Vector Space Models of Lexical Meaning

Stephen Clark

University of Cambridge Computer Laboratory
stephen.clark@cl.cam.ac.uk

A draft chapter for the Wiley-Blackwell *Handbook of Contemporary Semantics — second edition*, edited by Shalom Lappin and Chris Fox. This draft formatted on 13th September 2012.

1 Introduction

Much of this Handbook is based on ideas from Formal Semantics, in which the meanings of phrases or sentences are represented in terms of set-theoretic models. The key intuition behind Formal Semantics, very roughly, is that the world is full of objects; objects have properties; and relations hold between objects. Set-theoretic models are ideal for capturing this intuition, and have been succcessful at providing formal descriptions of key elements of natural language semantics, for example quantification.[1] This approach has also proven attractive for Computational Semantics – the discipline concerned with representing, and reasoning with, the meanings of natural language utterances using a computer. One reason is that the formalisms used in the set-theoretic approaches, e.g. first-order predicate calculus, have well-defined inference mechanisms which can be implemented on a computer (Blackburn & Bos, 2005).

The approach to natural language semantics taken in this chapter will be rather different, and will use a different branch of mathematics to the set theory employed in most studies in Formal Semantics, namely the mathematical framework of vector spaces and linear algebra. The attraction of using vector spaces is that they provide a natural mechanism for talking about distance and similarity, concepts from geometry. Why should a geometric approach to modelling natural language semantics be appropriate? There are many aspects of semantics, particularly lexical semantics, which require a notion of distance. For example, the meaning of the word *cat* is closer to the meaning of the word *dog* than the meaning of the word *car*. The modelling of such distances is now commonplace in Computational Linguistics, since many examples of language technology benefit from knowing how word meanings are related geometrically; for example, a search engine could expand the range of web pages being returned for a set of query terms by considering additional terms which are close in meaning to those in the query.

The meanings of words have largely been neglected in Formal Semantics, typically being represented as atomic entities such as dog', whose interpretation is to denote some object (or set of objects) in a set-theoretic model. In this chapter the meanings of words will be represented using vectors, as part of a high-dimensional "semantic space". The fine-grained structure of this space is provided by considering the contexts in which words occur in large corpora of text. Words can easily be compared for similarity in the vector space, using any of the standard similarity or distance measures available from linear algebra, for example the cosine of the angle between two vectors.

The hypothesis underlying distributional models of word meanings is the so-called distributional hypothesis: the idea that "Words that occur in similar contexts tend to have similar meanings" (Turney & Pantel, 2010). The next section discusses how this hypothesis has its roots in theoretical Linguistics.

[1] See the chapter on Generalized Quantifiers in this Handbook.

Turney & Pantel (2010) offer variations of this hypothesis which can be applied to linguistic units other than words, for example the "bag of words hypothesis" which can provide an indication of the meaning of documents and queries for the document retrieval problem (described in Section 2).

1.1 Distributional methods in Linguistics

In vector space models of word meaning, the set of contexts[2] in which a word occurs — or the *distribution* of the word's contexts — is considered key to deriving a suitable meaning representation; hence the term *distributional semantics* is often used to describe such models. Of course distributional techniques have a long history in theoretical linguistics. Harris (1954), in the tradition of the structural linguists, proposed that linguistic units, such as parts of speech, could be identified from corpora by observing the contexts in which the units occur. Perhaps the historical work most closely related to modern distributional semantics is that of Firth (1957), who was interested in the notion of collocation and how the distributional contexts of a word could be used to explain its behaviour. Firth was also interested in different word senses, arguing that the different senses of an ambiguous word could be revealed by looking at the different contexts in which the word occurs (Pulman, 2012). This idea was exploited by Schütze (1998) in his seminal paper on using distributional models for word sense disambiguation (described in Section 3.4). Finally, one classic piece of philosophical work that is often mentioned in the context of distributional semantics is Wittgenstein (1953). The link here is somewhat tenuous, since Wittgenstein was not concerned with the contexts of words in corpora, but rather the conventional, social nature of a whole language. However, Wittgenstein's slogan that "meaning is use" is certainly applicable, under some interpretation, to distributional semantics.

Pulman (2012) gives a more detailed account of how the historical use of distributional contexts in linguistics relates to modern distributional semantics, particularly in relation to the possibility of compositional distributional models (the question considered in Section 4.3 of this chapter).

A related discipline in which distributional semantic models have been studied is Cognitive Science. Such models have been successful at simulating a variety of semantic processing tasks, for example semantic priming (Lund & Burgess, 1996), episodic memory (Griffiths *et al.*, 2007) and text comprehension (Landauer & Dumais, 1997). In this chapter little attention will be paid to cognitive modelling; however, one interesting link with Section 4.3 is the perceived failure of distributional models in general, and connectionist models in particular, to provide a suitable account of compositionality in language (Fodor & Pylyshyn, 1988). Some further links with the cognitive science literature, particularly Smolensky (1990), will be made there.

[2] The definition of "context" varies according to the particular technique being used; Section 3.1 considers some alternatives.

1.2 Outline

The chapter will begin by describing how vector space models have been used for document retrieval – the task performed by internet search engines. These document-based models were not intially developed for explicitly modelling word meanings; rather they were developed to represent the meaning, or topic, of a whole document. However, document retrieval is a natural place to start since this application provided many of the key ideas in distributional semantics, and there is a natural progression from modelling documents using vector spaces to modelling the meanings of words.

Section 3 describes the main techniques used in building distributional models of word meanings, and considers some of the model parameters: the definition of context; the similarity measure used to compute the closeness of word vectors; and the weighting scheme used to reflect the fact that some contexts are more indicative of a word's meaning than others. This section will also consider the question of what lexical relations are being acquired using distributional techniques.

Finally, Section 4.3 considers a problem which is rapidly gaining interest in the Computational Linguistics community: how to enrich vector space models of lexical semantics with some notion of compositionality. This is an interesting problem from a practical perspective, since the ability to determine semantic similarity of phrases, or even whole sentences, would benefit language technology applications; but it is also interesting from a theoretical perspective, since a solution to the problem potentially offers a unification of the rich lexical representations from distributional semantics with the logic-based account of how meanings combine to give meanings of phrases and sentences.

The perspective from which this chapter is written is largely a Computational Linguistics one, which reflects both the expertise of the author and the fact that vector space models have received less attention in mainstream theoretical linguistics. However, it should be clear that there are questions and ideas in this chapter which will be of interest to linguists of all types: theoretical, computational, and cognitive.

2 Vector Space Models for Document Retrieval

The document retrieval problem in Information Retrieval (IR; Manning *et al.* (2008)) is as follows: given a query — typically represented as a set of query terms — return a ranked list of documents from some set, ordered by relevance to the query. Terms here can be words or lemmas, or multi-word units, depending on the lexical pre-processing being used.[3] The complete set of documents depends on the application; in the internet-search case it could be the whole web.

One of the features of most solutions to the document retrieval problem, and indeed Information Retrieval problems in general, is the lack of sophistication of the linguistic modelling employed: both the query and the documents are considered to be "bags of words", i.e. multi-sets in which the frequency of words is accounted for, but the order of words is not. From a linguistic perspective, this is a crude assumption (to say the least), since much of the meaning of a document is mediated by the order of the words, and the syntactic structures of the sentences. However, this simplifying assumption has worked surprisingly well, and attempts to exploit linguistic structure beyond the word level have not usually improved performance. For the document retrieval problem perhaps this is not too surprising, since queries, particularly on the web, tend to be short (a few words), and so describing the problem as one of simple word matching between query and document is arguably appropriate.

Once the task of document retrieval is described as one of word overlap between query and document, then a vector space model is a natural approach: the basis vectors of the space are words, and both queries and documents are vectors in that space (Salton *et al.*, 1975). The coefficient of a document vector for a particular basis vector, in the simplest case, is just the number of times that the word corresponding to the basis appears in the document. Queries are represented in the same way, essentially treating a query as a "pseudo-document". Measuring word overlap, or the similarity of a document vector $\overrightarrow{d}$ and query vector $\overrightarrow{q}$, can be achieved using the dot product:

$$\text{Sim}(\overrightarrow{d}, \overrightarrow{q}) = \overrightarrow{d} . \overrightarrow{q} \tag{1}$$

$$= \sum_i d_i \times q_i \tag{2}$$

where v_i is the ith coefficient of vector $\overrightarrow{v}$.

Figure 1 gives a simple example containing two short documents. (In practice, documents would typically contain many more sentences, and paragraphs, than this.) In this example the user is interested in finding documents describing the England cricketer, Matthew Hoggard, taking wickets against

[3] In this chapter *term* and *word* will be used interchangably, even though terms can be lemmas or multi-word units.

Term vocabulary: ⟨*England, Australia, Pietersen, Hoggard, run, wicket, catch, century, collapse*⟩

Document $d1$: *Australia collapsed as Hoggard took 6 wickets . Flintoff praised Hoggard for his excellent line and length .*

Document $d2$: *Flintoff took the wicket of Australia 's Ponting , to give him 2 wickets for the innings and 5 wickets for the match .*

Query q: { *Hoggard, Australia, wickets* }

$$\overrightarrow{q1} \,.\, \overrightarrow{d1} = \langle 0,1,0,1,0,1,0,0,0\rangle \,.\, \langle 0,1,0,2,0,1,0,0,1\rangle = 4$$

$$\overrightarrow{q1} \,.\, \overrightarrow{d2} = \langle 0,1,0,1,0,1,0,0,0\rangle \,.\, \langle 0,1,0,0,0,3,0,0,0\rangle = 4$$

Figure 1. Simple example of document and query similarity using the dot product, with term-frequency providing the vector coefficients. The documents have been tokenised, and word matching is performed between lemmas (so *wickets* matches *wicket*).

Australia, and so creates the query { *Hoggard, Australia, wickets* }. Here the query is simply the set of these three words. The vectors are formed by assuming the basis vectors given in the term vocabulary list at the top of the figure; so the coefficient for the basis vector *Hoggard*, for example, for the document vector $\overrightarrow{d2}$, is 2 (since *Hoggard* occurs twice in the corresponding document).

The point of this example is to demonstrate a weakness with using just term-frequency as the vector coefficients: all basis vectors count equally when calculating similarity. In this example, document $d2$ matches the query as well as $d1$, even though $d2$ does not mention Hoggard at all. The solution to this problem is to recognise that some words are more indicative of the meaning of a document (or query) than others. An extreme case is the set of function words: we would not want a query and document to be deemed similar simply because both contain instances of the word "the". Another useful intuition is that, if the task of document retrieval is to separate, or rank, the relevant documents from the non-relevant ones, given some query, then any term which appears in all documents will be useless in making this separation.

Continuing with the example, let us assume that the document set being searched contains documents describing cricket matches. Since *wicket* is likely to be contained in many such documents, let us assume that this term occurs in 100 documents in total. *Hoggard* is more specific in that it describes a particular England cricketer, so suppose this term occurs in only 5 documents. We would like to down-weight the basis vector for *wicket*, relative to *Hoggard*, since *wicket* is a less discriminating term than *Hoggard*. An obvious way to achieve this is to divide the term-frequency coefficient by the corresponding

$$\overrightarrow{q1} \,.\, \overrightarrow{d1} = \langle 0, 1, 0, 1, 0, 1, 0, 0, 0\rangle \,.\, \langle 0, 1/10, 0, 2/5, 0, 1/100, 0, 0, 1/3\rangle = 0.41$$

$$\overrightarrow{q1} \,.\, \overrightarrow{d2} = \langle 0, 1, 0, 1, 0, 1, 0, 0, 0\rangle \,.\, \langle 0, 1/10, 0, 0/5, 0, 3/100, 0, 0, 0/3\rangle = 0.13$$

Figure 2. Simple example of document and query similarity using the dot product, with term-frequency, inverse-document frequency providing the coefficients for the documents, using the same query and documents as Figure 1.

document frequency (the number of documents in which the term occurs), or equivalently multiply the term frequency by the *inverse document frequency* (IDF) (Sparck Jones, 1972). Figure 2 shows the simple example with IDF applied (assuming that the document frequency for *Australia* is 10 and *collapse* is 3). Document $d1$ is now a better match for the query than $d2$.

