Context-theoretic Semantics for Natural Language: an Algebraic Framework

Daoud Clarke

September 2007

Contents

Part I

The Context-theoretic Framework

Chapter 1

Introduction

This thesis deals with the philosophical and theoretical foundations of computational linguistics. We are interested in the nature of meaning in natural language and the ways in which meaning can be represented computationally, in particular the relationship between vector-based representations of meaning and logical representations.

In recent years, the abundance of text corpora and computing power has allowed the development of techniques to analyse statistical properties of words. These techniques have proved useful in many areas of computational linguistics, arguably providing evidence that they capture something about the nature of words that should be included in representations of their meaning. However, it is very difficult to reconcile these techniques with existing theories of meaning in language, which revolve around logical and ontological representations. The new techniques, almost without exception, can be viewed as dealing with vector-based representations of meaning, placing meaning (at least at the word level) within the realm of mathematics and algebra; conversely the older theories of meaning dwell in the realm of logic and ontology. It seems there is no unifying theory of meaning to provide guidance to those making use of the new techniques.

The problem appears to be a fundamental one in computational linguistics since the whole foundation of meaning seems to be in question. The older, logical theories often subscribe to a model-theoretic philosophy of meaning (Kamp and Reyle, 1993; Blackburn and Bos, 2005). According to this approach, sentences should be translated to a logical form that can be interpreted as a description of the state of the world. The new vector-based techniques, on the other hand, are often closer in spirit to the philosophy of "meaning as context", that the meaning of an expression is determined by how it is used. This is an old idea with origins in the philosophy of Wittgenstein (1953), who said that "meaning just *is* use" and Firth (1957a), "You shall know a word by the company it keeps", and the distributional hypothesis of Harris (1968), that words will occur in similar contexts if and only if they have similar meanings. Whilst the two philosophies are not obviously incompatible — especially since the former applies mainly at the sentence level

and the latter mainly at the word level — it is not clear how they relate to each other.

While the model-theoretic philosophy of meaning provides us with theories which allow a complete description of natural language from the word level to the sentence level and beyond, the same cannot be said for the philosophy of meaning as context. It is this philosophy that has inspired vector based techniques, yet there is currently no theory explaining how these vectors can be used to represent phrases and sentences. This lack of a firm theoretical foundation has far-reaching implications for computational linguists or engineers implementing systems that represent expressions using vectors.

Such a theoretical foundation would be applicable to a wide range of tasks involving natural language. The task of recognising textual entailment was developed as part of a PASCAL Challenge (Dagan et al., 2005a; Bar-Haim et al., 2006) in an attempt to identify a generic task that is inherent in a number of areas within natural language processing, including information retrieval, question answering, machine translation and paraphrase acquisition. The task is to determine, given two sentences or natural language expressions (called the *text* and *hypothesis* sentences), whether the first entails or implies the second, for example in the case of the two sentences

- *Text:* Once called the "Queen of the Danube," Budapest has long been the focal point of the nation and a lively cultural centre.

- *Hypothesis:* Budapest was once popularly known as the "Queen of the Danube."

the text sentence does entail the hypothesis. Finding a solution to this task necessarily means solving the majority of problems within computational linguistics and natural language processing because the task is so general.

The PASCAL Challenge provided a method of evaluating textual entailment systems using a large number of text-hypothesis pairs. A large proportion, 22 of the 41 entered runs, made use of corpus or web-based statistics, yet there is no linguistic theory of meaning that explains how to determine entailment between sentences using such statistics. We might be able to find vector representations for words or multi-word expressions by statistical analysis, but we are left without any guidelines about how sentences should be represented. Entailment systems making use of such statistics thus have to resort to somewhat ad-hoc methods tuned and evaluated empirically by their performance at the task. While this is fine from an engineering perspective, it leaves a lot to be desired from a linguistic perspective, since we are left without a deeper understanding of the nature of language.

In this thesis we attempt to solve these problems by identifying a framework to provide guidelines as to how to deal with vector-based representations of meaning in a principled

way. We were looking for specific properties from the framework, namely, we wanted the framework to:

- provide some guidelines describing in what way the representation of a phrase or sentence should relate to the representations of the individual words as vectors;
- require information about the probability of a string of words to be incorporated into the representation;
- provide a way to measure the degree of entailment between strings based on the particular meaning representation;
- be general enough to encompass logical representations of meaning;
- be able to incorporate the representation of ambiguity and uncertainty, including statistical information such as the probability of a parse or the probability that a word takes a particular sense;

The framework itself does not provide a recipe for how to represent meaning in natural language, instead it provides restrictions on the set of possibilities. The advantage of the framework is in ensuring that techniques are used in a way that is well-founded in a theory of meaning. For example, given vector representations of words, there is not one single way of combining these to give vector representations of phrases and sentences, but in order to fit within the framework there are certain properties of the representation that need to hold. Any method of combining these vectors in which these properties hold can be considered within the framework and is thus justified according to the underlying theory; in addition the framework instructs us as to how to measure the degree of entailment between strings according to that particular method. In the second part of the thesis, we show how the framework can be applied to problems in natural language processing.

Implementations of the framework are called *context theories* since we think of them as theories about the contexts that strings of the language occur in. By analogy with the term "model-theoretic" we use the term "context-theoretic" for concepts relating to context theories; in particular we will often call our framework "the context-theoretic framework".

Our approach to identifying the framework can be divided into several components, as depicted in Figure 1.1:

- We examine the philosophy of meaning as context, looking at the ideas of Wittgenstein, Firth and Harris as well as later developments to these ideas — see Chapter 2. In this chapter, we also review statistical techniques that analyse occurrences

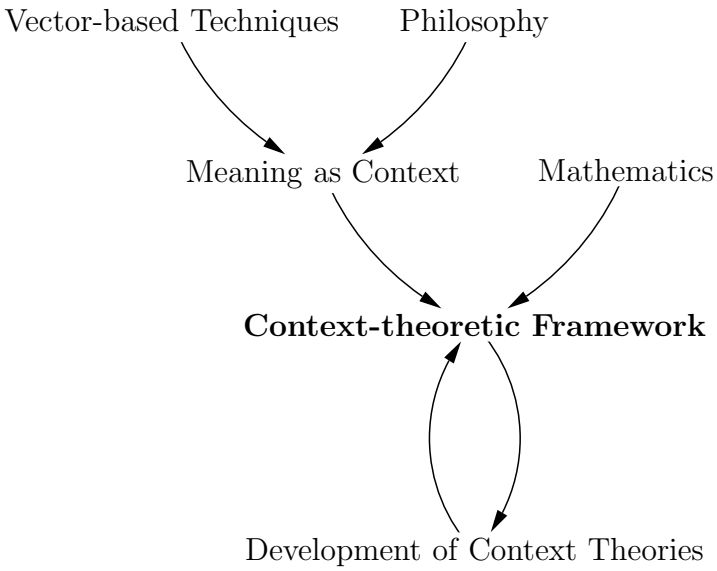


Figure 1.1: Method of Approach in developing the Context-theoretic Framework.

of words in corpora to determine something about their meaning; such techniques can usually be viewed as representing meaning in terms of vectors. Specifically we look at latent semantic analysis and its variations, and measures of distributional similarity.

