers. The first step was manually simulated by the Task 4 organizers and the goal of Task 4 was to design systems for the second step. This task attracted 14 teams who submitted 15 systems. Nakov and Hearst (2007) achieved good results using a pair–pattern matrix.

Automatic thesaurus generation: We discussed automatic thesaurus generation in Section 6.2, with word–context matrices, but arguably relational similarity is more relevant than attributional similarity for thesaurus generation. For example, most of the information in WordNet is in the relations between the words rather than in the words individually. Snow, Jurafsky, and Ng (2006) used a pair–pattern matrix to build a hypernym-hyponym taxonomy, whereas Pennacchiotti and Pantel (2006) built a meronymy and causation taxonomy. Turney (2008b) showed how a pair–pattern matrix can distinguish synonyms from antonyms, synonyms from non-synonyms, and taxonomically similar words (*hair* and *fur*) from words that are merely semantically associated (*cradle* and *baby*).

Analogical mapping: Proportional analogies have the form $a\!:\!b\!::\!c\!:\!d$, which means "a is to b as c is to d". For example, *mason* : *stone* :: *carpenter* : *wood* means "mason is to stone as carpenter is to wood". The 374 multiple-choice analogy questions from the SAT college entrance test (mentioned above) all involve proportional analogies. With a pair–pattern matrix, we can solve proportional analogies by selecting the choice that maximizes relational similarity (e.g., $\mathrm{sim_r}(mason\!:\!stone, carpenter\!:\!wood)$ has a high value). However, we often encounter analogies that involve more than four terms. The well-known analogy between the solar system and the Rutherford-Bohr model of the atom contains at least fourteen terms. For the solar system, we have *planet, attracts, revolves, sun, gravity, solar system*, and *mass*. For the atom, we have *revolves, atom, attracts, electromagnetism, nucleus, charge*, and *electron*. Turney (2008a) demonstrated that we can handle these more complex, systematic analogies by decomposing them into sets of proportional analogies.

7. Alternative Approaches to Semantics

The applications that we list in Section 6 do not necessarily require a VSM approach. For each application, there are many other possible approaches. In this section, we briefly consider a few of the main alternatives.

Underlying the applications for term–document matrices (Section 6.1) is the task of measuring the semantic similarity of documents and queries. The main alternatives to VSMs for this task are probabilistic models, such as the traditional probabilistic retrieval models in information retrieval (van Rijsbergen, 1979; Baeza-Yates & Ribeiro-Neto, 1999) and the more recent statistical language models inspired by information theory (Liu & Croft, 2005). The idea of statistical language models for information retrieval is to measure the similarity between a query and a document by creating a probabilistic language model of the given document and then measuring the probability of the given query according to the language model.

With progress in information retrieval, the distinction between the VSM approach and the probabilistic approach is becoming blurred, as each approach borrows ideas from the other. Language models typically involve multiplying probabilities, but we can view this as adding logs of probabilities, which makes some language models look similar to VSMs.

The applications for word–context matrices (Section 6.2) share the task of measuring the semantic similarity of words. The main alternatives to VSMs for measuring word similarity are approaches that use lexicons, such as WordNet (Resnik, 1995; Jiang & Conrath, 1997; Hirst & St-Onge, 1998; Leacock & Chodrow, 1998; Budanitsky & Hirst, 2001). The idea is to view the lexicon as a graph, in which nodes correspond to word senses and edges represent relations between words, such as hypernymy and hyponymy. The similarity between two words is then proportional to the length of the path in the graph that joins the two words.

Several approaches to measuring the semantic similarity of words combine a VSM with a lexicon (Turney et al., 2003; Pantel, 2005; Patwardhan & Pedersen, 2006; Mohammad & Hirst, 2006). Humans use both dictionary definitions and observations of word usage, so it is natural to expect the best performance from algorithms that use both distributional and lexical information.

Pair–pattern matrices (Section 6.3) have in common the task of measuring the semantic similarity of relations. As with word–context matrices, the main alternatives are approaches that use lexicons (Rosario & Hearst, 2001; Rosario, Hearst, & Fillmore, 2002; Nastase & Szpakowicz, 2003; Veale, 2003, 2004). The idea is to reduce relational similarity to attributional similarity, $\mathrm{sim_r}(a : b, c : d) \approx \mathrm{sim_a}(a, c) + \mathrm{sim_a}(b, d)$, and then use a lexicon to measure attributional similarity. As we discuss in Section 2.4, this reduction does not work in general. However, the reduction is often a good approximation, and there is some evidence that a hybrid approach, combining a VSM with a lexicon, can be beneficial (Turney et al., 2003; Nastase, Sayyad-Shirabad, Sokolova, & Szpakowicz, 2006).