A useful intuition for the effect of IDF on the vector space is that it effectively "shrinks" those basis vectors corresponding to words which appear in many documents — the words with little discriminating power as far as the document retrieval problem is concerned — and emphasises those basis vectors corresponding to words which appear in few documents. Figure 6 in Section 3.2 demonstrates this effect for word models.[4]

Finally, there is one more standard extension to the basic model, needed to counter the fact that the dot product will favour longer documents, since these are likely to have larger word frequencies and hence a greater numerical overlap with the query. The extension is to normalise the document vectors (and the query) by length, which results in the cosine of the angle between the two vectors:

$$\text{Sim}(\overrightarrow{d}, \overrightarrow{q}) = \frac{\overrightarrow{d} \,.\, \overrightarrow{q}}{\|\overrightarrow{d}\| \|\overrightarrow{q}\|} \tag{3}$$

$$= \frac{\overrightarrow{d} \,.\, \overrightarrow{q}}{\sqrt{\sum_i d_i^2} \sqrt{\sum_i q_i^2}} \tag{4}$$

$$= \text{Cosine}(\overrightarrow{d}, \overrightarrow{q}) \tag{5}$$

where $\|\overrightarrow{v}\| = \sqrt{\sum_i v_i^2}$ is the Euclidean length of vector $\overrightarrow{v}$.

These main ideas of the vector space model form the basis of modern search engines – together with some additional machinery such as the use of PageRank to introduce a bias towards more "important" pages, as determined by the hyperlink structure of the web (Brin & Page, 1998). Approaches which appear more sophisticated, such as BM25 (Robertson & Zaragoza, 2009), are essentially twists on the basic approach. Even the recent language modelling

[4] Another solution to eliminate the effect of highly frequent words is to simply ignore them, via the use of a so-called *stop list*. In practice IR systems often employ both a stop list and IDF weighting.

	$d1$	$d2$
England	0	0
Australia	1/10	1/10
Pietersen	0	0
Hoggard	2/5	0/5
run	0	0
wicket	1/100	3/100
catch	0	0
century	0	0
collapse	1/3	0/3

Figure 3. Term-document matrix for the simple running example, using *tf-idf* weights but without length normalisation.

approach to document retrieval can be seen as implementing these ideas, but in a probabilistic setting (Zhai & Lafferty, 2004).

Using a weighting scheme based on the frequency of terms in a document, and the inverse of the number of documents in which a term occurs, leads to a *term frequency-inverse document frequency* (*tf-idf*) model. In practice, the weighting formula is usually based on some function of *tf* and *idf*; for example *df* is often implemented as a function of the log of the document frequency, in order to introduce a damping effect when the document frequencies get very high (Manning *et al.*, 2008). It is also the case that different weighting schemes can be applied to the query compared with the documents (note that IDF was not applied at all to the query vector in the simple example in Figure 2).

The reason for describing the document retrieval model in this section is that all the ideas from the basic approach — representing the linguistic units of interest as vectors; using words as basis vectors; normalising vectors by length; and favouring some basis vectors over others using IDF — are carried over, in some form, into distributional models of word meaning, described in Section 3. The next subsection considers the term-document matrix, a useful bridge between the document models from IR and the word models from distributional lexical semantics.

2.1 Term-Document Matrix

One useful way to think about the document vectors is in terms of a *term-document* matrix. This matrix is formed by treating each term or word vector as a row in the matrix, and each document vector as a column. Figure 3 shows the term-document matrix for our simple running example.

The main reason for introducing the term-document matrix is that it provides an alternative perspective on the co-occurrence data, which will lead to vectors for terms themselves. But before considering this perspective, it

is important to mention the potential application of *dimensionality reduction* techniques to the matrix, such as singular value decomposition (SVD). The use of SVD, or alternative dimensionality reduction techniques such as non-negative matrix factorisation (Van de Cruys, 2010) or random indexing (Sahlgren, 2006), will not be covered in any detail here, since the chapter has been written assuming only a rudimentary understanding of linear algebra. However, it is important to mention these techniques since they are an important part of the linear algebra toolbox which should be considered by any researchers working in this area. Chapter 18 of Manning *et al.* (2008) provides an excellent textbook treatment of matrix decompositions in the context of Information Retrieval.

The application of SVD to the term-document matrix was introduced by Deerwester *et al.* (1990), who called the method *Latent Semantic Analysis* (LSA), or *Latent Semantic Indexing* (LSI) in the context of IR. Since then there has been a huge literature on LSA. Very briefly, LSA factors the original term document matrix into three matrices, and then uses those three matrices to form a smaller, low-rank approximation to the original matrix. In practice, the reduction in size is usually substantial; e.g. from a term-document matrix with tens of thousands of documents and vocabulary terms, to a low-rank approximation with only a few hundred basis vectors for each document.

Turney & Pantel (2010) provide three useful perspectives on LSA: one, it can be seen as uncovering *latent meaning*, essentially clustering words along a small number — typically a few hundred — of semantic, or topical, dimensions, which are teased out of the co-occurrence data automatically. Two, LSA is performing noise reduction, using the dimensionality reduction to uncover the true signal generating the data, and filtering out the noise which inevitably arises from a statistical sample. And three, LSA can be seen as a method for discovering higher-order co-occurrence, i.e. recognising words as similar when they appear in *similar contexts*. Similarity of contexts can be defined recursively in terms of lower-order co-occurrence (Turney & Pantel, 2010). This last perspective, in particular, is getting us closer to a view in which the words themselves, rather than the documents, are the main focus. It is easy to see how the term-document matrix provides an insight into the similarity of documents (which we have exploited already): documents (columns) will be similar if they contain many of the same terms (rows). But we can also use the matrix as a way of obtaining term similarity: terms will be similar if they occur in the many of the same documents. Landauer & Dumais (1997) adopted this perspective and applied the resulting term similarities to the multiple-choice synonym questions from the TOEFL (Teaching English as a Foreign Language) test, obtaining promising results. The next section will generalise this idea of obtaining term similarity: terms will be similar if they occur in many of the same *contexts*.

3 Representing Word Meanings as Vectors

The term-document matrix introduced in the previous section gives us the basic structure for determining word similarity. There the intuition was that words or terms are similar if they tend to occur in the same documents. However, this is a very broad notion of word similarity, producing what we might call topical similarity, based on a coarse notion of context. The trick in arriving at a more refined notion of similarity is to think of the term-document matrix as a term-*context* matrix, where, in the IR case, context was thought of as a whole document. But we can narrow the context down to a sentence, or perhaps even a few words either side of the target word.[5]

Once the context has been shortened in this way, then a different perspective is needed on the notion of context. Documents are large enough to consider which words appear in the same documents, but once we reduce the context to a sentence, or only a few words, then similar words will tend not to appear in the same *instance* of a contextual window. Another intuition that is useful here is that (near-)synonymous words, such as *boat* and *ship*, will tend not to occur in the same sentence (or even document), since a writer will tend to use one of the alternatives, and not the other, in the same sentence.

The new perspective is to consider single words as contexts, and count the number of times that a context word occurs in the context of the target word. Sahlgren (2006) stresses the distinction between the two interpretations by describing one as using *syntagmatic* relations as context, and the other as using *paradigmatic* relations. Syntagmatically related words are ones that co-occur in the same text region, whereas paradigmatically related words are ones whose surrounding words are often the same (Sahlgren, 2006).

Figure 4 gives a simple example demonstrating the construction of a *word-word* (or *term-term*) co-occurrence matrix, based on paradigmatic relations with a single sentence as the context window. The example is somewhat contrived, but demonstrates the method of constructing a term-term matrix. Note that the target words — the words for which context vectors are calculated — do not have to be part of the term vocabulary, which provide the context. Determining which words are similar can be performed using the cosine measure (equation 5), as before.

In the example, *football* is similar in meaning to *soccer* since the context vector for *football* (the row corresponding to *football* in the term-term matrix) has a large numerical overlap with the context vector for *soccer*; i.e. many of the words surrounding instances of *football* — within a contextual window of a sentence — are the same as the words surrounding instances of *soccer*. A similar pattern is seen for *automobile* and *car*, but not for the other pairings of target words.

Once the move was made from syntagmatic to paradigmatic relations as context, then researchers began to consider alternatives to simply taking a

[5] *Target word* refers to the word for which we are attempting to obtain similar words.

An automobile is a wheeled motor vehicle used for transporting passengers .
A car is a form of transport, usually with four wheels and the capacity to carry around five passengers .
Transport for the London games is limited , with spectators strongly advised to avoid the use of cars .
The London 2012 soccer tournament began yesterday , with plenty of goals in the opening matches .
Giggs scored the first goal of the football tournament at Wembley , North London .
Bellamy was largely a passenger in the football match , playing no part in either goal .

Term vocab: ⟨*wheel, transport, passenger, tournament, London, goal, match*⟩

	wheel	*transport*	*passenger*	*tournament*	*London*	*goal*	*match*
automobile	1	1	1	0	0	0	0
car	1	2	1	0	1	0	0
soccer	0	0	0	1	1	1	1
football	0	0	1	1	1	2	1

$automobile\ .\ car = 4$
$automobile\ .\ soccer = 0$
$automobile\ .\ football = 1$
$car\ .\ soccer = 1$
$car\ .\ football = 2$
$soccer\ .\ football = 5$

Figure 4. A small example corpus and term vocabulary with the corresponding term-term matrix, with term-frequency as the vector coefficients. Each sentence provides a contextual window, and the sentences are assumed to have been lemmatised when creating the matrix.

fixed-word window containing the target word as context. One popular extension (Grefenstette, 1994; Curran, 2004; Pado & Lapata, 2007) is to take the words which are related syntactically to the target word, perhaps using the type of the relation as part of the definition of context. Section 3.1 will consider this alternative in more detail.

The term-term matrix can have dimensionality reduction techniques applied to it, such as SVD, as was the case for the term-document matrix. Some researchers do apply SVD, or some related technique, arguing that it produces word vectors which are less sparse and less affected by statistical noise. However, it is unclear whether SVD is beneficial in the general case. One potential

disadvantage of SVD is that the induced hidden dimensions are difficult to interpret, whereas basis vectors defined in terms of syntactic relations, for example, can be related to conceptual properties and given a psycholinguistic interpretation (Baroni *et al.*, 2010). Whether the potential for interpretation is important again depends on the task or application at hand.

Distributional semantics is a corpus-based, experimental discipline, but there has been little discussion so far of how the vector spaces are constructed in practice, and how they are evaluated. Section 3.3 will discuss some of these issues, after alternatives for the some of the main paramaters of vector space word models have been described.