- There are many areas of mathematics that could be of benefit to the problems we are addressing; in the Appendix we summarise those areas that are particularly relevant to our approach.

- Based on the philosophy of meaning as context, and inspired by the statistical techniques, we develop a mathematical theory of meaning as context by making use of the abstract mathematical idea of a *corpus model*, and we examine the mathematical properties of such models. This theory is vital in formulating the framework; in fact the framework can be viewed as a mathematical abstraction of the properties of the theory (see Section 3.1).

- In Section 3.2 we abstract the theory of meaning as context to define the context-theoretic framework, based on an analysis of the features that were important to include in the framework and the mathematics presented in the Appendix.

- The second half of the thesis is devoted to describing applications of the context-theoretic framework, in order to demonstrate its usefulness in describing natural language (these are summarised in Table 1.1). The applications were developed

simultaneously with the framework itself; this also helped us to identify which features were important to include in the framework. The areas we look at are as follows:

  - In Chapter 4 we look at the application of the framework to the task of recognising textual entailment, comparing our framework to the approach of others and showing how several existing approaches can be described in terms of context theories.
  - In Chapter 5 we show how the framework can be used to extend standard logical semantics for natural language to include statistical information about uncertainty of meaning.
  - In Chapter 6 we discuss the relationship between ontological representations of meaning and vector-based representations, and show how to construct vector-based representations of meaning from a taxonomy.
  - in Chapter 7 we show how syntactic structure can be represented within the framework, leading to new representations for syntax, and potentially new techniques for statistical parsing of natural language.

In summary, the major contributions of the thesis are as follows:

- the development of a mathematical theory of meaning as context that solidifies ideas implicit in existing philosophies and techniques
- the identification of a framework for natural language semantics that abstracts the salient features from the theory of meaning, providing guidelines for implementations that make use of vector-based representations of meaning;
- a demonstration of the application of the framework to important problems in natural language processing, most importantly the representation of statistical information about uncertainty and ambiguity in logical semantics.
- in the development of applications of the framework some theoretical discoveries were made:
  - in Chapter 6 we describe *vector lattice embeddings* of partial orderings — that is, ways to associate vectors with elements of a partially ordered set such as a taxonomy describing a hierarchy of concepts in such a way that the partial ordering is preserved;
  - in Chapter 7 we provide new ways of describing link grammar both in terms of operators on a Hilbert space and in terms of inverse semigroups.

Context Theory	Purpose	Section
Document projections	Relate Glickman and Dagan's (2005) approach to the task of recognising textual entailment to the context-theoretic framework.	4.2.1
Subsequence matching	Estimate the degree of entailment based on the number of shared subsequences.	4.2.2
Lexical overlap	Relate the degree of lexical overlap, commonly used as a baseline in the task of recognising textual entailment, to the framework.	4.2.2
Projections for Logic	Represent logical sentences within the framework, allowing statistical information about ambiguity and uncertainty to be incorporated.	5.2.2
Ideal Projection Completion	Represent concepts from an ontology in such a way that words can be represented as weighted sums over the vector representation of its component senses.	6.2.2
Lambek Calculus	Represent syntactic categories in terms of the Lambek Calculus.	7.1.5
Link Grammar	Describe syntax in terms of link grammar as operators on a Hilbert space.	7.2.2
Semigroups	Construct a context theory from any semigroup.	7.3.6

Table 1.1: The context theories described in the thesis, together with a summary of their purpose and the location of their full descriptions.

Chapter 2

Background

2.1 Philosophy

The development of a theory of meaning inevitably requires subscription to a philosophy of *what meaning is*. We are interested in describing representations resulting from techniques that make use of context in order to determine meaning, therefore it is natural that we look for a philsophy in which meaning is closely connected to context. The closest we have found is in the ideas of Firth (1957a), and before him, Wittgenstein (1953).

2.1.1 Wittgenstein

Wittgenstein was concerned with understanding language for the purpose of applying it to philosophy. He believed that many errors in philosophical reasoning arose out of an incorrect understanding of what meaning is. In *Philosophical Investigations* Wittgenstein especially combats the idea that the meaning of a word is an object:

> 1. "When they (my elders) named some object, and accordingly moved towards something, I saw this and I grasped that the thing was called by the sound they uttered when they meant to point it out. Their intention was shown by their bodily movements, as it were the natural language of all peoples; the expression of the face, the play of the eyes, the movement of other parts of the body, and the tone of the voice which expresses our state of mind in seeking, having, rejecting, or avoiding something. Thus, as I heard words repeatedly used in their proper places in various sentences, I gradually learnt to understand what objects they signified; and after I had trained my mouth to form these signs, I used them to express my own desires."[1]
>
> These words, it seems to me, give us a particular picture of the essence of human language. It is this: the individual words in language name objects —

[1] A quotation from Augustine (Confessions, I.8.)

> sentences are combinations of such names. In this picture of language we find the roots of the following idea: Every word has a meaning. The meaning is correlated with the word. It is the object for which the word stands.

He later continues, "That philosophical concept of meaning has its place in a primitive idea of the way language functions".

Wittgenstein's own idea of meaning is later expressed as follows:

> 43. For a large class of cases — though not for all — in which we employ the word "meaning" it can be defined thus: the meaning of a word is its use in the language.

In other words, if we know exactly how a word should be used, then in general, we know its meaning. Note that Wittgenstein requires that we know the "use" of a word rather than merely the contexts it is used in. This implies a much stronger knowledge since it seems to require knowing the reason behind using a word in terms of the impact it will produce; knowing the contexts a word occurs in merely means we can list the particular situations in which the use of the word is appropriate.