8. The Future of Vector Space Models of Semantics

Several authors have criticized VSMs (French & Labiouse, 2002; Padó & Lapata, 2003; Morris & Hirst, 2004; Budanitsky & Hirst, 2006). Most of the criticism stems from the fact that term–document and word–context matrices typically ignore word order. In LSA, for instance, a phrase is commonly represented by the sum of the vectors for the individual words in the phrase; hence the phrases *house boat* and *boat house* will be represented by the same vector, although they have different meanings. In English, word order expresses relational information. Both *house boat* and *boat house* have a Tool-Purpose relation, but *house boat* means Tool-Purpose(*boat*, *house*) (a boat that serves as a house), whereas *boat house* means Tool-Purpose(*house*, *boat*) (a house for sheltering and storing boats).

Landauer (2002) estimates that 80% of the meaning of English text comes from word choice and the remaining 20% comes from word order. However, VSMs are not inherently limited to 80% of the meaning of text. Mitchell and Lapata (2008) propose composition models sensitive to word order. For example, to make a simple additive model become syntax-aware, they allow for different weightings of the contributions of the vector components. Constituents that are more important to the composition therefore can participate more actively. Clark and Pulman (2007) assigned distributional meaning to sentences using the Hilbert space tensor product. Widdows and Ferraro (2008), inspired by quantum mechanics, explores several operators for modeling composition of meaning. Pair–pattern matrices are sensitive to the order of the words in a pair (Turney, 2006). Thus there are several ways to handle word order in VSMs.

This raises the question, what are the limits of VSMs for semantics? Can all semantics be represented with VSMs? There is much that we do not yet know how to represent with VSMs. For example, Widdows (2004) and van Rijsbergen (2004) show how disjunction, conjunction, and negation can be represented with vectors, but we do not yet know how to represent arbitrary statements in first-order predicate calculus. However, it seems possible that future work may discover answers to these limitations.

In this survey, we have assumed that VSMs are composed of elements with values that are derived from event frequencies. This ties VSMs to some form of distributional hypothesis (see Sections 1.1 and 2.7); therefore the limits of VSMs depend on the limits of the family of distributional hypotheses. Are statistical patterns of word usage sufficient to figure out what people mean? This is arguably the major open question of VSMs, and the answer will determine the future of VSMs. We do not have a strong argument one way or the other, but we believe that the continuing progress with VSMs suggests we are far from reaching their limits.

9. Conclusions

When we want information or help from a person, we use words to make a request or describe a problem, and the person replies with words. Unfortunately, computers do not understand human language, so we are forced to use artificial languages and unnatural user interfaces. In science fiction, we dream of computers that understand human language, that can listen to us and talk with us. To achieve the full potential of computers, we must enable them to understand the semantics of natural language. VSMs are likely to be part of the solution to the problem of computing semantics.

Many researchers who have struggled with the problem of semantics have come to the conclusion that the meaning of words is closely connected to the statistics of word usage (Section 2.7). When we try to make this intuition precise, we soon find we are working with vectors of values derived from event frequencies; that is, we are dealing with VSMs.

In this survey, we have organized past work with VSMs according to the structure of the matrix (term–document, word–context, or pair–pattern). We believe that the structure of the matrix is the most important factor in determining the types of applications that are possible. The linguistic processing (Section 3) and mathematical processing (Section 4) play smaller (but important) roles.

Our goal in this survey has been to show the breadth and power of VSMs, to introduce VSMs to those who less familiar with them, and to provide a new perspective on VSMs to those who are already familiar with them. We hope that our emphasis on the structure of the matrix will inspire new research. There is no reason to believe that the three matrix types we present here exhaust the possibilities. We expect new matrix types and new tensors will open up more applications for VSMs. It seems possible to us that all of the semantics of human language might one day be captured in some kind of VSM.

Acknowledgments

Thanks to Annie Zaenen for prompting this paper. Thanks to Saif Mohammad and Mariana Soffer for their comments. Thanks to Arkady Borkovsky and Eric Crestan for developing the distributed sparse-matrix multiplication algorithm, and to Marco Pennacchiotti for his invaluable comments. Thanks to the anonymous reviewers of *JAIR* for their very helpful comments and suggestions.