3.1 Context

The previous example in Figure 4 uses what is often called a *window* method, where the contextual words for a partiular instance are taken from a sequence of words containing the target word. In the example, the window boundaries are provided by each sentence. When the window is as large as a sentence — or a paragraph or document — the relation that is extracted tends to be one of topical similarity, for example relating *gasoline* and *car*. If the intention is to extract a more fine-grained relation, such as synonymy, then a smaller, and more fine-grained, notion of context is appropriate.[6]

Curran (2004) investigates a range of context definitions for the task of automatic thesaurus extraction, focusing on the synonymy relation. For the window methods, he uses a relatively short window consisting of two words either side of the target word. The simplest window approach is to use the context words as basis vectors, and define the vector coefficients as (weighted) frequencies of the number of times each context word occurs in the contextual windows surrounding instances of the target word.

More fine-grained defintions are possible, even for the window method. One possibility is to pair a context word with a direction. Now the vector coefficients are weighted counts of the number of times the context word appears to the left, or right, of the target word, respectively. A further possibility is to take position into account, so that a basis vector corresponds to a context word appearing a particular number of words to the left or right of the target word. Whether such modifications improve the quality of the extracted relations is not always clear, and depends on the lexical relations that one hopes to extract. (The difficulty of evaluation will be discussed in Section 3.3.)

The next step in refining the context definition is to introduce some linguistic processing. One obvious extension to the window methods is to add part-of-speech tags to the context words. A more sophisticated technique is to only consider context words that are related syntactically to the target word,

[6] Of course *gasoline* and *car* are related semantically in a particular way — gasoline is something that is put in a car as fuel — but window methods would be unable to extract this relation in particular (excluding other semantic relations).

and to use the syntactic relation as part of the definition of the basis vector. Figure 5 shows how these various refinements pick out different elements of the target word's linguistic environment. The pipe or bar notation (|) is simply to create pairs, or tuples — for example pairing a word with its part-of-speech tag. The term *contextual element* is used to refer to a basis vector term which is present in the context of a particular instance of the target word.

The intuition for building the word vectors remains the same, but now the basis vectors are more complex. For example, in the grammatical relations case, counts are required for the number of times that *goal*, say, occurs as the direct object of the verb *scored*; and in an adjective modifier relation with *first*; and so on for all word-grammatical relation pairs chosen to constitute the basis vectors. The idea is that these more informative linguistic relations will be more indicative of the meaning of the target word.

The linguistic processing applied to the sentence in the example is standard in the Computational Linguistics literature. The part-of-speech tags are from the Penn Treebank tagset (Marcus *et al.*, 1993) and could be automatically applied using any number of freely available part-of-speech taggers (Brants, 2000; Toutanova *et al.*, 2003; Curran & Clark, 2003). The grammatical relations — expressing syntactic head-dependency relations — are from the Briscoe & Carroll (2006) scheme, and could be automatically applied using e.g. the RASP (Briscoe *et al.*, 2006) or C&C (Clark & Curran, 2007) parsers. Another standard practice is to lemmatise the sentence (Minnen *et al.*, 2001) so that the direct object of *scored*, for example, would be equated with the direct object of *score* or *scores* (i.e. each of these three word-grammatical relation pairs would correspond to the same basis vector).

For the window method, a 7-word window is chosen so that some informative words, such as *scored* and *football*, are within the context of the target word *goal* in this example sentence. In practice the size of the window is a parameter which must be set in some way (usually by trial and error, seeing which value produces the best performance according to whatever evaluation metric is being used to assess the output). Note that, for the simplest window method, *the* occurs twice in the context in the example; hence this particular instance of the target word *goal* would contribute a frequency of 2 to the coefficient of the basis vector corresponding to *the*.

The most detailed basis representation we have seen so far is based on single grammatical relations. But one may wonder whether it is possible to extend this further and use even more refined notions of context. For example, in the sentence in Figure 5, the target word *goal* is the direct object of *scored*, but *goal* is also linked to *Giggs*, since *Giggs* is the subject of the same verb. This extended notion of grammatical relation leads to the idea of *dependency paths*, described by Pado & Lapata (2007). Now the basis vectors correspond to whole sequences of grammatical relations, relating the target word and context word. Which paths to choose is a parameter of the approach, with the idea that some paths will be more informative than others; for example, it is possible to reach the second instance of *the* in the example sentence from

Giggs|NNP *scored*|VBD *the*|DT *first*|JJ *goal*|NN *of*|IN *the*|DT *football*|NN *tournament*|NN *at*|IN *Wembley*|NNP ,|, *North*|NNP *London*|NNP .|.

(`ncmod` _ *goal first*)
(`det` *goal the*)
(`ncmod` _ *tournament football*)
(`det` *tournament the*)
(`ncmod` _ *London North*)
(`dobj` *at Wembley*)
(`ncmod` _ *scored at*)
(`dobj` *of tournament*)
(`ncmod` _ *goal of*)
(`dobj` *scored goal*)
(`ncsubj` *scored Giggs* _)

Contextual elements for target word *goal* using a 7-word window method:
{*scored, the, first, of, football*}

Contextual elements with parts-of-speech:
{*scored*|VBD, *the*|DET, *first*|JJ, *of*|IN, *football*|NN}

Contextual elements with direction (L for left, R for right):
{*scored*|L, *the*|L, *first*|L, *of*|R, *the*|R, *football*|R}

Contextual elements with position (e.g. 1L is 1 word to the left):
{*scored*|3L, *the*|2L, *first*|1L, *of*|1R, *the*|2R, *football*|3R}

Contextual elements as grammatical relations:
{*first*|`ncmod`, *the*|`det`, *scored*|`dobj`}

Figure 5. Example sentence with part-of-speech tags from the Penn Treebank tagset (Marcus *et al.*, 1993) and grammatical relations from the Briscoe & Carroll (2006) scheme. Contextual elements for the target word *goal* are shown for various definitions of context.

goal (via *of* and *football*, assuming undirected dependency links). But this path would most likely be ruled out as uninformative. (See Pado & Lapata (2007) for how certain paths can be ruled out.)

One potential problem with this extended approach is data sparsity: since the basis vectors are so detailed, the counts for many combinations of target word and basis vector may be unreliable (or zero), even for very large corpora. Pado & Lapata (2007) overcome this difficulty by ignoring the label provided by the sequence; in the example sentence, *Giggs* would provide a contextual element for *goal*, but the basis vector would consist only of the word *Giggs*,

and not any of the grammatical relations. Again, whether discarding the labels improves performance is an empirical question, and provides yet another parameter to the system.

Pado and Laptata did find that the syntactically-enriched models outperformed the simple window method on a number of tasks from language processing and cognitive science (simulating semantic priming, detection of synonymy in the TOEFL tests, and ranking the senses of polysemous words in context). Curran also found that using grammatical relations as basis vectors generally outperformed the window methods for the task of synonymy extraction.

So far we have been assuming that all contextual elements are equally useful as basis vectors. However, this is clearly not the case: the word *the* provides very little information regarding the meaning of the word *goal*. One option is to employ a *stop list*, as for the document retrieval case, and simply ignore those contextual elements which are extremely frequent in a large corpus. A more principled approach, again borrowing ideas from document retrieval, is to *weight* the basis vectors. The next section describes some possible weighting schemes.

3.2 Weighting and Similarity

One simple method for weighting a basis vector is to divide the corresponding term frequency by the number of times that the term occurs in a large corpus, or, to borrow an idea from document retrieval, the number of documents that the term occurs in. Figure 6 demonstrates the effect of using Inverse Document Frequency (IDF) in this way for the extreme case of *the* as a basis vector. The effect is a little hard to capture in two dimensions, but the idea is that, in the vector space at the top of the figure, the vectors for *dog* and *cat* will be pointing much further out of the page — along the *the* basis — than in the vector space at the bottom. A useful intuition for the effect of IDF is that it effectively "shrinks" those basis vectors corresponding to highly frequent terms, reducing the impact of such bases on the position of the word vectors, and thereby reducing the amount of overlap on those bases when calculating word similarity.

One feature of IDF is that the shrinking effect applies in the same way to all target words (since IDF for a basis vector is independent of any target word). However, we may want to weight basis vectors differently for different target words. For example, the term *wear* may be highly indicative of the meaning of *jacket*, but less indicative of the meaning of *car*. Hence we would want to emphasise the basis vector for *wear* when building the vector for *jacket*, but emphasise other basis vectors — such as *gasoline* — when building the vector for *car*.

Curran (2004) uses collocation statistics to allow the weighting scheme to have this dependence on the target word. For example, *jacket* and *wear* will be highly correlated according to a collocation statistic because *jacket*

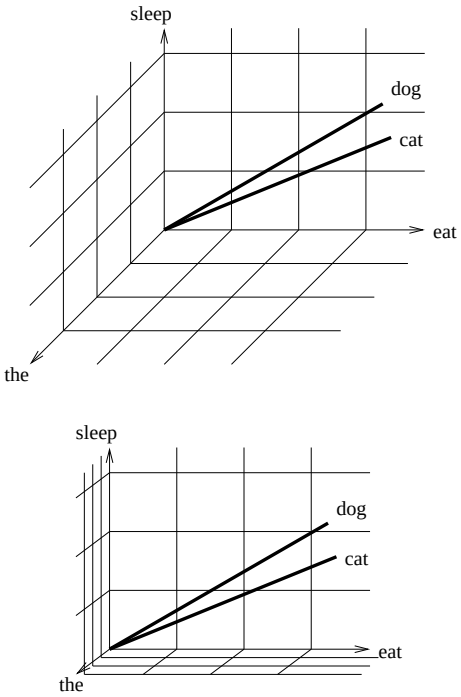


Figure 6. The effect of IDF on a simple example vector space.

co-occurs frequently with *wear* (relative to other basis vector terms). Here we are effectively using Sahlgren's syntagmatic relations to provide a weighting scheme to better capture paradigmatic relations. There are many collocation statistics which could be used here (Manning & Schutze (1999) provide a textbook treatment of some of the alternatives), and each statistic has parameters which must be set empirically (e.g. the size of the collocation window). Curran (2004) investigates a range of alternatives for the task of synonymy extraction, finding that some do perform better than others for his particular evaluation. Rather than try to summarise the results here, the reader is referred to Curran (2004) for the details, with the summary conclusion that using some form of weighting typically improves performance over simply using term frequencies.

We have described techniques for building word vectors; now we need a method of comparing them, in order to determine similarity of word meaning. Again we can borrow a key idea from document retrieval, namely the cosine similarity measure:

$$\mathrm{Sim}(\overrightarrow{w1}, \overrightarrow{w2}) = \frac{\overrightarrow{w1}.\overrightarrow{w2}}{\|\overrightarrow{w1}\|\|\overrightarrow{w2}\|} \tag{6}$$

$$= \frac{\overrightarrow{w1}.\overrightarrow{w2}}{\sqrt{\sum_i w1_i^2}\sqrt{\sum_i w2_i^2}} \quad (7)$$

$$= \text{Cosine}(\overrightarrow{w1}, \overrightarrow{w2}) \quad (8)$$

where $\overrightarrow{w1}$ and $\overrightarrow{w2}$ are vectors for words $w1$ and $w2$.

Note that the dot product in the numerator is calculating numerical overlap between the word vectors, and dividing by the respective lengths provides a length normalisation which leads to the cosine of the angle between the vectors. Normalisation is important because we would not want two word vectors to score highly for similarity simply because those words were frequent in the corpus (leading to high term frequency counts as the vector coefficients, and hence high numerical overlap).

The cosine measure is commonly used in studies of distributional semantics; there are alternatives, however. Rather than go into the details here, the reader is referred to Lin (1998) and Chapter 4 of Curran's thesis (Curran, 2004). It is difficult to reach a conclusion from the literature regarding which similarity measure is best; again this appears to depend on the application and which relations one hopes to extract.