2.1.2 Firth

Honeybone (2005) describes Firth's perception of language:

> ...Firth saw language as a set of events which speakers uttered, a mode of action, a way of "doing things", and therefore linguists should focus on speech events themselves. This rejected the common view that speech acts are only interesting for linguists to gain access to the "true" object of study — their underlying grammatical systems.
>
> As utterances occur in real-life contexts, Firth argued that their meaning derived just as much from the particular situation in which they occurred as from the string of sounds uttered. This integrationist idea, which mixes language with the objects physically present during a conversation to ascertain the meaning involved, is known as Firth's "contextual theory of meaning"...

This is summed up in the famous quote, "You shall know a word by the company it keeps" (Firth, 1957a).

Firth comes closer to the idea of "meaning as context" as it used in modern techniques in computational linguistics in his article *Modes of Meaning* (Firth, 1957b) in discussing "collocation":

> The following sentences show that a part of the meaning of the word *ass* in modern colloquial English can be by collocation:
>
> 1. An ass like Bagson might easily do that.
> 2. He is an ass.
> 3. You silly ass!
> 4. Don't be an ass!
>
> One of the meanings of *ass* is its habitual collocation with an immediately preceding *you silly*, and with other phrases of address or of personal reference.

He then clarifies the relationship between what he calls "meaning by collocation" and "contextual meaning":

> It must be pointed out that meaning by collocation is not at all the same thing as contextual meaning, which is the functional relation of the sentence to the processes of a context of situation in the context of culture.

For Firth, part of the meaning of a word may be determined by "collocation", but to know its meaning is to know its "use" in the general sense of Wittgenstein.

2.1.3 Harris

Neither Wittgenstein nor Firth make strong statements connecting meaning to its observed textual context. The first to do this was Harris (1968), whose work is often cited as first presenting the *distributional hypothesis*: that words will occur in similar contexts if and only if they have similar meanings. Harris is the first to suggest that meanings of words can be determined by statistical analysis of their occurrences in large amounts of text.

He describes this idea as follows (Harris, 1985, section 2.3 (b)):

> The fact that, for example, not every adjective occurs with every noun can be used as a measure of meaning difference. For it is not merely that different members of the one class have different selections of members of the other class with which they are actually found. More than that: if we consider words or morphemes A and B to be more different in meaning than A and C, then we will often find that the distributions of A and B are more different than the distributions of A and C. In other words, difference of meaning correlates with difference of distribution.

> If we consider *oculist* and *eye-doctor* we find that, as our corpus of actually occurring utterances grows, these two occur in almost the same environments... In contrast, there are many sentence environments in which *oculist* occurs but *lawyer* does not: e.g. *I've had my eyes examined by the same oculist for twenty years*, or *Oculists often have their prescription blanks printed for them by opticians*. It is not a question of whether the above sentence with *lawyer* substituted is true or not; it might be true in some situation. It is rather a question of the relative frequency of such environments with *oculist* and with *lawyer*, or of whether we will obtain *lawyer* here if we ask an informant to substitute any word he wishes for *oculist* (not asking which words have the same meaning).

Harris also proposes the idea that similarity in meaning can be quantified in terms of the difference in their environments (contexts):

> If A and B have almost identical environments except chiefly for sentences which contain both, we say they are synonyms: *oculist* and *eye-doctor*. If A and B have some environments in common and some not (e.g. *oculist* and *lawyer*) we say that they have different meanings, the amount of meaning difference corresponding roughly to the amount of difference in their environments. (This latter amount would depend on the numerical relation of different to same environments, with more weighting being given to differences of selectional subclasses.) If A and B never have the same environment, we say that they are members of two different grammatical classes (this aside from homonymity and from any stated position where both these classes occur).

There is a subtle distinction between the two statements

1. Words that have similar meanings will occur in similar contexts.
2. Words that occur in similar contexts will have similar meanings.

Harris does not seem to make this distinction explicitly, however it is clear from the above passage that he intends both since he proposes that "difference of meaning correlates with difference of distribution" in addition to proposing that words with similar meanings occur in similar contexts. For this reason we have stated the distributional hypothesis as "words will occur in similar contexts if and only if they have similar meanings".

While Harris notes that distributional features extend beyond the sentence level, he does not attempt to extend the connection between meaning and context significantly beyond the word level. He also talks only about similarity in meaning, and does not discuss the asymmetric relationship of entailment, and how this relates to context.

2.1.4 Later Developments

Harris's distributional hypothesis has been the inspiration for much of the statistical work on determining meaning from corpora. Very recently, attempts have been made to refine the distributional hypothesis.

Weeds et al. (2004) take this one step further with the introduction of the idea of "distributional generality". A term w_1 is distributionally more general than another term w_2 if w_2 occurs in a subset of the contexts that w_1 occurs in. They relate this to their measures of precision and recall which they use to define a variety of measures of distributional similarity.

The idea is that distributional generality may be connected to *semantic generality*. An example of this is the *hypernymy relation* or "is a" relation between nouns: a word w_1 is a hypernym of w_2 if w_1 refers to a concept that generalises the concept referred to by w_2, for example the term *animal* is a hypernym of *dog* since a dog is an animal. They explain the connection to distributional generality as follows:

> Although one can obviously think of counter-examples, we would generally expect that the more specific term *dog* can only be used in contexts where *animal* can be used and that the more general term *animal* might be used in all of the contexts where *dog* is used and possibly others. Thus, we might expect that distributional generality is correlated with semantic generality...

This has been refined by Geffet and Dagan (2005) with the introduction of two "distributional inclusion hypotheses". They define these in terms of "lexical entailment" between senses of words, rather than the hypernymy relation which is more specific in meaning and is defined between words. They also only consider what they call "syntactic-based features" which would include, for example, dependency relations, and discount co-occurrences within a window as providing useful knowledge about entailment. Finally, they assume that it is possible to distinguish the "characteristic" features — that is, those features that have an impact on the meaning of a word. Let s_1 and s_2 be two senses of words. Their hypotheses, then are:

1. If s_1 lexically entails s_2 then all the characteristic (syntactic-based) features of s_1 are expected to appear with s_2.

2. If all the characteristic (syntactic-based) features of s_1 appear with s_2 then we expect that s_1 lexically entails s_2.