References

Acar, E., & Yener, B. (2009). Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, *21*(1), 6–20.

Agirre, E., & Edmonds, P. G. (2006). *Word Sense Disambiguation: Algorithms and Applications*. Springer.

Ando, R. K. (2000). Latent semantic space: Iterative scaling improves precision of inter-document similarity measurement. In *Proceedings of the 23rd Annual ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-2000)*, pp. 216–223.

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, *6*, 1817–1853.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley.

Barr, C., Jones, R., & Regelson, M. (2008). The linguistic structure of English web-search queries. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Bayardo, R. J., Ma, Y., & Srikant, R. (2007). Scaling up all pairs similarity search. In *Proceedings of the 16th international conference on World Wide Web (WWW '07)*, pp. 131–140, New York, NY. ACM.

Biçici, E., & Yuret, D. (2006). Clustering word pairs to answer analogy questions. In *Proceedings of the Fifteenth Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN 2006)*, Akyaka, Mugla, Turkey.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Brand, M. (2006). Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and Its Applications*, *415*(1), 20–30.

Breese, J., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 43–52. Morgan Kaufmann.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the Seventh World Wide Web Conference (WWW7)*, pp. 107–117.

Broder, A. (1997). On the resemblance and containment of documents. In *In Compression and Complexity of Sequences (SEQUENCES'97)*, pp. 21–29. IEEE Computer Society.

Budanitsky, A., & Hirst, G. (2001). Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Proceedings of the Workshop on WordNet and Other Lexical Resources, Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001)*, pp. 29–24, Pittsburgh, PA.

Budanitsky, A., & Hirst, G. (2006). Evaluating wordnet-based measures of semantic distance. *Computational Linguistics*, *32*(1), 13–47.

Bullinaria, J., & Levy, J. (2007). Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, *39*(3), 510–526.

Buntine, W., & Jakulin, A. (2006). Discrete component analysis. In *Subspace, Latent Structure and Feature Selection: Statistical and Optimization Perspectives Workshop at SLSFS 2005*, pp. 1–33, Bohinj, Slovenia. Springer.

Cafarella, M. J., Banko, M., & Etzioni, O. (2006). Relational web search. Tech. rep., University of Washington, Department of Computer Science and Engineering. Technical Report 2006-04-02.

Cao, G., Nie, J.-Y., & Bai, J. (2005). Integrating word relationships into language models. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '05)*, pp. 298–305, New York, NY. ACM.

Cao, H., Jiang, D., Pei, J., He, Q., Liao, Z., Chen, E., & Li, H. (2008). Context-aware query suggestion by mining click-through and session data. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*, pp. 875–883. ACM.

Carroll, J. D., & Chang, J.-J. (1970). Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, *35*(3), 283–319.

Chang, W., Pantel, P., Popescu, A.-M., & Gabrilovich, E. (2009). Towards intent-driven bidterm suggestion. In *Proceedings of WWW-09 (Short Paper)*, Madrid, Spain.

Charikar, M. S. (2002). Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (STOC '02)*, pp. 380–388. ACM.

Chew, P., Bader, B., Kolda, T., & Abdelali, A. (2007). Cross-language information retrieval using PARAFAC2. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD07)*, pp. 143–152. ACM Press.

Chiarello, C., Burgess, C., Richards, L., & Pollock, A. (1990). Semantic and associative priming in the cerebral hemispheres: Some words do, some words don't . . . sometimes, some places. *Brain and Language*, *38*, 75–104.

Chklovski, T., & Pantel, P. (2004). VerbOcean: Mining the web for fine-grained semantic verb relations. In *Proceedings of Experimental Methods in Natural Language Processing 2004 (EMNLP-04)*, pp. 33–40, Barcelona, Spain.

Choi, F. Y. Y. (2000). Advances in domain independent linear text segmentation. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 26–33.

Chu-carroll, J., & Carpenter, B. (1999). Vector-based natural language call routing. *Computational Linguistics*, *25*(3), 361–388.

Church, K. (1995). One term or two?. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 310–318.

Church, K., & Hanks, P. (1989). Word association norms, mutual information, and lexicography. In *Proceedings of the 27th Annual Conference of the Association of Computational Linguistics*, pp. 76–83, Vancouver, British Columbia.