3.3 Experiments

A key question when extracting similarity relations in practice is which corpus to use. The answer may depend on the application; for example, a query expansion system for document retrieval may build word vectors using web data or query logs; a system extracting synonymous terms for biomedical entities would use a corpus of biomedical research papers. Most research on distributional semantics has tended to use standard, freely available corpora such as the British National Corpus (BNC), which contains around 100 million words from a range of genres.

The size of the corpus is an important factor in the quality of the extracted relations, with the general message that more data is better data (a common theme in statistical natural language processing). Curran (2004) used a corpus containing over 2 billion words, made up from the BNC, the Reuters Corpus Volume 1 (Rose *et al.*, 2002), and much of the English news corpora from the Linguistic Data Consortium (LDC). It is worth noting that, even with the computing resources available today, 2 billion words is still large enough to make computational concerns a major factor in any experimental work, and the efficient creation and comparison of word vectors built from 2 billion words is a non-trivial computer science task. Agirre *et al.* (2009) used a web corpus containing roughly 1.6 *terawords*, many orders of magnitude larger than Curran's 2 billion word corpus. However, Agirre et al. had access to Google's distributed computing infrastructure, meaning that experiments could be run in a matter of minutes, even on corpora of this size. Such infrastructure is not generally available to academic researchers.

introduction: launch, implementation, advent, addition, adoption, arrival, absence, inclusion, creation, departure, availability, elimination, emergence, use, acceptance, abolition, array, passage, completion, announcement, ...

evaluation: assessment, examination, appraisal, review, audit, analysis, consultation, monitoring, testing, verification, counselling, screening, audits, consideration, inquiry, inspection, measurement, supervision, certification, checkup, ...

context: perspective, significance, framework, implication, regard, aspect, dimension, interpretation, meaning, nature, importance, consideration, focus, beginning, scope, continuation, relevance, emphasis, backdrop, subject, ...

similarity: resemblance, parallel, contrast, flaw, discrepancy, difference, affinity, aspect, correlation, variation, contradiction, distinction, divergence, commonality, disparity, characteristic, shortcoming, significance, clue, hallmark, ...

method: technique, procedure, means, approach, strategy, tool, concept, practice, formula, tactic, technology, mechanism, form, alternative, standard, way, guideline, methodology, model, process, ...

result: consequence, outcome, effect, finding, evidence, response, possibility, kind, impact, datum, reason, extent, report, example, series, aspect, account, amount, degree, basis, ...

conclusion: finding, outcome, interpretation, assertion, assessment, explanation, judgment, assumption, decision, recommendation, verdict, completion, inference, suggestion, result, answer, view, comment, testimony, argument, ...

Figure 7. Ranked lists of synonyms for the target words in bold from Curran's system.

When discussing experimental distributional semantics, it is instructive to consider some example output. Figure 7 shows the top 20 extracted synonyms for a number of example target words, taken from Curran (2004). The lists for each target word are ordered in terms of similarity, so *launch*, for example, is the closest synonym to *introduction* according to Curran's system. The example target words are somewhat abstract in nature, since these are the titles of Curran's thesis chapters.

The output demonstrates features which are common to many language resources automatically extracted from corpora. First, there are many automatically derived synonyms which might not immediately occur to a human creating such lists, such as *advent* for *introduction*. Second, some of the ouput is incorrect; if the task is to extract synonyms, then *elimination*, for example,

is more of an antonym of *introduction* than a synonym. Third, automatically extracted resources will always contain anomalous content which is difficult to explain, such as *array* as a highly ranked synonym for *introduction*.

Evaluation methods for such resources are typically of two types: intrinsic and extrinsic. Intrinsic evaluation involves comparing the resource directly against a manually created gold standard, such as an existing thesaurus. The problem with this approach is that automatic methods are designed to oversome some of the limitations of manually created resources, such as lack of coverage. So it may be that the automatic system correctly posits a synonym for some target word which was not considered by the lexicographer creating the gold standard.

Curran (2004) used this intrinsic method, but pooled together the output from a number of manually created thesauri, such as Roget's and Macquarie, in an attempt to offset the coverage problem. A standard measure for this evaluation is *precision at rank*, where the precision (accuracy) of the output for each target word is measured at various points in the synonym ranking list for each word. A point is scored for each synonym extracted by the automatic system which is also in the gold standard thesaurus. An overall score is obtained by averaging the scores for all the target words in the test set. Curran (2004) carried out such an evaluation for a test set of 300 nouns, selected to contain a range of concrete and abstract nouns, and to cover a range of corpus frequencies. Curran reports a score of 68% precision at rank 1, meaning that 68% of the top-ranked synonyms for the 300 target nouns in the test set were also in the manually created gold standard. Precision was 55% at rank 5, 45% at rank 10, and 35% at rank 20.

Extrinsic evaluation involves applying the extracted resource to some task or language processing application, and observing performance on the task. Psycholinguistic tasks have been used for this purpose. For example, Pado & Lapata (2007) use semantic priming (Lowe & McDonald, 2000), modelling the reading time for prime-target pairs using the semantic similarity between the prime and target. The intention is that the similarity of a word and related prime, e.g. *pain* and *sensation*, will be higher than a word and an unrelated prime. Furthermore, the distance between the prime and target should correlate with the corresponding reading times from the original psycholingusitic study (Hodgson, 1991). Pado and Lapata found that their dependency-based method for building word vector spaces (described in Section 3.1) produced higher correlations than the window-based method.

Pado & Lapata (2007) also use the TOEFL (Teaching of English as a Foreign Language) Tests, following Landauer & Dumais (1997). Here the task is to determine, for a number of target words, the closest synonym from a choice of four alternatives. Pado and Lapata give the following example:

You will find the office at the main **intersection**.
(a) place (b) crossroads (c) roundabout (d) building

where the task is to determine that *crossroads* is the closest synonym from the alternatives to *intersection*, for this particular sentence. The method applied was to simply pick the alternative closest to the target in the automatically constructed semantic space. Pado and Lapata again showed that their dependency-based method performed better than the window-based method, scoring 73% vs. 61% accuracy.

Other extrinsic methods exist in the literature; for example Pado & Lapata (2007) perform a third task of ranking word senses in context. Which method is most appropriate depends on the research goals: if the goal is to model how humans process language, then using psycholinguistic data would be appropriate; if the goal is to improve language processing applications, then the automatically extracted resource should be embedded in such applications; if the goal is to simply build a more representative thesaurus, then the intrinsic evaluation may be most appropriate.

3.4 Discussion

One issue that has been glossed over so far is the question of which lexical semantic relations are being acquired with distributional techniques. Curran (2004) explicitly describes his work as synonymy extraction, but it is clear from the examples in Figure 7 that Curran's system is extracting more than just synonyms. Not only are there examples of antonyms in the output, but also hyponyms (e.g. Curran's system finds *subsidiary* as closely related to *firm*, where a subsidiary is arguably a kind of firm); and also examples where the two words are semantically related in other ways.

Which relations are appropriate again depends on the task or application at hand. For example, if the task is query expansion for a search engine, it may not be unreasonable to consider lexical relations in addition to synonymy. If a user query contains the term *firm*, it may be helpful to expand the query with additional terms referring to kinds of firm, such as *subsidiary* (since if a user wishes to retrieve documents about firms, the user will most likely be interested in documents about subsidiaries, also).

There is some work on attempting to identify antonyms, and also analogies and associations, as separate from synonyms (Turney, 2008). There is also a huge literature on pattern-based techniques for identifying hyponyms, following the original insight of Hearst (1992) that hyponyms could be automatically extracted from text by searching for patterns such as *X such as Y*, for example *computers such as the Mac and PC* (allowing Mac and PC to be extracted as kinds of computers).[7] The pattern-based approaches are somewhat orthogonal to the distributional methods described in this chapter, and perhaps more grounded in a clever insight rather than any underlying theory.

[7] The previous sentence itself is such an example, allowing *X such as Y* to be extracted as a kind of pattern.

Turney (2006) makes the useful distinction between *attributional similarity* and *relational similarity*, arguing that there is some confusion in the distributional semantics literature regarding terminology. Attributional similarity is defined as correspondence between attributes, and relational similarity as correspondence between relations (Turney cites Medin *et al.* (1990) in this context). Synonyms are cases of words with a high degree of attributional similarity. Section 3 described methods for deriving attributional similarity.

Relational similarity extends the ideas in this chapter to capture analogies, such as *mason* has a similar relation to *stone* as *carpenter* to *wood*. Another distinction made in the literature is between words that are *semantically associated*, such as *bee-honey*, and words that are *semantically similar*, such as *deer-pony* (Turney, 2006). Turney argues that these are both cases of attributional similarity. Turney also argues that the term *semantic relatedness*, argued by Budanitsky & Hirst (2006) to capture a different relation to semantic similarity, is the same as attributional similarity. Our summary conclusion here is that distributional models can capture a variety of semantic relations, depending on how the various model parameters are defined, and it is important to be clear what relations one hopes to extract with a particular method.

Another issue that has been glossed over is that of word sense. Curran (2004) conflates the various senses of a word, so that the gold standard synonyms for *company*, for example, against which the output of his automatic system are compared, include synonyms for the companionship sense of *company*, the armed forces sense, the gathering sense, and so on. One piece of work in distributional semantics which explicitly models senses is Schütze (1998). Schütze uses a two-stage clustering method in which the first stage derives word vectors using the methods described in Section 3 (specifically using the window method with a window size of 50). Then, for a particular target word in context, each of the word vectors *for the context words* are added together (and divided by the number of context words) to derive a centroid for the particular context. The effect of deriving this second-order context vector is that the contextual words will act as word sense disambiguators, and when added together will emphasise the basis vectors pertaining to the particular sense.

Schütze gives the example of *suit*. Consider an instance of *suit* in a clothing, rather than legal, context, so that it is surrounded by words such as *tie*, *jacket*, *wear*, and so on. Of course these context words are potentially ambiguous as well, but the effect of adding them together is to emphasise those basis vectors which most of the context words have in common, which in this case is basis vectors relating to clothes.

In practice the second-order context vectors for a particular sense are obtained by clustering the context vectors for all instances of a target word (where, in Schütze's case, the number of clusters or senses needs to be known in advance). Then, given a particular instance of a target word, the word can

be sense disambiguated by calculating the 2nd-order centroid context vector for that instance, and calculating which cluster the centroid is closest to.

Despite the success of Schütze's method in performing word sense disambiguation, it is still the case that much work in distributional semantics ignores word senses (Pulman, 2012). To repeat what is becoming a common theme in this chapter, whether distinguishing word senses is useful in distributional semantics depends on the task or application at hand. There is also a long ongoing debate about whether word sense disambiguation is useful more generally for language processing applications (e.g. Chan *et al.* (2007)).

Finally, there are a number of ways in which the basic distributional hypothesis used in this chapter — that "Words that occur in similar contexts tend to have similar meanings" (Turney & Pantel, 2010) — has been entended. The work by Turney (2006) on automatically extracting analogies from text has already been mentioned. In an earlier influential paper, Lin & Pantel (2001) extract similar *patterns*, such as *X wrote Y* and *X is the author of Y*. The extended distributional hypothesis used to extract the patterns is the idea that "Patterns that co-occur with similar pairs tend to have similar meanings" (Turney & Pantel, 2010; Lin & Pantel, 2001). The two sets of instances of *X* in *X wrote Y* and *X is the author of Y*, derived from a large corpus, will have a large overlap (consisting of names of authors, for example); likewise for *Y*. The large overlap for both argument slots in the pattern can be used to infer similarity of pattern.