The two hypotheses effectively tie the meaning (in terms of lexical entailment) to specific features of the contexts that terms occur in, however, the authors do not go so far as to attempt to equate the two.

2.1.5 Discussion

We view the ideas we have presented here as a progression in our understanding of meaning; this is not to say that each author was aware of the previous author's work, but that the ideas themselves relate to the previous ones. Wittgenstein first attempted to free people from existing perceptions of meaning by proposing that knowledge of the meaning of a word meant nothing more than knowing how to use it. Firth then proposed that part of the meaning of a word may be by collocation in his example of the word "ass". Harris went further in his proposal that words occur in similar contexts if and only if they have similar meanings. Recent work arising from computational techniques refines this idea by focussing on distributional and semantic generality, suggesting that a term with a more general meaning will occur in a wider range of contexts.

None of the authors go so far as to equate meaning with context: for example, Harris talks only about how meanings of words relate to one another, and that similarity and difference of meaning can be determined by examining the contexts words occur in. Thus Harris does not contradict earlier philosophers, since it is possible to know how the meanings of words relate to one another without knowing their meaning as Wittgenstein intended it.

For practical purposes of applications in computing however, we argue that knowing how meanings relate to one another is enough. This is something that has become clearer through the development of the notion of textual entailment, which can be applied to so many areas in natural language processing yet only requires a relative understanding of meaning. For this reason, in this thesis we will equate meaning with context, that is, we assume that a relative knowledge of meaning is sufficient. This is not a statement of our philosophical position, rather it is a simplification that is convenient for the problem we are addressing. We hope however that through this simplification and subsequent mathematical analysis we will be able to give a new perspective on meaning that can add to and enrich, rather than detract from, existing ideas of meaning.

2.2 Vector Based Representations of Meaning

By "vector-based representations of meaning" we really mean two main areas of research: that of latent semantic analysis and its variants, and that of measures of distributional similarity between natural language expressions. In general, both these areas involve representing expressions in terms of vectors which are built according to the contexts that the expression of interest occurs in in some large corpus. Figure 2.1 gives a sample of occurrences of the term "fruit" in the British National Corpus; typically context vectors are built from many more occurrences of a term.

In latent semantic analysis, a transformation is applied to the vectors, resulting in a new vector representation of an expression which is supposed to describe "latent" features of meaning of the expression. By contrast, measures of distributional similarity leave the initial vector representation intact, but use mathematical analysis to measure the similarity between these vectors in various ways.

Both techniques are dependent on how the initial vectors are built:

- The vector representation of an expression may depend purely on what document the expression occurs in: the representation is simply the multiset or bag of document identifiers corresponding to occurrences of the expression. The order of occurrences of words in a document is thus deemed unimportant in this model. Each dimension of the vector representation corresponds to a document in the corpus, and the size of a component of the representation of a word will be its frequency of occurrence in the corresponding document.

- In a *windowing model* the representation of an expression is built from words that occur within a certain "window" of n words from the expression of interest; again order of occurrence is unimportant. Each dimension of the vector representation now corresponds to a different word that expressions may co-occur with.

- The text may be parsed with a dependency parser and some or all of the resulting dependency relations are then used to build vectors. In this case, each dimension would correspond to a different relationship: a noun occurring as object of a verb would be in a different dimension to the same noun occurring as the subject of the verb.

The first of these relates closely to information retrieval applications, and it was this application that led to the development of latent semantic analysis; the second representation is also commonly used in latent semantic analysis. Variations on the third representation are more commonly used in measures of distributional similarity.

2.2.1 Latent Semantic Analysis

The technique of latent semantic analysis and the similar probabilistic techniques that followed it, arose from the work of Deerwester et al. (1990), in the context of the task of information retrieval. We will give only a brief overview here, since the details are not directly relevant to our work.

It is common in information retrieval to represent a term by the vector of documents it occurs in. Table 2.1 gives a set of hypothetical occurrences of six terms in eight documents.

end some medicine for her, but she will need fruit and milk, and some other special things that
our own. Here we give you ideas for foliage, fruit and various festive trimmings that you can i
part II). However, other strategies can bear fruit and are described under three sections which
       supper tomatoes, potato chips, dried fruit and cake. And they drank water out of tea-cu
erent days, as the East Berliners queue for fruit and cheap stereos, a Turkish beggar sleeps i
dening; and Pests -- how to control them on fruit and vegetables. Both are produced by the Hen
me,"Silver Queen" is male so will never bear fruit   At the opposite end of the prickliness sca
 lifted away   Like an orange lifted from a fruit-bowl   And darkness, blacker   Than an oil-
ed in your wreath. Christmas ribbon and wax fruit can be added for colour. Essentials are scis
e you need to start developing your very own fruit collection   KEEPING OUT THE COLD   Need e
ly with Jeyes fluid   THE KITCHEN GARDEN   FRUIT   Cut out cankers on fruit trees, except tho
wn and watered   AUTUMN HUES   Foliage and fruit enrich the autumn garden, whether glowing th
- have forgotten the maxim: " tel arbre tel fruit ". If I were willing to unstitch the past
 of three children of Alfred Roger Ackerley, fruit importer of London, and his mistress, Janett
rful didactic spirit, much that was to bear fruit in his years as a mature artist. Although thi
e all made with natural vegetable, plant and fruit ingredients such as chamomile, kukai nut and
ack in the soup.   He re-visits the Copella fruit juice farm in Suffolk, the business he told
rategic relationship" with Lotus, the first fruit of which is a mail gateway between Office and
, choose your plants carefully to enjoy the fruit of your labour all year round.   PLACES TO V
 and I love chips. Otherwise I'll nibble on fruit or something to convince myself that I'm eat
 tone and felt the softness and warmth of a fruit ripening against a wall? If she had she migh
ol place to set. Calories per slice: 395   Fruit Scones with cinnamon Butter   (makes 12)
ought me water. Another monster gave me some fruit to eat. A few monsters lay against my body a
ney fungus.   Cut out diseased wood on most fruit trees   VEGETABLES   Continue winter diggin
age and chafing.   Remove old, unproductive fruit trees by cutting them down to shoulder heigh
ITCHEN GARDEN   FRUIT   Cut out cankers on fruit trees, except those on peaches, plums and ch
ps remain, then stir in the sugar and dried fruit. Using a round- ended knife, stir in the mil
 of a homeland, well others dream too,   De fruit was forbidden an now yu can't chew,   How ca
onnoisseurs. We take a bite from an unusual fruit. We come away neither nourished nor ravished,

Figure 2.1: Occurrences and some context of occurrences of the word *fruit* in the British National Corpus.