Clark, S., & Pulman, S. (2007). Combining symbolic and distributional models of meaning. In *Proceedings of AAAI Spring Symposium on Quantum Interaction*, pp. 52–55.

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML-08)*, pp. 160–167.

Croft, W. B. (1977). Clustering large files of documents using the single-link method. *Journal of the American Society for Information Science*, *28*(6), 341–344.

Crouch, C. J. (1988). A cluster-based approach to thesaurus construction. In *Proceedings of the 11th Annual International ACM SIGIR Conference*, pp. 309–320, Grenoble, France.

Curran, J. R., & Moens, M. (2002). Improvements in automatic thesaurus extraction. In *Unsupervised Lexical Acquisition: Proceedings of the Workshop of the ACL Special Interest Group on the Lexicon (SIGLEX)*, pp. 59–66, Philadelphia, PA.

Cutting, D. R., Karger, D. R., Pedersen, J. O., & Tukey, J. W. (1992). Scatter/gather: a cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference*, pp. 318–329.

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

Dang, H. T., Lin, J., & Kelly, D. (2006). Overview of the TREC 2006 question answering track. In *Proceedings of the Fifteenth Text REtrieval Conference (TREC 2006)*.

Dasarathy, B. (1991). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press.

Davidov, D., & Rappoport, A. (2008). Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated SAT analogy questions. In *Proceedings of the 46th Annual Meeting of the ACL and HLT (ACL-HLT-08)*, pp. 692–700, Columbus, Ohio.

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107–113.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science (JASIS)*, *41*(6), 391–407.

Elsayed, T., Lin, J., & Oard, D. (2008). Pairwise document similarity in large collections with mapreduce. In *Proceedings of Association for Computational Linguistics and Human Language Technology Conference 2008 (ACL-08: HLT), Short Papers*, pp. 265–268, Columbus, Ohio. Association for Computational Linguistics.

Erk, K. (2007). A simple, similarity-based model for selectional preferences. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 216–223,, Prague, Czech Republic.

Erk, K., & Padó, S. (2008). A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, pp. 897–906, Honolulu, HI.

Fellbaum, C. (Ed.). (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

Firth, J. R. (1957). A synopsis of linguistic theory 1930–1955. In *Studies in Linguistic Analysis*, pp. 1–32. Blackwell, Oxford.

Foltz, P. W., Laham, D., & Landauer, T. K. (1999). The intelligent essay assessor: Applications to educational technology. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, *1*(2).

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, *3*, 1289–1305.

French, R. M., & Labiouse, C. (2002). Four problems with extracting human semantics from large text corpora. In *Proceedings of the 24th Annual Conference of the Cognitive Science Society*.

Furnas, G. W., Landauer, T. K., Gomez, L. M., & Dumais, S. T. (1983). Statistical semantics: Analysis of the potential performance of keyword information systems. *Bell System Technical Journal*, *62*(6), 1753–1806.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, *7*(2), 155–170.

Gilbert, J. R., Moler, C., & Schreiber, R. (1992). Sparse matrices in MATLAB: Design and implementation. *SIAM Journal on Matrix Analysis and Applications*, *13*(1), 333–356.

Girju, R., Nakov, P., Nastase, V., Szpakowicz, S., Turney, P., & Yuret, D. (2007). Semeval-2007 task 04: Classification of semantic relations between nominals. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval 2007)*, pp. 13–18, Prague, Czech Republic.

Gleich, D., & Zhukov, L. (2004). SVD based term suggestion and ranking system. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM '04)*, pp. 391–394. IEEE Computer Society.

Golub, G. H., & Van Loan, C. F. (1996). *Matrix Computations* (Third edition). Johns Hopkins University Press, Baltimore, MD.

Gorman, J., & Curran, J. R. (2006). Scaling distributional similarity to large corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (ACL 2006)*, pp. 361–368. Association for Computational Linguistics.

Gorrell, G. (2006). Generalized Hebbian algorithm for incremental singular value decomposition in natural language processing. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, pp. 97–104.

Gospodnetic, O., & Hatcher, E. (2004). *Lucene in Action*. Manning Publications.

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

Harris, Z. (1954). Distributional structure. *Word*, *10*(23), 146–162.

Harshman, R. (1970). Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, *16*.

Hearst, M. (1997). Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, *23*(1), 33–64.