The next section considers a nascent but rapidly growing research area in which distributional techniques are being extended to capture similarity of phrases, and even whole sentences.

4 Compositional Vector Space Models

This section describes a recent development in Computational Linguistics, in which distributional, vector-based models of meaning have been given a compositional treatment allowing the creation of vectors for larger phrase, and even sentence, meanings. The bulk of the section will be a presentation of the theoretical framework of Coecke *et al.* (2010), which has been implemented in Grefenstette *et al.* (2011) and Grefenstette & Sadrzadeh (2011), in a form designed to be accessible to linguists not familiar with the mathematics of Category Theory on which the framework is based. The reason for focusing on the Coecke *et al.* (2010) work is that it has interesting links with existing compositional frameworks in linguistics, in particular Montague semantics (Dowty *et al.*, 1981).

One way of stating the problem is that we would like a procedure which, given vectors for each word in a phrase or sentence, combines the vectors in some way to produce a single vector representing the meaning of the whole phrase or sentence. Why might such a procedure be desirable? The first reason is that considering the problem of compositionality in natural language from a geometric viewpoint may provide an interesting new perspective on the problem. Traditionally, compositional methods in natural language semantics, building on the foundational work of Montague, have assumed the meanings of words to be given, and effectively atomic, without any internal structure. Once we assume that the meanings of words are vectors, with significant internal structure, then the problem of how to compose them takes on a new light.

A second, more practical reason is that language processing applications would benefit from a framework in which the meanings of whole phrases and sentences can be easily compared. For example, suppose that a sophisticated search engine is issued the following query: *Find all car showrooms with sales on for Ford cars.* Suppose futher that a web page has the heading *Cheap Fords available at the car salesroom.* Knowing that the above two sentences are similar in meaning would be of huge benefit to the search engine. Further, if the two sentence meanings could be represented in the same vector space, then comparing meanings for similarity is easy: simply use the cosine measure between the sentence vectors, as was used for word vectors.

One counter-argument to the above example might be that compositionality is not required in this case, in order to determine sentence similarity, only similarity at the word level. For this example that may be true, but it is uncontroversial that sentence meaning is mediated by syntactic structure. To take another search engine example, the query *A man killed his dog*, entered into Google on January 5, 2012, from the University of Cambridge Computer Laboratory, returned a top-ranked page with the snippet *Dog shoots man* (as opposed to *Man shoots dog*), and the third-ranked page had the snippet *The*

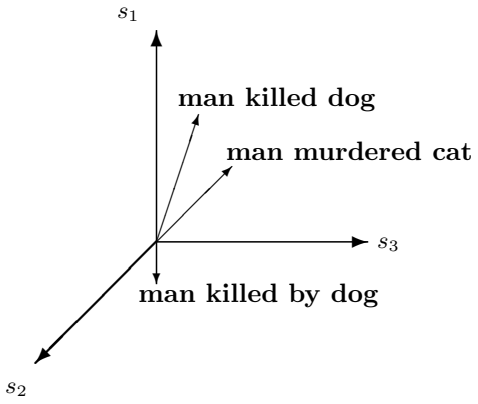


Figure 8. Example vector space for sentence meanings

Man who Killed His Friend for Eating his Dog After it was Killed ....[8] Of course the order of words matters when it comes to sentence meaning.

Figure 8 shows the intuition behind comparing sentence vectors. The framework described in this section will provide a mechanism for creating vectors for sentences, based on the vectors for the words, so that *man killed dog* and *man murdered cat* will (ideally) be relatively close in the "sentence space", but crucially *man killed by dog* will be located in another part of the space (since in the latter case it is the animal killing the man, rather than vice versa). Note that, in the sentence space in the figure, no commitment has been made regarding the basis vectors of the sentence space (s_1, s_2 and s_3 are not sentences, but unspecified basis vectors). In fact, the question of what the basis vectors of the sentence space should be is not answered by the compositional framework, but is left to the model developer to answer. The mathematical framework simply provides a compositional device for combining vectors, assuming the sentence space is given.

The rest of this section is structured as follows. Section 4.1 considers some general questions related to the idea of having a vector-based, distributional representation for sentence meanings. Section 4.2 briefly describes some existing work on this topic. Finally, Section 4.3 describes the theoretical framework of Coecke *et al.* (2010).

4.1 Distributional Sentence Representations

The intuition underlying vector-based representations for words is clear, and exemplified in the distributional hypothesis from Section 1: "Words that occur in similar contexts tend to have similar meanings" (Turney & Pantel, 2010). However, it is not clear how to extend this hypothesis to larger phrases

[8] Thanks to Mehrnoosh Sadrzadeh for this example.

or sentences. Do we want to say that sentences have similar meanings when they occur in similar contexts? Baroni & Zamparelli (2010) do carry over the intuition to phrases, but only adjective noun pairs, for which the intuition arguably still holds. Clarke (2008) develops a theoretical framework underpinning all distributional semantic models, and extends the intuition to linguistic units of any size, including sentences.

Whether the meaning of a sentence can be represented or determined by its contexts is unclear, and it could be argued that such a proposal is in opposition to the notion of compositionality, for which the meaning of a sentence is determined by the meanings of its parts and how those parts are combined (Werning *et al.*, 2012), not determined by the contexts in which the sentence *as a whole* is found. In this chapter we will make no commitment to the theoretical status of a sentence space, only the idea that comparing sentences for meaning similarity, and representing sentence meanings in a vector space, is a sensible enterprise.

A related question concerns what a vector-based sentence representation is providing a semantics *of*. One answer is that it is providing a semantics of *similarity*. However, whether traditional features of natural language semantics, such as logical operators, quantification, inference, and so on, can be integrated into the vector-based setting, is an open question. There is some preliminary work in this direction, e.g. (Preller & Sadrzadeh, 2009; Clarke, 2008; Widdows, 2004), and the question is creating substantial interest in the computational linguistics community; see e.g. the 2013 workshop Towards a Formal Distributional Semantics, attached to the 10th International Conference on Computational Semantics.

4.2 Existing Vector Composition Methods

Much of the work in Information Retrieval simply uses vector addition whenever word vectors need to be combined to obtain a distributional representation for larger linguistic units such as sentences or documents (e.g. Landauer & Dumais (1997)). Mitchell & Lapata (2008) compare vector addition with a number of other binary vector operations, such as pointwise multiplication and tensor product, on the task of disambiguating verb meanings in the context of a noun subject. Here both the verb and noun are represented using context vectors, using the window method from Section 3.1. Pointwise multiplication is found to perform best on this task, perhaps because the multiplication is having the effect of emphasising those contextual elements that the verb and noun have in common, and thereby performing sense disambiguation, as Pulman (2012) argues.

As mentioned in the introduction, the cognitive science literature already contains a large body of work addressing a similar question to the main question of this section: how can distributional, connectionist representations be given a compositional treatment to reflect the compositional nature of language? One researcher who has addressed this question in detail is Smolensky

(1990), who argues for the tensor product operator as the way of binding predicate vectors to their arguments. The compositional framework in Section 4.3 also uses a tensor product representation, but there are two main differences with the earlier cognitive science work. First, the distributional representations used in cognitive science tend to follow the neural network model, in which the basis vectors are effectively induced automatically from the data, rather than specified in advance in terms of linguistic contextual elements. Second, one innovative aspect of the compositional framework described below is that tensors are used to represent relational word types, such as verbs and adjectives, which are then "reduced" when combined with an argument. In contrast, Smolensky uses the tensor product operation as a way of binding a predicate to its argument, so that the resulting representation is, in some sense, bigger, rather than smaller, than each combining element.

The closest existing work to the framework described below is Baroni & Zamparelli (2010), in that Baroni & Zamparelli use the syntactic type of an adjective as motivation for a matrix representation in the distributional semantics. Adjective noun combinations are obtained using matrix multiplication of the noun vector by the adjective. Another innovation in Baroni & Zamparelli (2010), similar to an idea in Guevara (2011), is to learn the matrix automatically from data using supervised machine learning. Training data for the learning is provided by the contexts of adjective noun combinations attested in corpora. So one way of viewing the learning is as a way of overcoming sparse data: with enough data the context vectors for adjective noun combinations could be simply obtained from corpora, with no compositional operation required. Whether this is in the spirit of compositional semantics, as discussed briefly in Section 4.1, is open to debate. It is also unclear whether contexts for linguistic units larger than adjective noun combinations could be obtained in the same way.

This section has only described some of the growing body of work looking at how to combine vector representations of word meanings. Some additional work includes Widdows (2008); Zanzotto *et al.* (2010); Clarke (2012). The next section describes a framework, which, in contrast to the work mentioned above, provides a recipe for constructing phrase and sentence vectors in step with a syntactic, type-driven process.

4.3 The Compositional Framework

A key idea underlying the vector-based compositional framework of Coecke *et al.* (2010) is that *syntax drives the compositional process*, in much the same way that it does in Montague semantics (Dowty *et al.*, 1981). Another key idea borrowed from formal semantics is that the syntactic and semantic descriptions will be *type-driven*, reflecting the fact that many word types in natural language, such as verbs and adjectives, have a relation, or functional, role. In fact, the syntactic formalism assumed here will be a variant of Categorial Grammar, which is the grammatical framework also used by Montague.

The next section describes pregroup grammars, which provide the syntactic formalism used in Coecke *et al.* (2010). However, it should be noted that the use of pregroup grammars is essentially a mathematical expedient, and it is likely that other type-driven formalisms, for example Combinatory Categorial Grammar (Steedman, 2000), can be accommodated in the compositional framework. The grammar formalism is described in some detail here because the link between type-driven syntax and semantics is a key part of the compositional framework.

The following section shows how the use of syntactic functional types leads naturally to the use of tensor products for the meanings of words such as verbs and adjectives, and then an example sentence space is provided, giving some intuition for how to compose a tensor product with one of its "arguments" – effectively providing the analogue of function application in the semantic vector space.

Syntactic Types and Pregoup Grammars

The key idea in any form of Categorial Grammar is that all grammatical constituents correspond to a syntactic type, which identifies a constituent as either a function, from one type to another, or as an argument (Steedman & Baldridge, 2011). Combinatory Categorial Grammar (CCG) (Steedman, 2000), following the original work of Lambek (1958), uses *slash operators* to indicate the directionality of arguments. For example, the syntactic type (or *category*) for a transitive verb such as *likes* is as follows:

$$likes := (S\backslash NP)/NP$$

The way to read this category is that *likes* is the sort of verb which first requires an NP argument to its right (note the outermost slash operator pointing to the right), resulting in a category which requires an NP argument to its left (note the innermost slash operator pointing to the left), finally resulting in a sentence (S). Categories with slashes are known as *complex* categories; those without slashes, such as S and NP, are known as *basic*, or *atomic*, categories.

A categorial grammar lexicon is a mapping from words onto sets of possible syntactic categories for each word. In addition to the lexicon, there is a small number of rules which combine the categories together. In classical categorial grammar, there are only two rules, forward ($>$) and backward ($<$) application:

$$X/Y \;\; Y \Rightarrow X \quad (>) \tag{9}$$

$$Y \;\; X\backslash Y \Rightarrow X \quad (<) \tag{10}$$

These rules are technically rule schemata, in which the X and Y variables can be replaced with any category. Combinatory Categorial Grammar adds a number of additional rules, such as function composition and type-raising, which can increase the power of the grammar from context-free to mildly context-sensitive (Weir, 1988).