Given a user query term, the information retrieval software may return the documents that have the most occurrences of that term. From the vector perspective, the documents are identified with the *components* of the vector representation of a term; given a query term, the most suitable document corresponds to the *greatest component* of the term's vector representation.

It is often the case, however, that there are documents that are suitable matches for a given query which do not contain that query term often, or even at all. These will not be returned by the straightforward matching technique, and latent semantic analysis aims to get around this problem. It aims to deduce "latent" information about where terms may be expected to appear, by reducing the number of dimensions in which vectors are represented. This is performed in such a way that the most important components of meaning are retained, while those thought to represent noise are discarded. This *dimensionality reduction* has the effect of moving vectors that were unrelated closer together, as they are "squashed" into a space of lower dimensionality. For example, in table 2.1, *banana* and *orange* never occur together, however they both occur with *apple* and *fruit* which provides evidence that they are related. Latent semantic analysis aims to deduce this relation.

Figure 2.2 is intended to give an idea of how this works. The outer rectangles represent

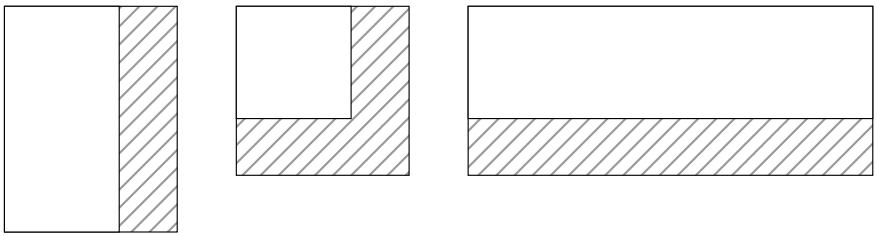


Figure 2.2: Matrix decomposition and dimensionality reduction in latent semantic analysis.

the matrices arrived at by singular value decomposition, their product gives the original matrix representing the table of term-document co-occurrences. These matrices are arranged so that the most important information is stored in the top and left areas, with less important information being stored towards the bottom and right. In latent semantic analysis, a rectangle of the most important information is chosen (the inner rectangles); this information is kept and the remaining areas of the matrices (those shaded in the diagram) are discarded — these are assumed to contain only noise information.

Table 2.2 shows the latent semantic analysis approximation to table 2.1. In this case we chose to keep only two dimensions for the inner rectangles. We can see that in the new table, *banana* and *orange* now have components in common — latent semantic analysis has forced them into a shared space. Because there were only two dimensions available, the term *computer*, which before only shared components with *apple* has been forced nearer to all the other terms, but remains closest to the term *apple* as we would expect.

Latent semantic analysis works as follows. The matrix M representing the original table can be decomposed into three matrices,

$$M = UDV,$$

where U and V are unitary matrices and D is a diagonal matrix containing the *singular values* of M. Figure 2.2 shows how the dimensionality reduction is performed. The decomposition can be rearranged so that the most important components — those with the greatest singular values — are in the top left of the matrix D, the dimensionality reduction is then performed by discarding the less important components, resulting in smaller matrices U', V' and D'. The matrix M is then *approximated* by the product of the new matrices, $M \simeq U'D'V'$.

For example, if we take table 2.1 as matrix M, then the decomposed and reduced matrices are those in figure 2.3. In this case we chose two keep only two dimensions corresponding to the greatest singular values (12.8 and 9.46); keeping more dimensions

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8
banana	2	–	–	–	5	–	5	–
apple	4	3	4	6	3	–	–	–
orange	–	2	1	–	–	7	–	3
fruit	–	1	3	–	4	3	5	3
tree	–	–	5	–	–	5	–	–
computer	–	–	–	6	–	–	–	–

Table 2.1: A table of hypothetical occurrences of words in a set of documents, d_1 to d_8.

$$\begin{pmatrix} .335 & -.175 \\ .504 & -.619 \\ .392 & .514 \\ .564 & .177 \\ .374 & .341 \\ .141 & -.415 \end{pmatrix} \begin{pmatrix} 12.8 & 0 \\ 0 & 9.46 \end{pmatrix} \begin{pmatrix} .209 & -.298 \\ .223 & -.0686 \\ .466 & .0295 \\ .302 & -.655 \\ .425 & -.213 \\ .492 & .617 \\ .351 & .00101 \\ .224 & .219 \end{pmatrix}^{\mathrm{T}}$$

Figure 2.3: The matrices U', D' and V' formed from singular value decomposition and dimensionality reduction. The product approximates the original matrix in table 2.1. Here A^{T} is used to mean the transpose of matrix A.

would mean that more features of the original matrix would be preserved.

Latent semantic analysis in its original form has some problems many of which have now been resolved to a large degree by new techniques. For example, the new, approximate matrix may contain negative values, as our example shows (table 2.2). This is undesirable, as the matrix is intended to represent expected co-occurrence frequencies, and these cannot be negative; this is a result of the technique's lack of grounding in a sound probabilistic analysis of the situation.

2.2.2 Probabilistic Latent Semantic Analysis

Probabilistic latent semantic analysis (Hofmann, 1999) is a technique which has the same aim as latent semantic analysis, but solves the problems of the technique in a probabilistic fashion, resolving the issue of negative values, and putting the technique on a firmer theoretical foundation. It treats the occurrence of a word w and a document d as random variables, and postulates the existence of a hidden variable z (see figure 2.4), and makes the assumption that d and w are independent conditioned on z. The parameters of the

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8
banana	1.40	1.08	1.95	2.40	2.19	1.09	1.51	.597
apple	3.11	1.85	2.84	5.80	4.00	-.44	2.26	.17
orange	-.40	.795	2.48	-1.68	1.10	5.49	1.77	2.20
fruit	1.02	1.50	3.41	1.08	2.71	4.60	2.53	1.99
tree	.041	.847	2.33	-.68	1.35	4.36	1.68	1.78
computer	1.56	.679	.731	3.13	1.62	-1.53	.635	-.455

Table 2.2: An approximation to the table obtained from a singular value decomposition followed by a dimensionality reduction to two dimensions.