Hirst, G., & St-Onge, D. (1998). Lexical chains as representations of context for the detection and correction of malapropisms. In Fellbaum, C. (Ed.), *WordNet: An Electronic Lexical Database*, pp. 305–332. MIT Press.

Hofmann, T. (1999). Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval (SIGIR '99)*, pp. 50–57, Berkeley, California.

Huang, C.-K., Chien, L.-F., & Oyang, Y.-J. (2003). Relevant term suggestion in interactive web search based on contextual information in query session logs. *Journal of the American Society for Information Science and Technology*, *54*(7), 638–649.

Hull, D. (1996). Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society for Information Science*, *47*(1), 70–84.

Jain, A., Murty, N., & Flynn, P. (1999). Data clustering: A review. *ACM Computing Surveys*, *31*(3), 264–323.

Jarmasz, M., & Szpakowicz, S. (2003). Roget's thesaurus and semantic similarity. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-03)*, pp. 212–219, Borovets, Bulgaria.

Jiang, J. J., & Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics (ROCLING X)*, pp. 19–33, Tapei, Taiwan.

Johnson, H., & Martin, J. (2003). Unsupervised learning of morphology for English and Inuktitut. In *Proceedings of HLT-NAACL 2003*, pp. 43–45.

Jones, M. P., & Martin, J. H. (1997). Contextual spelling correction using latent semantic analysis. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pp. 166–173, Washington, DC.

Jones, R., Rey, B., Madani, O., & Greiner, W. (2006). Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web (WWW '06)*, pp. 387–396, New York, NY. ACM.

Jones, W. P., & Furnas, G. W. (1987). Pictures of relevance: A geometric analysis of similarity measures. *Journal of the American Society for Information Science*, *38*(6), 420–442.

Kanerva, P. (1993). Sparse distributed memory and related models. In Hassoun, M. H. (Ed.), *Associative neural memories*, pp. 50–76. Oxford University Press, New York, NY.

Karlgren, J., & Sahlgren, M. (2001). From words to understanding. In Uesaka, Y., Kanerva, P., & Asoh, H. (Eds.), *Foundations of Real-World Intelligence*, pp. 294–308. CSLI Publications.

Kim, S.-M., Pantel, P., Chklovski, T., & Pennacchiotti, M. (2006). Automatically assessing review helpfulness. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pp. 423–430.

Kolda, T., & Bader, B. (2009). Tensor decompositions and applications. *SIAM Review*, *51*(3), 455–500.

Konchady, M. (2008). *Building Search Applications: Lucene, LingPipe, and Gate*. Mustru Publishing.

Kraaij, W., & Pohlmann, R. (1996). Viewing stemming as recall enhancement. In *Proceedings of the 19th Annual International ACM SIGIR Conference*, pp. 40–48.

Lakoff, G. (1987). *Women, Fire, and Dangerous Things*. University Of Chicago Press, Chicago, IL.

Landauer, T. K. (2002). On the computational basis of learning and cognition: Arguments from LSA. In Ross, B. H. (Ed.), *The Psychology of Learning and Motivation: Advances in Research and Theory*, Vol. 41, pp. 43–84. Academic Press.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, *104*(2), 211–240.

Landauer, T. K., & Littman, M. L. (1990). Fully automatic cross-language document retrieval using latent semantic indexing. In *Proceedings of the Sixth Annual Conference of the UW Centre for the New Oxford English Dictionary and Text Research*, pp. 31–38, Waterloo, Ontario.

Landauer, T. K., McNamara, D. S., Dennis, S., & Kintsch, W. (2007). *Handbook of Latent Semantic Analysis*. Lawrence Erlbaum, Mahwah, NJ.

Lavrenko, V., & Croft, W. B. (2001). Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01)*, pp. 120–127, New York, NY. ACM.

Leacock, C., & Chodrow, M. (1998). Combining local context and WordNet similarity for word sense identification. In Fellbaum, C. (Ed.), *WordNet: An Electronic Lexical Database*. MIT Press.

Leacock, C., Towell, G., & Voorhees, E. (1993). Corpus-based statistical sense resolution. In *Proceedings of the ARPA Workshop on Human Language Technology*, pp. 260–265.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by nonnegative matrix factorization. *Nature*, *401*, 788–791.

Lee, L. (1999). Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pp. 25–32.

Lemaire, B., & Denhière, G. (2006). Effects of high-order co-occurrences on word semantic similarity. *Current Psychology Letters: Behaviour, Brain & Cognition*, *18*(1).