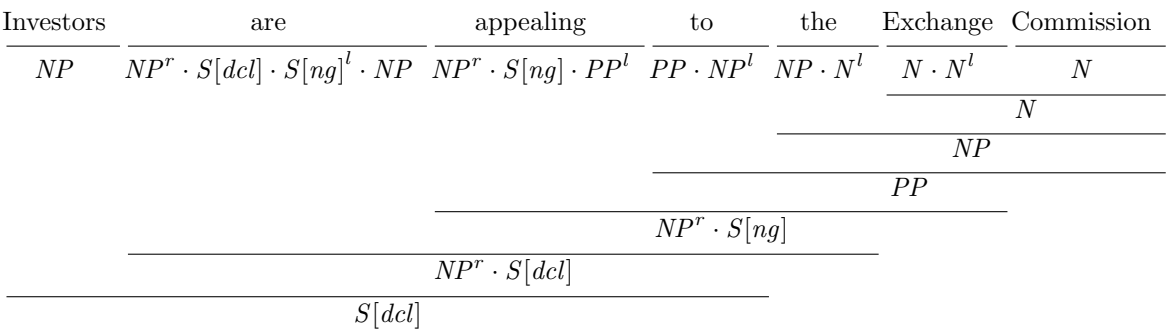


Figure 9. Example pregroup derivation

The mathematical move in pregroup grammars, a recent incarnation of categorial grammar due to Lambek (2008), is to replace the slash operators with different kinds of categories (*adjoint* categories), and to adopt a more algebraic, rather than logical, perspective compared with the original work (Lambek, 1958). The category for a transitive verb now looks as follows:

$$likes := NP^r \cdot S \cdot NP^l$$

The first difference to notice is notational, in that the order of the categories is different: in CCG the arguments are ordered from the right in the order in which they are cancelled; pregroups use the type-logical ordering (Moortgat, 1997) in which left arguments appear to the left of the result, and right arguments appear to the right.

The key difference is that a left argument is now represented as a *right adjoint*: NP^r, and a right argument is represented as a *left adjoint*: NP^l; so the adjoint categories have effectively replaced the slash operators in traditional categorial grammar. One potential source of confusion is that arguments to the left are right adjoints, and arguments to the right are left adjoints. The reason is that the "cancellation rules" of pregroups state that:

$$X \cdot X^r \rightarrow 1 \tag{11}$$

$$X^l \cdot X \rightarrow 1 \tag{12}$$

That is, any category X cancels with its right adjoint to the right, and cancels with its left adjoint to the left. Figure 9 gives the pregroup derivation for an example newspaper sentence, using the CCG lexical categories from CCGbank (Hockenmaier & Steedman, 2007) translated into pregroup types.[9]

[9] Pregroup derivations are usually represented as "reduction diagrams" in which cancelling types are connected with a link, similar to dependency representations in computational lingusitics. Here we show a categorial grammar-style derivation.

Mathematically we can be more precise about the pregroup cancellation rules:

> A pregroup is a partially ordered monoid[10] in which each object of the monoid has a left and right adjoint subject to the cancellation rules above, where $\rightarrow$ is the partial order, $\cdot$ is the monoid operation, and 1 is the unit object of the monoid.

In the linguistic setting, the objects of the monoid are the syntactic types; the associative monoid operation ($\cdot$) is string concatenation; the identity element is the empty string; and the partial order ($\rightarrow$) encodes the derivation relation. Lambek (2008) has many examples of linguistic derivations, including a demonstration of how iterated adjoints, e.g. NP^{ll}, can deal with interesting syntactic phenomena such as object extraction.

There are two key ideas from the syntactic description which will carry over to the vector-based semantics. First, linguistic constituents are represented using syntactic types, many of which are functional, or relational, in nature. Second, there is a mechanism in the pregroup grammar for combining a functional type with an argument, using the adjoint operators and the partial order (effectively encoding cancellation rules). Hence, there are two key questions for the semantic analysis: one, for each syntactic type, what is the corresponding semantic type in the world of vector spaces? And two, once we have the semantic types represented as vectors, how can the vectors be combined to encode a "cancellation rule" in the semantics?

Semantic Types and Tensor Products

The question we will now answer is: what are the semantic types corresponding to the syntactic types of the pregroup grammar? We will use the type of a transitive verb in English as an example, although in principle the same mechanism can be applied to all syntactic types.

The syntactic type for a transitive verb in English, e.g. *likes*, is as follows:

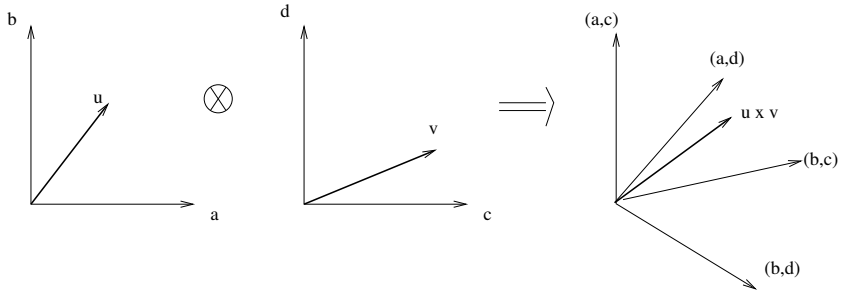
$$likes := NP^r \cdot S \cdot NP^l$$

Let us assume that noun phrases live in the vector space $\mathbf{N}$ and that sentences live in the vector space $\mathbf{S}$. We have methods available for building the noun space, $\mathbf{N}$, detailed earlier in the chapter; how to represent $\mathbf{S}$ is a key question for this whole section — for now we simply assume that there is such a space.

Following the form of the syntactic type above, the semantic type of a transitive verb — i.e. the vector space containing the vectors for transitive verbs such as *likes* — is as follows:

$$\overrightarrow{likes} \in \mathbf{N} \cdot \mathbf{S} \cdot \mathbf{N}$$

[10] A monoid is a set, together with an associative binary operation, where one of the members of the set is an identity element with respect to the operation.



$$(u \otimes v)_{(a,d)} = u_a \; . \; v_d$$

Figure 10. The tensor product of two vector spaces; $(u \otimes v)_{(a,d)}$ is the coefficient of $(u \otimes v)$ on the (a, d) basis vector

Now the question becomes what should the monoidal operator $(\cdot)$ be in the vector-space case? Coecke *et al.* (2010) use category theory to motivate the use of the tensor product as the monoidal operator which binds the individual vector spaces together:

$$\overrightarrow{likes} \;\in\; \mathbf{N} \otimes \mathbf{S} \otimes \mathbf{N}$$

Figure 10 shows two vector spaces being combined together using the tensor product. The vector space on the far left has two basis vectors, a and b, and is combined with a vector space also with two basis vectors, c and d. The resulting tensor product space has $2 \times 2 = 4$ basis vectors, given by the cartesian product of the basis vectors of the individual spaces $\{a, b\} \times \{c, d\}$. Hence basis vectors in the tensor product space are pairs of basis vectors from the individual spaces; if three vector spaces are combined, the resulting tensor product space has basis vectors consisting of triples; and so on.

Figure 10 also shows two individual vectors, u and v, being combined $(u \otimes v)$. The expression at the bottom of the figure gives the coefficient for one of the basis vectors in the tensor product space, (a, d), for the vector $u \otimes v$. The recipe is simple: take the coefficient of u on the basis vector corresponding to the first element of the pair (coefficient denoted u_a), and multiply it by the coefficient of v on the basis vector corresponding to the second element of the pair (coefficient denoted v_d). Combining vectors in this way results in what is called a *simple* or *pure* vector in the tensor product space.

One of the interesting properties of the tensor product space is that it is much larger than the set of pure vectors; i.e. there are vectors in the tensor product space in Figure 10 which cannot be obtained by combining vectors u and v in the manner described above. It is this property of tensor products which allows the representation of *entanglement* in quantum mechanics

(Nielsen & Chuang, 2000), and leads to *entangled* vectors in the tensor product space. Coecke *et al.* (2010) argue that it is also this property which allows the *interactive* nature of the verb to be captured, in terms of how a transitive verb interacts with a subject and object.

An individual vector for a transitive verb can now be written as follows (Coecke *et al.*, 2010):

$$\Psi = \sum_{ijk} C_{ijk} \left(\overrightarrow{n_i} \otimes \overrightarrow{s_j} \otimes \overrightarrow{n_k} \right) \in \mathbf{N} \otimes \mathbf{S} \otimes \mathbf{N} \tag{13}$$

Here n_i and n_k are basis vectors in the noun space, $\mathbf{N}$; s_j is a basis vector in the sentence space, $\mathbf{S}$; $\overrightarrow{n_i} \otimes \overrightarrow{s_j} \otimes \overrightarrow{n_k}$ is alternative notation for the basis vector in the tensor product space (i.e. the triple $\langle \overrightarrow{n_i}, \overrightarrow{s_j}, \overrightarrow{n_k} \rangle$); and C_{ijk} is the coefficient for that basis vector.

The intuition we would like to convey is that the vector for the verb is relational, or functional, in nature, as it is in the syntactic case. A similar, and useful, intuition lies behind the use of matrices to represent the meanings of adjectives in Baroni & Zamparelli (2010). Baroni & Zamparelli argue that an adjective is functional in nature, taking a noun (or noun phrase) as argument and returning another noun. A natural way to represent such a function in vector spaces is via a linear map, represented as a matrix. One way to understand the (Coecke *et al.*, 2010) framework is that it is an extension of Baroni & Zamparelli (2010) to cover all functional types, not just adjectives. The generalisation from adjectives to transitive verbs, for example, is to move from a matrix — in linear algebra known as a tensor of order 2 — to a "cube", which is a tensor of order 3. A cube is required to represent a transitive verb since three indices $(\overrightarrow{n_i}, \overrightarrow{s_j}, \overrightarrow{n_k})$ are needed to specify an element of the order 3 transitive verb space.

Informally, the expression in (13) for the verb vector can be read as follows:

> The vector for a transitive verb can be thought of as a function, which, given a particular basis vector from the subject, n_i, and a particular basis vector from the object, n_k, returns a value C_{ijk} for each basis vector s_j in the sentence space.

We have now answered one of the key questions for this section: what is the semantic type corresponding to a particular syntactic type? The next section answers the remaining question: how can the transitive verb in (13) be combined with instances of a subject and object to give a vector in the sentence space?

Composition for an Example Sentence Space

In this section we will use an example sentence space which can be thought of as a "plausibility space". Note that this is a fictitious example, in that no such space has been built and no suggestions will be made for how it might be built; however, it is a useful example because the sentence space is small, with only

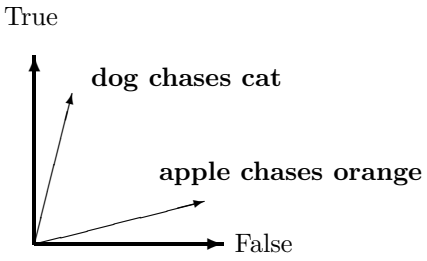


Figure 11. An example "plausibility space" for sentences

	fluffy	run	fast	aggressive	tasty	buy	juice	fruit
$\overrightarrow{dog}$	0.8	0.8	0.7	0.6	0.1	0.5	0.0	0.0
$\overrightarrow{cat}$	0.9	0.8	0.6	0.3	0.0	0.5	0.0	0.0
$\overrightarrow{apple}$	0.0	0.0	0.0	0.0	0.9	0.9	0.8	1.0
$\overrightarrow{orange}$	0.0	0.0	0.0	0.0	1.0	0.9	1.0	1.0

Table 1. Example noun vectors in **N**

two dimensions, and conceptually simple and easy to understand. Note also that the compositional framework places no limits of how large the sentence space can be. Grefenstette *et al.* (2011) and Grefenstette & Sadrzadeh (2011) describe a sentence space which has been implemented which has many more than 2 basis vectors.