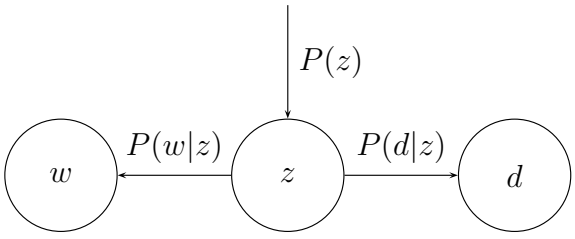


Figure 2.4: The probabilistic latent semantic analysis model of words w and documents d modelled as dependent on a latent variable z.

z_1	z_2
POWER	load
spectrum	memori
omega	vlsi
mpc	POWER
hsup	systolic
larg	input
redshift	complex
galaxi	arrai
standard	present
model	implement

Table 2.3: Most probable words given two topic variables relating to the term "power" (taken from Hofmann (1999)).

model are the probability distributions $P(z)$, $P(w|z)$ and $P(d|z)$. As Hofmann (1999) shows, these can be estimated by the Expectation Maximisation algorithm, using the following equations for the Expectation step

$$P(z|d,w) = \frac{P(z)P(d|z)P(w|z)}{\sum_{z' \in \mathcal{Z}} P(z')P(d|z')P(w|z')}$$

and the Maximisation step

$$\begin{aligned} P(w|z) &\propto \sum_{d \in \mathcal{D}} n(d,w)P(z|d,w) \\ P(d|z) &\propto \sum_{w \in \mathcal{W}} n(d,w)P(z|d,w) \\ P(z) &\propto \sum_{d \in \mathcal{D}} \sum_{w \in \mathcal{W}} n(d,w)P(z|d,w) \end{aligned}$$

where $\mathcal{D}$ denotes the set of documents, $\mathcal{W}$ the set of words and $\mathcal{Z}$ the set of values that the hidden variable z may take, and $n(d,w)$ represents the observed count of the number of occurrences of word w in document d.

Hofmann (1999) demonstrate the results of their analysis by selecting specific values for z and showing the ten most probable words according to $P(w|z)$. For example, they identified two values of z relating to the term "power" one which related to radiating objects in astronomy and one relating to electrical engineering (see Table 2.3).

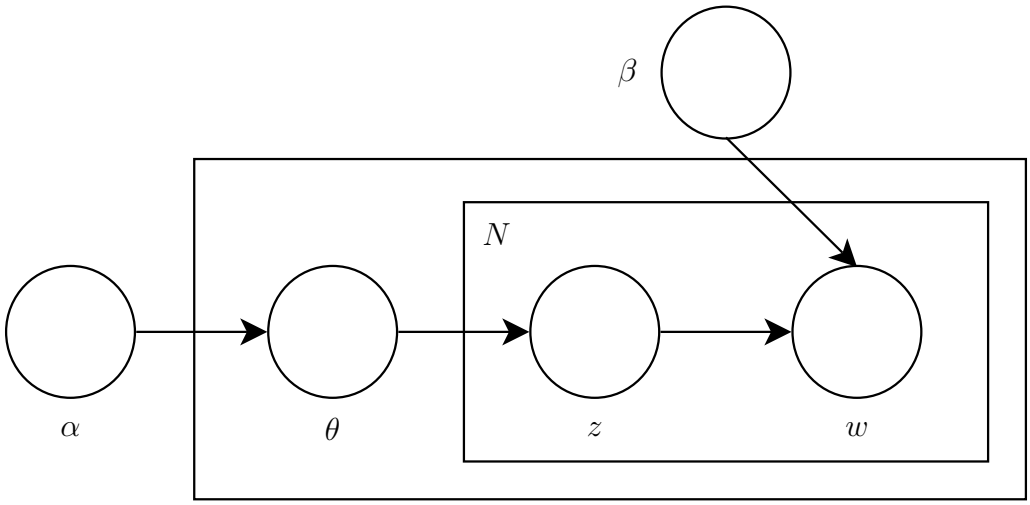


Figure 2.5: Graphical representation of the Dirichlet model, adapted from Blei et al. (2003). The inner box shows the choices that are repeated for each word in the document; the outer box the choice that is made for each document; the parameters outside the boxes are constant for the model.

1. Choose $\theta \sim \text{Dirichlet}(\alpha)$
2. For each of the N words:
   (a) Choose $z \sim \text{Multinomial}(\theta)$
   (b) Choose w according to $p(w|z)$

Figure 2.6: Generative process assumed in the Dirichlet model

2.2.3 Latent Dirichlet Allocation

Latent Dirichlet allocation (Blei et al., 2003) provides an even more in-depth Bayesian analysis of the situation. Blei et al. claim that the problem with probabilistic latent semantic analysis is that there is an assumed finite number of documents. This is not the true situation, they claim: the documents available should be viewed as a sample from an infinite set of documents. In order to achieve this, they model documents as samples from a *multinomial distribution* — a generalisation of the binomial distribution.

Figure 2.5 shows a graphical representation of the latent Dirichlet allocation generative model, and figure 2.6 shows how the model generates a document of length N. In this model, the probability of occurrence of a word w in a document is considered to be a multinomial variable conditioned on a k-dimensional "topic" variable z. The number of

topics k is generally chosen to be much fewer than the number of possible words, so that topics provide a "bottleneck" through which the latent similarity in meaning between words becomes exposed.

The topic variable is assumed to follow a multinomial distribution parameterised by a k-dimensional variable θ, satisfying

$$\sum_{i=1}^{k} \theta_i = 1,$$

and which is in turn assumed to follow a Dirichlet distribution. The Dirichlet distribution is itself parameterised by a k-dimensional vector α. The components of this vector can be viewed as determining the marginal probabilities of topics, since:

$$\begin{aligned} p(z_i) &= \int p(z_i|\theta)p(\theta)d\theta \\ &= \int \theta_i p(\theta)d\theta. \end{aligned}$$

This is just the expected value of θ_i, which is given by

$$p(z_i) = \frac{\alpha_i}{\sum_j \alpha_j}.$$

The model is thus entirely specified by α and the conditional probabilies $p(w|z)$ which we can assume are specified in a $k \times V$ matrix β where V is the number of words in the vocabulary. The parameters α and β can be estimated from a corpus of documents by a variational expectation maximisation algorithm, as described by Blei et al. (2003).