Lin, D. (1998). Automatic retrieval and clustering of similar words. In *roceedings of the 17th international conference on Computational linguistics*, pp. 768–774. Association for Computational Linguistics.

Lin, D., & Pantel, P. (2001). DIRT – discovery of inference rules from text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2001*, pp. 323–328.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 76–80.

Liu, X., & Croft, W. B. (2005). Statistical language modeling for information retrieval. *Annual Review of Information Science and Technology*, *39*, 3–28.

Lovins, J. B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, *11*, 22–31.

Lowe, W. (2001). Towards a theory of semantic space. In *Proceedings of the Twenty-first Annual Conference of the Cognitive Science Society*, pp. 576–581.

Lund, K., & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*, *28*(2), 203–208.

Lund, K., Burgess, C., & Atchley, R. A. (1995). Semantic and associative priming in high-dimensional semantic space. In *Proceedings of the 17th Annual Conference of the Cognitive Science Society*, pp. 660–665.

Manning, C., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.

Miller, G., Leacock, C., Tengi, R., & Bunker, R. (1993). A semantic concordance. In *Proceedings of the 3rd DARPA Workshop on Human Language Technology*, pp. 303–308.

Minnen, G., Carroll, J., & Pearce, D. (2001). Applied morphological processing of English. *Natural Language Engineering*, *7*(3), 207–223.

Mitchell, J., & Lapata, M. (2008). Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pp. 236–244, Columbus, Ohio. Association for Computational Linguistics.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill, Columbus, OH.

Mohammad, S., & Hirst, G. (2006). Distributional measures of concept-distance: A task-oriented evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2006)*, pp. 35–43.

Monay, F., & Gatica-Perez, D. (2003). On image auto-annotation with latent space models. In *Proceedings of the Eleventh ACM International Conference on Multimedia*, pp. 275–278.

Morris, J., & Hirst, G. (2004). Non-classical lexical semantic relations. In *Workshop on Computational Lexical Semantics, HLT-NAACL-04*, Boston, MA.

Nakov, P., & Hearst, M. (2007). UCB: System description for SemEval Task 4. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval 2007)*, pp. 366–369, Prague, Czech Republic.

Nakov, P., & Hearst, M. (2008). Solving relational similarity problems using theweb as a corpus. In *Proceedings of ACL-08: HLT*, pp. 452–460, Columbus, Ohio.

Nastase, V., Sayyad-Shirabad, J., Sokolova, M., & Szpakowicz, S. (2006). Learning noun-modifier semantic relations with corpus-based and WordNet-based features. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pp. 781–786.

Nastase, V., & Szpakowicz, S. (2003). Exploring noun-modifier semantic relations. In *Fifth International Workshop on Computational Semantics (IWCS-5)*, pp. 285–301, Tilburg, The Netherlands.

Niwa, Y., & Nitta, Y. (1994). Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In *Proceedings of the 15th International Conference On Computational Linguistics*, pp. 304–309, Kyoto, Japan.

Nosofsky, R. (1986). Attention, similarity, and the identification-categorization relationship. *Journal of Experimental Psychology: General*, *115*(1), 39–57.

Ogden, C. K. (1930). *Basic English: A General Introduction with Rules and Grammar*. Kegan Paul, Trench, Trubner and Co.

Padó, S., & Lapata, M. (2003). Constructing semantic space models from parsed corpora. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 128–135, Sapporo, Japan.

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

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 79–86, Philadelphia, PA.

Pantel, P. (2005). Inducing ontological co-occurrence vectors. In *Proceedings of Association for Computational Linguistics (ACL-05)*, pp. 125–132.

Pantel, P., & Lin, D. (1998). Spamcop: A spam classification and organization program. In *Learning for Text Categorization: Papers from the AAAI 1998 Workshop*, pp. 95–98.

Pantel, P., & Lin, D. (2002a). Discovering word senses from text. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 613–619, Edmonton, Canada.

Pantel, P., & Lin, D. (2002b). Document clustering with committees. In *Proceedings of the 25th Annual International ACM SIGIR Conference*, pp. 199–206.

Paşca, M., Lin, D., Bigham, J., Lifchits, A., & Jain, A. (2006). Names and similarities on the Web: Fact extraction in the fast lane. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pp. 809–816, Sydney, Australia.