Figure 11 gives two example vectors in the plausibility space, which has basis vectors corresponding to True and False (which can also be thought of as "highly plausible" and "not at all plausible"). The sentence *dog chases cat* is considered highly plausible, since it is close to the True basis vector, whereas *apple chases orange* is considered highly implausible, since it is close to the False basis vector.

We will use the example sentence *dog chases cat*, and show how a vector can be built for this sentence in the plausibility space, assuming vectors for the nouns, and the transitive verb (which will be an order 3 tensor), already exist. The vectors assumed for the nouns are given in Table 1, together with example vectors for the nouns *apple* and *orange*. Note that, again, these are fictitious counts in the table, assumed to have been obtained using one of the methods described in Section 3.

The compositional framework is agnostic towards the particular noun vectors, in that it does not matter how those vectors are built. However, for

	⟨fluffy,T,fluffy⟩	⟨fluffy,F,fluffy⟩	⟨fluffy,T,fast⟩	⟨fluffy,F,fast⟩	⟨fluffy,T,juice⟩	⟨fluffy,F,juice⟩	⟨tasty,T,juice⟩ . . .
$\overrightarrow{chases}$	0.8	0.2	0.75	0.25	0.2	0.8	0.1
$\overrightarrow{eats}$	0.7	0.3	0.6	0.4	0.9	0.1	0.1

Table 2. Example transitive verb vectors in $\mathbf{N} \otimes \mathbf{S} \otimes \mathbf{N}$

	⟨fluffy,T,fluffy⟩	⟨fluffy,F,fluffy⟩	⟨fluffy,T,fast⟩	⟨fluffy,F,fast⟩	⟨fluffy,T,juice⟩	⟨fluffy,F,juice⟩	⟨tasty,T,juice⟩ . . .
$\overrightarrow{chases}$	0.8	0.2	0.75	0.25	0.2	0.8	0.1
dog,cat	0.8,0.9	0.8,0.9	0.8,0.6	0.8,0.6	0.8,0.0	0.8,0.0	0.1,0.0

Table 3. Vector for *chases* (order 3 tensor) together with subject *dog* and object *cat*

explanatory purposes it will be useful to think of the basis vectors of the noun space as corresponding to properties of the noun, obtained using the output of a dependency parser, as described in Section 3.1. For example, the count for *dog* corresponding to the basis vector *fluffy* is assumed to be some weighted, normalised count of the number of times the adjective *fluffy* has modified the noun *dog* in some corpus; intuitively this basis vector has received a high count for *dog* because dogs are generally fluffy. Similarly, the basis vector *buy* corresponds to the object position of the verb *buy*, and *apple* has a high count for this basis vector because apples are the sorts of things that are bought.

Table 2 gives example vectors for the verbs *chases* and *eats*. Note that, since transitive verbs live in $\mathbf{N} \otimes \mathbf{S} \otimes \mathbf{N}$ — i.e. are order 3 tensors — the basis vectors across the top of the table are triples of the form (n_i, s_j, n_k), where n_i and n_k are basis vectors from $\mathbf{N}$ (properties of the noun) and s_j is a basis vector from $\mathbf{S}$ (True or False in the plausibility space).

The way to read the fictitious numbers is as follows: *chases* has a high (normalised) count for the ⟨fluffy,T,fluffy⟩ basis vector, because *it is highly plausible that* fluffy things chase fluffy things. Conversely, *chases* has a low count for the ⟨fluffy,F,fluffy⟩ basis vector, because *it is* not *highly implausible that* fluffy things chase fluffy things.[11] Similarly, *eats* has a low count for the ⟨tasty,T,juice⟩ basis vector, because *it is* not *highly plausible that* tasty things eat things which can be juice.

Table 3 gives the vector for *chases*, together with the corresponding counts for the subject and object in *dog chases cat*. For example, the (*dog*, *cat*) count pair for the basis vectors ⟨fluffy,T,fluffy⟩ and ⟨fluffy,F,fluffy⟩ is $(0.8, 0.9)$, meaning that dogs are fluffy to an extent 0.8, and cats are fluffy to an extent 0.9.

[11] The counts in the table for (n_i, T, n_k) and (n_i, F, n_k), for some n_i and n_k, always sum to 1, but this is not required and no probabilistic interpretation of the counts is being assumed.

The property counts for the subject and object are taken from the noun vectors in Table 1.

The reason for presenting the vectors in the form in Table 3 is that a procedure for combining the vector for *chases* with the vectors for *dog* and *cat* now suggests itself. How plausible is it that a dog chases a cat? Well, we know the extent to which fluffy things chase fluffy things, and the extent to which a dog is fluffy and a cat is fluffy; we know the extent to which things that are bought chase things that can be juice, and we know the extent to which dogs can be bought and cats can be juice; more generally, for each property pair, we know the extent to which the subjects and objects of *chases*, in general, embody those properties, and we know the extent to which the particular subject and object in the sentence embody those properties. So multiplying the corresponding numbers together brings information from both the verb and the particular subject and object.

The calculation for the True basis vector of the sentence space for *dog chases cat* is as follows:

$$\overrightarrow{dog\ chases\ cat}_{\text{True}} = 0.8\,.\,0.8\,.\,0.9 + 0.75\,.\,0.8\,.\,0.6 + 0.2\,.\,0.8\,.\,0.0 + 0.1\,.\,0.1\,.\,0.0 + \ldots$$

The calculation for the False basis vector is similar:

$$\overrightarrow{dog\ chases\ cat}_{\text{False}} = 0.2\,.\,0.8\,.\,0.9 + 0.25\,.\,0.8\,.\,0.6 + 0.8\,.\,0.8\,.\,0.0 + \ldots$$

We would expect the coefficient for the True basis vector to be much higher than that for the False basis vector, since *dog* and *cat* embody exactly those elements of the property pairs which score highly on the True basis vector for *chases*, and do not embody those elements of the property pairs which score highly on the False basis vector.

Through a particular example of the sentence space, we have now derived the expression in Coecke *et al.* (2010) for combining a transitive verb, $\overrightarrow{\Psi}$, with its subject, $\overrightarrow{\pi}$, and object, $\overrightarrow{o}$:

$$f(\overrightarrow{\pi} \otimes \overrightarrow{\Psi} \otimes \overrightarrow{o}) = \sum_{ijk} C_{ijk} \langle \overrightarrow{\pi} | \overrightarrow{\pi_i} \rangle \overrightarrow{s_j} \langle \overrightarrow{o} | \overrightarrow{o_k} \rangle \tag{14}$$

$$= \sum_{j} \left(\sum_{ik} C_{ijk} \langle \overrightarrow{\pi} | \overrightarrow{\pi_i} \rangle \langle \overrightarrow{o} | \overrightarrow{o_k} \rangle \right) \overrightarrow{s_j} \tag{15}$$

The expression $\langle \overrightarrow{\pi} | \overrightarrow{\pi_i} \rangle$ is the Dirac notation for the inner product between $\overrightarrow{\pi}$ and $\overrightarrow{\pi_i}$, and in this case the inner product is between a vector and one of its basis vectors, so it simply returns the coefficient of $\overrightarrow{\pi}$ for the $\overrightarrow{\pi_i}$ basis vector. From the linguistic perspective, these inner products are simply picking out particular properties of the subject, $\overrightarrow{\pi_i}$, and object, $\overrightarrow{o_k}$, and combining the corresponding property coefficients with the corresponding coefficient for the verb, C_{ijk}, for a particular basis vector in the sentence space, s_j.

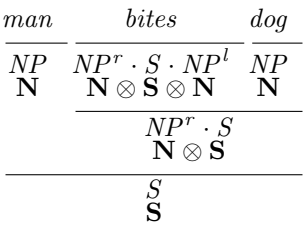


Figure 12. Example pregroup derivation with semantic types

Returning to the Baroni & Zamparelli (2010) case for adjective-noun combinations, the compositional procedure there is simply matrix multiplication of the noun vector by the adjective matrix. Note that the expression in (15) reduces to matrix multiplication when the relational type is of order 2 (as it is for an adjective). So a further useful intuition to take from Baroni & Zamparelli is that the reduction mechanism in the general case is a generalisation of matrix multiplication to higher-order tensors. In both the matrix multiplication case, and the reduction in (15), these compositional mechanisms take the form of linear maps.

Figure 12 shows a pregroup derivation for a simple transitive verb sentence, together with the corresponding semantic types. The point of the example is to demonstrate how the semantic types "become smaller" as the derivation progresses, in much the same way that the syntactic types do. The reduction, or cancellation, rule for the semantic component is given in (15), which can be thought of as the semantic vector-based analogue of the syntactic reduction rules in (11) and (12). Similar to a model-theoretic semantics, the semantic vector for the verb can be thought of as encoding all the ways in which the verb could interact with a subject and object, in order to produce a sentence, and the introduction of a particular subject and object reduces those possibilities to a single vector in the sentence space.

In summary, the meaning of a sentence $w_1 \cdots w_n$ with the grammatical (pregroup) structure $p_1 \cdots p_n \rightarrow^{\alpha} S$, where p_i is the grammatical type of w_i, and $\rightarrow^{\alpha}$ is the pregroup reduction to a sentence, can be represented as follows:

$$\overrightarrow{w_1 \cdots w_n} \quad = \quad F(\alpha)(\overrightarrow{w_1} \otimes \cdots \otimes \overrightarrow{w_n})$$

Here we have generalised the previous discussion of transitive verbs and extended the idea to syntactic reductions or derivations containing any types. The point is that the semantic reduction mechanism described in this section can be generalised to any syntactic reduction, α, and there is a function, F, which, given α, produces a linear map to take the word vectors $\overrightarrow{w_1} \otimes \cdots \otimes \overrightarrow{w_n}$ to a sentence vector $\overrightarrow{w_1 \cdots w_n}$. F can be thought of as Montague's "homomorphic passage" from syntax to semantics.

Concluding Remarks

One aspect of the compositional framework that was not stressed in the description is the close link between the mathematical structure used for the syntax — pregroup grammars — and that used for the distributional semantics — vector spaces with linear maps. Indeed this was one of the main reasons for using pregroup grammars. Coecke *et al.* (2010) discusses this connection in depth showing how both structures are examples of an abstract mathematical object called a compact closed category.

The question of how to evaluate the models described in this section, and compositional distributional models more generally, is an important one, but one that is still under active investigation. Pulman (2012) discusses the issue of evaluation, and describes a recent method of using compositional distributional models to disambiguate verbs in context (Mitchell & Lapata, 2008; Grefenstette & Sadrzadeh, 2011).

There are some obvious ways in which the existing work could be extended. First, we have only presented a procedure for building vectors for sentences with a simple transitive verb structure. The mathematical framework in Coecke *et al.* (2010) is general and in principle applies to any syntactic reduction from any sequence of syntactic types, but how to build relational vectors for all complex types is an open question. Second, it needs to be demonstrated that the compositional distributional representations presented in this section can be useful for language processing tasks and applications. Third, there is the question of whether the tensors for relational types can be learnt automatically from data using machine learning. Some related work in this vein which uses recursive neural networks is Socher *et al.* (2012).

5 Conclusion

Vector space models of meaning have proven successful in computational linguistics and cognitive science and are currently an extremely active area of research. One of the reasons for their popularity in computational linguistics is that they provide a level of robustness required for effective language technology which has notoriously not been provided by more traditional logic-based approaches.[12] Even accounting for laudable attempts such as Bos & Markert (2006), building on earlier work such as Alshawi (1992), it is probably fair to say that the classical logic-based enterprise in natural language processing has failed. However, whether fundamental concepts from semantics, such as inference, can be suitably incorporated into vector space models of meaning is an open question.