Latent Dirichlet allocation was applied by Blei et al. (2003) to the tasks of document modelling, document classification and collaborative filtering. They compare latent Dirichlet allocation to several techniques including probabilistic latent semantic analysis; latent Dirichlet allocation outperforms these on all of the applications. Recently, latent Dirichlet allocation has been applied to the task of word sense disambiguation (Cai et al., 2007; Boyd-Graber et al., 2007) with significant success.

2.2.4 Measures of Distributional Similarity

The use of distributional similarity measures (or often, more accurately, distance measures) has been an area of intense interest in computational linguistics in recent years (Lin, 1998; Lee, 1999; Curran and Moens, 2002; Kilgarriff, 2003; Weeds et al., 2004). A wide variety of measures have been suggested; we describe here some of the most commonly

used. The variety of measures derives from the variety of ways of viewing the occurrences of words in their contexts; of these, some of the most important are as follows (see table 2.4):

- We can associate a vector u with a word w in the manner previously described; the components of the vector are the frequencies of occurrence of w in each component c. Viewing occurrences in contexts from this perspective leads to measures based on the geometric properties of the vectors.

- We can renormalise the vector u to give a probability distribution p over contexts. This leads to information theoretic measures of dissimilarity based on standard measures of the difference in probability distributions.

- A consideration of which of the contexts are most important leads to measures which emphasise certain contexts over others, for example, mutual information may be used as an indication of which features are important.

Geometric measures

The most obvious are those with a clear geometric interpretation, namely measuring angles and distances between vectors (see table 2.4). The cosine of the angle between vectors is often used as a measure of similarity since it takes values between 0 and 1 and is equal to 1 only when the vectors are exactly the same. The Euclidean distance is the measure familiar to us in physical space and the L^1 norm or "city block" distance corresponds to the distance measured using only vertical and horizontal lines (in two dimensions).

Information theoretic measures

The more complex measures are more probabilistic in nature; vectors are normalised so that they can be considered as an estimate of a probability distribution over contexts. The basis of many of these measures is the Kullback-Leibler (KL) divergence $D(p\|q) = \sum_c p \log \frac{p}{q}$ of two distributions p and q. This measures the inefficiency of describing the true distribution p while assuming the distribution is q, and is thus an (asymmetric) measure of the difference between the two distributions. Using the KL divergence directly is not generally practical, however as it will be infinite if there is a context c for which $q(c) = 0$ and $p(c) \neq 0$. The Jenson-Shannon and α-skew measures get around this problem.

Measure	*Formula*		
Cosine	$\cos\theta$	$=$	$\frac{u \cdot v}{\|u\|\|v\|}$
Euclidean distance	$\|u - v\|$	$=$	$\sqrt{\sum_i (u_i - v_i)^2}$
City block distance	$\|u - v\|_1$	$=$	$\sum_i \|u_i - v_i\|$
Kullback-Leibler	$D(p\|q)$	$=$	$\sum_c p \log \frac{p}{q}$
Jenson-Shannon	$\text{dist}_{JS}(q, p)$	$=$	$\frac{1}{2}(D(p\|\frac{p+q}{2}) + D(q\|\frac{p+q}{2}))$
α-skew	$\text{dist}_{\alpha}(q, p)$	$=$	$D(p\|(\alpha q + (1-\alpha)p))$
Jaccard's	$\text{sim}_{ja}(w_2, w_1)$	$=$	$\frac{\|F(w_1) \cap F(w_2)\|}{\|F(w_1) \cup F(w_2)\|}$
Jaccard's (MI)	$\text{sim}_{ja+mi}(w_2, w_1)$	$=$	$\frac{\|S(w_1) \cap S(w_2)\|}{\|S(w_1) \cup S(w_2)\|}$
Lin's	$\text{sim}_{lin}(w_2, w_1)$	$=$	$\frac{\sum_{S(w_1) \cap S(w_2)} I(c,w_1) + I(c,w_2)}{\sum_{S(w_1)} I(c,w_1) + \sum_{S(w_2)} I(c,w_2)}$

Table 2.4: Eight measures of similarity and distance: geometric measures between vectors u and v, where u_i indicates the components of vector u, $u \cdot v$ indicates the dot product and $\|u\|$ denotes the Euclidean norm of u; measures based on the Kullback-Leibler divergence, where p and q are estimates of probability distributions describing the occurrences of words in contexts c; and measures based on the features of a word, either defined with respect to probability of occurrence, $F(w) = \{c : P(c|w) > 0\}$ or with respect to mutual information (this is also called the support of w), $S(w) = \{c : I(c, w) > 0\}$, where the mutual information I is given by $I(c, w) = \log(P(c|w)/P(c))$.

Feature-based measures

The "features" of a word are those contexts which are considered to provide interesting information about the word. The features can simply be the contexts that occur with non-zero probability with a word, as used in Jaccard's coefficient, which measures the proportion of contexts occurring with either word that are shared by both words. An alternative is to include only those contexts with positive mutual information, I, where $I(c, w) = \log(P(c|w)/P(c))$, this can be applied directly to the formula for Jaccard's coefficient, and also leads to Lin's measure, which is based on an information theoretic analysis of similarity (Lin, 1998).

2.3 Discussion

The most important aspects of the work we have discussed for our purposes are those which they have in common — they are all techniques which attempt to describe something about the meaning of a term based on the contexts the term appears in. The techniques are all flexible as to the exact interpretation of what context is — for example, a window of text may be used or a bag of grammatical relations. The input to the techniques is a bag or multiset of pairs of terms and contexts. We can also view this input as representing a term as a function from the set of contexts to the natural numbers, or more generally, as positive, real-valued functions on the set of contexts. Equivalently, we can think of such functions as positive elements of a real vector space whose dimensionality is given by the number of contexts. It is this latter perspective that is the starting point for our theory of meaning as context that is developed in the next chapter.

From here the techniques differ: latent semantic analysis and its variants attempt to transform these vectors to extract "latent" information about their meaning, while measures of distributional similarity leave the vectors as they are but make use of various methods to measure the similarity or difference between the vectors. Whilst our theory builds on what is common between the techniques, there are elements of each that will be of importance to us later. The concept of *corpus model* that we describe in the next chapter is a generalisation of the generative model of a corpus that is used in latent Dirichlet allocation. Distributional similarity has made use of various norms; this is an important topic for us in the development of *context theoretic probability* in the next chapter. We also discuss the relationship between certain measures of distributional similarity and context theories later in the thesis.