Patwardhan, S., & Pedersen, T. (2006). Using wordnet-based context vectors to estimate the semantic relatedness of concepts. In *Proceedings of the Workshop on Making Sense of Sense at the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, pp. 1–8.

Pedersen, T. (2006). Unsupervised corpus-based methods for WSD. In *Word Sense Disambiguation: Algorithms and Applications*, pp. 133–166. Springer.

Pennacchiotti, M., Cao, D. D., Basili, R., Croce, D., & Roth, M. (2008). Automatic induction of FrameNet lexical units. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, pp. 457–465, Honolulu, Hawaii.

Pennacchiotti, M., & Pantel, P. (2006). Ontologizing semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pp. 793–800. Association for Computational Linguistics.

Pereira, F., Tishby, N., & Lee, L. (1993). Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics*, pp. 183–190.

Porter, M. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137.

Rabin, M. O. (1981). Fingerprinting by random polynomials. Tech. rep., Center for research in Computing technology, Harvard University. Technical Report TR-15-81.

Rapp, R. (2003). Word sense discovery based on sense descriptor dissimilarity. In *Proceedings of the Ninth Machine Translation Summit*, pp. 315–322.

Ravichandran, D., Pantel, P., & Hovy, E. (2005). Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, pp. 622–629, Morristown, NJ. Association for Computational Linguistics.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM 1994 Conference on Computer Supported Cooperative Work*, pp. 175–186. ACM Press.

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 448–453, San Mateo, CA. Morgan Kaufmann.

Rosario, B., & Hearst, M. (2001). Classifying the semantic relations in noun-compounds via a domain-specific lexical hierarchy. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-01)*, pp. 82–90.

Rosario, B., Hearst, M., & Fillmore, C. (2002). The descent of hierarchy, and selection in relational semantics. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pp. 247–254.

Rosch, E., & Lloyd, B. (1978). *Cognition and Categorization*. Lawrence Erlbaum, Hillsdale, NJ.

Ruge, G. (1997). Automatic detection of thesaurus relations for information retrieval applications. In Freksa, C., Jantzen, M., & Valk, R. (Eds.), *Foundations of Computer Science*, pp. 499–506. Springer.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A Bayesian approach to filtering junk e-mail. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*.

Sahlgren, M. (2005). An introduction to random indexing. In *Proceedings of the Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering (TKE)*, Copenhagen, Denmark.

Sahlgren, M. (2006). *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Ph.D. thesis, Department of Linguistics, Stockholm University.

Salton, G. (1971). *The SMART retrieval system: Experiments in automatic document processing*. Prentice-Hall, Upper Saddle River, NJ.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, *24*(5), 513–523.

Salton, G., Wong, A., & Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620.

Sarawagi, S., & Kirpal, A. (2004). Efficient set joins on similarity predicates. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD '04)*, pp. 743–754, New York, NY. ACM.

Scholkopf, B., Smola, A. J., & Muller, K.-R. (1997). Kernel principal component analysis. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN-1997)*, pp. 583–588, Berlin.

Schütze, H. (1998). Automatic word sense discrimination. *Computational Linguistics*, *24*(1), 97–124.

Schütze, H., & Pedersen, J. (1993). A vector model for syntagmatic and paradigmatic relatedness. In *Making Sense of Words: Proceedings of the Conference*, pp. 104–113, Oxford, England.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, *34*(1), 1–47.

Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*, 379–423, 623–656.

Singhal, A., Salton, G., Mitra, M., & Buckley, C. (1996). Document length normalization. *Information Processing and Management*, *32*(5), 619–633.

Smith, E., Osherson, D., Rips, L., & Keane, M. (1988). Combining prototypes: A selective modification model. *Cognitive Science*, *12*(4), 485–527.

Snow, R., Jurafsky, D., & Ng, A. Y. (2006). Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pp. 801–808.

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

Spearman, C. (1904). "General intelligence," objectively determined and measured. *American Journal of Psychology*, *15*, 201–293.

Sproat, R., & Emerson, T. (2003). The first international Chinese word segmentation bakeoff. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pp. 133–143, Sapporo, Japan.

Stone, P. J., Dunphy, D. C., Smith, M. S., & Ogilvie, D. M. (1966). *The General Inquirer: A Computer Approach to Content Analysis*. MIT Press, Cambridge, MA.

Tan, B., & Peng, F. (2008). Unsupervised query segmentation using generative language models and Wikipedia. In *Proceeding of the 17th international conference on World Wide Web (WWW '08)*, pp. 347–356, New York, NY. ACM.