In addition to incorporating traditional concepts from semantics, another area which is likely to grow is the incorporation of other modalities, such as vision, into the vector representations (Bruni *et al.*, 2012). A semantic theory incorporating distributional representations, at the word, phrase, sentence — and perhaps even document — level; multi-modal features; and effective and robust methods of inference, is a grand vision aimed at solving one of the most difficult and fundamental problems in Artificial Intelligence and the cognitve sciences. We still have a long way to go.

[12] A recent attempt at a more robust semantics closer to more traditional logic-based approaches is Copestake (2009).

Acknowledgements

Almost all of the ideas in Section 4 arose from discussions with Merhnoosh Sadrzadeh, Ed Grefenstette, Bob Coecke and Stephen Pulman. Much of the author's understanding of distributional word models described in Section 3 arose from discussions with James Curran in Edinburgh (2002-2004) and from proof-reading James' PhD thesis.

References

Agirre, Eneko, Enrique Alfonseca, Keith B. Hall, Jana Kravalova, & Marius Pascaand Aitor Soroa (2009), A study on similarity and relatedness using distributional and wordnet-based approaches, in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Boulder, CO, (19–27).

Alshawi, Hiyan (ed.) (1992), *The Core Language Engine*, The MIT Press.

Baroni, M., B. Murphy, E. Barbu, & M. Poesio (2010), Strudel: A corpus-based semantic model based on properties and types, *Cognitive Science* 34(2):222–254.

Baroni, M. & R. Zamparelli (2010), Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space, in *Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*, Cambridge, MA, (1183–1193).

Blackburn, Patrick & Johan Bos (2005), *Representation and Inference for Natural Language. A First Course in Computational Semantics*, CSLI Publications.

Bos, Johan & Katja Markert (2006), When logical inference helps determining textual entailment (and when it doesn't), in *Proceedings of the 2nd Recognising Textual Entailment Challenge*, Venice, Italy, (98–103).

Brants, Thorsten (2000), TnT - a statistical part-of-speech tagger, in *Proceedings of the 6th Conference on Applied Natural Language Processing*, (224–231).

Brin, Sergey & Lawrence Page (1998), The anatomy of a large-scale hypertextual web search engine, *Computer Networks* 30:107–117.

Briscoe, Ted & John Carroll (2006), Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank, in *Proceedings of the Poster Session of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL-06)*, Sydney, Austrailia, (41–48).

Briscoe, Ted, John Carroll, & Rebecca Watson (2006), The second release of the RASP system, in *Proceedings of the Interactive Demo Session of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL-06)*, Sydney, Austrailia, (77–80).

Bruni, E., G. Boleda, M. Baroni, & N. Tran. 2012 (2012), Distributional semantics in technicolor, in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Jeju Island, Korea, (136–145).

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

Chan, Yee Seng, Hwee Tou Ng, & David Chiang (2007), Word sense disambiguation improves statistical machine translation, in *Proceedings of the 45th Meeting of the ACL*, Prague, Czech Republic, (33–40).

Clark, Stephen & James R. Curran (2007), Wide-coverage efficient statistical parsing with CCG and log-linear models, *Computational Linguistics* 33(4):493–552.

Clarke, Daoud (2008), *Context-theoretic Semantics for Natural Language: An Algebraic Framework*, Ph.D. thesis, University of Sussex.

Clarke, Daoud (2012), A context-theoretic framework for compositionality in distributional semantics, *Computational Linguistics* 38(1):41–71.

Coecke, B., M. Sadrzadeh, & S. Clark (2010), Mathematical foundations for a compositional distributional model of meaning, in J. van Bentham, M. Moortgat,

& W. Buszkowski (eds.), *Linguistic Analysis (Lambek Festschrift)*, volume 36, (345–384).

Copestake, Ann (2009), Slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go, in *Proceedings of the 12th Meeting of the EACL (EACL-09)*, Athens, Greece, (1–9).

Van de Cruys, Tim (2010), A non-negative tensor factorization model for selectional preference induction, *Journal of Natural Language Engineering* 16(4):417–437.

Curran, James R. (2004), *From Distributional to Semantic Similarity*, Ph.D. thesis, University of Edinburgh.

Curran, James R. & Stephen Clark (2003), Investigating GIS and smoothing for maximum entropy taggers, in *Proceedings of the 10th Meeting of the EACL*, Budapest, Hungary, (91–98).

Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, & Richard Harshman (1990), Indexing by latent semantic analysis, *Journal of the American Society for Information Science* 41(6):391–407.

Dowty, D.R., R.E. Wall, & S. Peters (1981), *Introduction to Montague Semantics*, Dordrecht.

Firth, J. R. (1957), A synopsis of linguistic theory 1930-1955, in *Studies in Linguistic Analysis*, Oxford: Philological Society, (1–32), reprinted in F.R. Palmer (ed.), Selected Papers of J.R. Firth 1952-1959, London: Longman (1968).

Fodor, Jerry & Zenon Pylyshyn (1988), Connectionism and cognitive architecture: A critical analysis, *Cognition* 28:3–71.

Grefenstette, Edward & Mehrnoosh Sadrzadeh (2011), Experimental support for a categorical compositional distributional model of meaning, in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Edinburgh, Scotland, UK, (1394–1404).

Grefenstette, Edward, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke, & Stephen Pulman (2011), Concrete sentence spaces for compositional distributional models of meaning, in *Proceedings of the 9th International Conference on Computational Semantics (IWCS-11)*, Oxford, UK, (125–134).

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

Griffiths, Thomas L., Mark Steyvers, & Joshua B. Tenenbaum (2007), Topics in semantic representation, *Psychological Review* 114(2):211–244.

Guevara, Emiliano Raul (2011), Computing semantic compositionality in distributional semantics, in *Proceedings of the 9th International Conference on Computational Semantics (IWCS-11)*, Oxford, UK.

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

Hearst, Marti (1992), Automatic acquisition of hyponyms from large text corpora, in *Proceedings of the Fourteenth International Conference on Computational Linguistics*, Nantes, France.

Hockenmaier, Julia & Mark Steedman (2007), CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank, *Computational Linguistics* 33(3):355–396.

Hodgson, James M. (1991), Informational constraints on pre-lexical priming, *Language and Cognitive Processes* 6:169–205.

Lambek, Joachim (1958), The mathematics of sentence structure, *American Mathematical Monthly* (65):154–170.

Lambek, Joachim (2008), *From Word to Sentence. A Computational Algebraic Approach to Grammar*, Polimetrica.

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

Lin, Dekang (1998), An information-theoretic definition of similarity, in *Proceedings of the Fifteenth International Conference on Machine Learning*, Madison, WI, (296–304).

Lin, Dekang & Patrick Pantel (2001), Discovery of inference rules for question answering, *Journal of Natural Language Engineering* 7(4):343–360.

Lowe, Will & Scott McDonald (2000), The direct route: Mediated priming in semantic space, in *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, Philadelphia, PA, (675–680).

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

Manning, Christopher & Hinrich Schutze (1999), *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge, Massachusetts.

Manning, Christopher D., Prabhakar Raghavan, & Hinrich Schutze (2008), *Introduction to Information Retrieval*, Cambridge University Press.

Marcus, Mitchell, Beatrice Santorini, & Mary Marcinkiewicz (1993), Building a large annotated corpus of English: The Penn Treebank, *Computational Linguistics* 19(2):313–330.

Medin, Douglas L., Robert L. Goldstone, & Dedre Gentner (1990), Similarity involving attributes and relations: Judgements of similarity and difference are not inverses, *Psychological Science* 1(1):64–69.

Minnen, Guido, John Carroll, & Darren Pearce (2001), Applied morphological processing of English, *Natural Language Engineering* 7(3):207–223.

Mitchell, Jeff & Mirella Lapata (2008), Vector-based models of semantic composition, in *Proceedings of ACL-08*, Columbus, OH, (236–244).

Moortgat, Michael (1997), Categorial type logics, in Johan van Benthem & Alice ter Meulen (eds.), *Handbook of Logic and Language*, Elsevier, Amsterdam and MIT Press, Cambridge MA, chapter 2, (93–177).

Nielsen, Michael A. & Isaac L. Chuang (2000), *Quantum Computation and Quantum Information*, Cambridge University Press.

Pado, Sebastian & Mirella Lapata (2007), Dependency-based construction of semantic space models, *Computational Linguistics* 33(2):161–199.

Preller, Anne & Mehrnoosh Sadrzadeh (2009), Bell states as negation in natural languages, *ENTCS, QPL* .

Pulman, Stephen (2012), Distributional semantic models, in Sadrzadeh Heunen & Grefenstette (eds.), *Compositional Methods in Physics and Linguistics*, Oxford University Press.

Robertson, Stephen & Hugo Zaragoza (2009), The probabilistic relevance framework: BM25 and beyond, *Foundations and Trends in Information Retrieval* 3(4):333–389.

Rose, T., M. Stevenson, & M. Whitehead (2002), The reuters corpus – from yesterday's news to tomorrow's language resources, in *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-02)*, Las Palmas, Canary Islands, (827–832).

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

Salton, G., A. Wang, & C. Yang (1975), A vector-space model for information retrieval, *Journal of the American Society for Information Science* 18:613–620.

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

Smolensky, Paul (1990), Tensor product variable binding and the representation of symbolic structures in connectionist systems, *Artificial Intelligence* 46:159–216.

Socher, Richard, Brody Huval, Christopher D. Manning, & Andrew Y. Ng (2012), Semantic compositionality through recursive matrix-vector spaces, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Jeju, Korea, (1201–1211).

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

Steedman, Mark (2000), *The Syntactic Process*, The MIT Press, Cambridge, MA.

Steedman, Mark & Jason Baldridge (2011), Combinatory categorial grammar, in Robert Borsley & Kersti Borjars (eds.), *Non-Transformational Syntax: Formal and Explicit Models of Grammar*, Wiley-Blackwell.

Toutanova, Kristina, Dan Klein, Christopher Manning, & Yoram Singer (2003), Feature-rich part-of-speech tagging with a cyclic dependency network, in *Proceedings of the HLT/NAACL conference*, Edmonton, Canada, (252–259).

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

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

Turney, Peter D. & Patrick Pantel (2010), From frequency to meaning: Vector space models of semantics, *Journal of Artificial Intelligence Research* 37:141–188.

Weir, David (1988), *Characterizing Mildly Context-Sensitive Grammar Formalisms*, Ph.D. thesis, University of Pennsylvania.

Werning, Markus, Wolfram Hinzen, & Edouard Machery (eds.) (2012), *The Oxford Handbook of Compositionality*, Oxford University Press.

Widdows, Dominic (2004), *Geometry and Meaning*, CSLI Publications, Stanford University.

Widdows, Dominic (2008), Semantic vector products: Some initial investigations, in *Proceedings of the Second AAAI Symposium on Quantum Interaction*, Oxford, UK.

Wittgenstein, Ludwig (1953), *Philosophical Investigations*, Blackwell.

Zanzotto, Fabio Massimo, Ioannis Korkontzelos, Francesca Fallucchi, & Suresh Manandhar (2010), Estimating linear models for compositional distributional semantics, in *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, Beijing, China, (1263–1271).

Zhai, ChengXiang & John Lafferty (2004), A study of smoothing methods for language models applied to information retrieval, *ACM Transactions on Information Systems* 2(2).