Tellex, S., Katz, B., Lin, J., Fern, A., & Marton, G. (2003). Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 41–47.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, *31*(3), 279–311.

Turney, P. D. (2001). Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-01)*, pp. 491–502, Freiburg, Germany.

Turney, P. D. (2005). Measuring semantic similarity by latent relational analysis. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pp. 1136–1141, Edinburgh, Scotland.

Turney, P. D. (2006). Similarity of semantic relations. *Computational Linguistics*, *32*(3), 379–416.

Turney, P. D. (2007). Empirical evaluation of four tensor decomposition algorithms. Tech. rep., Institute for Information Technology, National Research Council of Canada. Technical Report ERB-1152.

Turney, P. D. (2008a). The latent relation mapping engine: Algorithm and experiments. *Journal of Artificial Intelligence Research*, *33*, 615–655.

Turney, P. D. (2008b). A uniform approach to analogies, synonyms, antonyms, and associations. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pp. 905–912, Manchester, UK.

Turney, P. D., & Littman, M. L. (2003). Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, *21*(4), 315–346.

Turney, P. D., & Littman, M. L. (2005). Corpus-based learning of analogies and semantic relations. *Machine Learning*, *60*(1–3), 251–278.

Turney, P. D., Littman, M. L., Bigham, J., & Shnayder, V. (2003). Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-03)*, pp. 482–489, Borovets, Bulgaria.

Van de Cruys, T. (2009). A non-negative tensor factorization model for selectional preference induction. In *Proceedings of the Workshop on Geometric Models for Natural Language Semantics (GEMS-09)*, pp. 83–90, Athens, Greece.

van Rijsbergen, C. J. (2004). *The Geometry of Information Retrieval*. Cambridge University Press, Cambridge, UK.

van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworths.

Veale, T. (2003). The analogical thesaurus. In *Proceedings of the 15th Innovative Applications of Artificial Intelligence Conference (IAAI 2003)*, pp. 137–142, Acapulco, Mexico.

Veale, T. (2004). WordNet sits the SAT: A knowledge-based approach to lexical analogy. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pp. 606–612, Valencia, Spain.

Vozalis, E., & Margaritis, K. (2003). Analysis of recommender systems algorithms. In *Proceedings of the 6th Hellenic European Conference on Computer Mathematics and its Applications (HERCMA-2003)*, Athens, Greece.

Vyas, V., & Pantel, P. (2009). Semi-automatic entity set refinement. In *Proceedings of NAACL-09*, Boulder, CO.

Weaver, W. (1955). Translation. In Locke, W., & Booth, D. (Eds.), *Machine Translation of Languages: Fourteen Essays*. MIT Press, Cambridge, MA.

Weeds, J., Weir, D., & McCarthy, D. (2004). Characterising measures of lexical distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*, pp. 1015–1021. Association for Computational Linguistics.

Wei, X., Peng, F., & Dumoulin, B. (2008). Analyzing web text association to disambiguate abbreviation in queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*, pp. 751–752, New York, NY. ACM.

Wen, J.-R., Nie, J.-Y., & Zhang, H.-J. (2001). Clustering user queries of a search engine. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*, pp. 162–168, New York, NY. ACM.

Widdows, D. (2004). *Geometry and Meaning*. Center for the Study of Language and Information, Stanford, CA.

Widdows, D., & Ferraro, K. (2008). Semantic vectors: A scalable open source package and online technology management application. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2008)*, pp. 1183–1190.

Witten, I. H., & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco.

Wittgenstein, L. (1953). *Philosophical Investigations*. Blackwell. Translated by G.E.M. Anscombe.

Wolfe, M. B. W., Schreiner, M. E., Rehder, B., Laham, D., Foltz, P. W., Kintsch, W., & Landauer, T. K. (1998). Learning from text: Matching readers and texts by latent semantic analysis. *Discourse Processes*, *25*, 309–336.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval*, *1*(1), 69–90.

Yuret, D., & Yatbaz, M. A. (2009). The noisy channel model for unsupervised word sense disambiguation. *Computational Linguistics*. Under review.

Zamir, O., & Etzioni, O. (1999). Grouper: a dynamic clustering interface to Web search results. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, *31*(11), 1361–1374.

Zhao, Y., & Karypis, G. (2002). Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pp. 515–524, McLean, Virginia.