Chapter 3

Meaning as Context

We discussed in the previous chapter how vectors intended to represent the meaning of terms can be formed by looking at the contexts that terms appear in. However, these techniques do not provide any guidance as to how to represent the meaning of sentences or phrases in terms of these vector representations. Our approach to solving this problem is to build an abstract model of language based on the notion of meaning as context. In this chapter we first describe this model, in which both words and sequences of words are represented by vectors; we are then able to examine the mathematical properties of this model to providing guidelines as to how to combine vector representations of words to form representations of phrases and sentences. These properties form the basis of the context-theoretic framework, described in the second part of this chapter.

In particular there are three key properties that will be incorporated into the framework:

- The vectors associated with strings can be endowed with a lattice structure, making the object of study a *vector lattice*. This can be seen by looking in a very general manner at the way in which vectors in computational linguistics are derived. We shall interpret the associated partial ordering relation of the lattice as *entailment*; thus the lattice structure can be thought of as carrying the "meaning".

- We can define multiplication on the vector space in such a manner that the vector associated with the concatenation of two strings is the product of the vectors associated with each individual string. Remarkably, the multiplication makes the vector space an *algebra over a field* — a structure which has been the object of much study in mathematics.

- We shall show that according to this model, the size of the context vector of a term should correspond to its frequency of occurrence, we call this measure the *context theoretic probability* of a vector, denoted ϕ. This value makes two probability spaces from context vectors in two separate ways: the lattice structure of the vector space

can be viewed as a (traditional, measure-theoretic) probability space using using ϕ, while the algebra becomes a *non-commutative probability space* with ϕ as a linear functional.

These properties put strong requirements on the nature of an algebra to represent natural language; and it is these properties that will be required of any implementation of our framework. We will also show how a *degree of entailment* can be defined in terms of context vectors according to the ideas of distributional generality described previously. Later, when we discuss implementations of the framework, the same definition of the degree of entailment can be employed to the implementations because they have the same properties as the structure we derive in this chapter. This approach ensures that we can measure entailment for any implementation of the framework in a manner consistent with the context-theoretic philosophy.

3.1 A Model of Meaning as Context

We wish to build an abstract mathematical model based on techniques which build vector representations of words in terms of their contexts. These techniques are designed to deal with a limited amount of data, however in our abstract model, we are free to imagine that we have at our disposal an unlimited amount of data. This allows us to choose a very simple definition of what we mean by "context" — the context of a string will be the strings surrounding that string in a document. We are able to do this because we can always find enough data in our abstract model (we assume there is no problem of data sparseness). Simplifying the definition of context allows us to easily examine the mathematical properties of our model.

We view a real world text corpus (a finite collection of documents) as a sample of some hypothetical infinite collection of documents. Specifically, we assume a *probabilistic generative model* of corpora (Blei et al., 2003); one way to define such models is as follows:

Definition 3.1 (Corpus Model). A corpus model C on a set A of symbols is a probability distribution over A^*.

We can view a (real world) corpus as having been produced by a machine which repeatedly outputs strings according to the probability distribution C. Note that the machine is oblivious to what strings it has output previously; we can think of the individual strings output by the machine as *documents*: the order of the strings is unimportant with respect to the machine, and typically the order of documents in a corpus is unimportant (whereas the order of sentences, for example, often is important). Of course if may be useful in practice to think of the strings as sentences, paragraphs or any other unit of text.

Abstracting in this way allows us to discover the nature of meaning as context according to our assumptions in the hypothetical situation of having an infinite amount of data available to us by analysing the mathematical properties of the resulting mathematical structure. It also allows us to make use of techniques which build corpus models from finite corpora, such as Latent Dirichlet Allocation and associate meanings with strings according to the corpus model generated from the finite corpus.

3.1.1 Meaning as Context

How should we think about the meaning of an expression? For many applications in computational linguistics it suffices to know the relationships between the meanings of expressions: for example we should know if one entails another, or if two expressions are contradictory. For the purposes of what follows, we shall assume a purely *relative* interpretation of the word "meaning"; that is knowing the meaning of an expression means knowing how the expression relates to other expressions.

Techniques such as those discussed in the previous chapter typically build vector representations of meaning based on the context in which words or phrases appear; such representations only describe meaning in the relative sense described above.

Because of the problem of data sparseness, these techniques typically only make use of a part of the context of a string, for example using a limited window and ignoring the order of words in this window. Because we are assuming we have at our disposal a corpus model in which data sparseness is not a problem, we instead make full use of the context: the context of an expression in a document is everything surrounding the expression in the document.

Mathematically, the context vector of a string x will be a function over pairs of strings (u, v) with $u, v \in A^*$ such that uxv is a document. More formally:

Definition 3.2. The context vector of a string $x \in A^*$ in a corpus model C is a real-valued function $\hat{x} \in L^\infty(A^* \times A^*)$ on the set of contexts $A^* \times A^*$, defined by

$$\hat{x}(u, v) = C(uxv).$$

We stated here that the context vector of a string lives in the vector space $L^\infty(A^* \times A^*)$, that is, the set of bounded functions from $A^* \times A^*$ to the real numbers; we know that the functions are bounded because they are formed from the probability distribution C.

Thus, from this definition, we are able to associate with each string in A^* a vector representing the contexts it occurs in in the corpus model C, accordingly, we have all the properties of vector spaces at our disposal to study strings with respect to C: for example,

we can add, subtract and scale their associated context vectors.

3.1.2 Entailment

Consider the methods of forming vectors for terms described in the last chapter. In each method, vectors are formed from components which correspond to different contexts that the term may occur in: the components may correspond to words the term can occur with, or its possible dependency relations with other terms. What is important to note is that components of vectors in computational linguistics applications are *attached to concepts*; the exact method of determining the vectors is not of importance to us here.

In fact, this situation is somewhat special in comparison to other applications of vectors. For example, we live in a universe with three (observable) spatial dimensions. If we want, we can find a *basis* (see A.2.2) for this space, consisting of three vectors, x, y and z, say, allowing us to locate any point in space by a linear combination of these vectors; equivalently we can decompose any vector into components with respect to this basis. However, in general, we don't have a *preferred* choice of basis. There may be a basis which is convenient for us to use (for example we may choose x and y to be north and east and z to be up, with a length of 1 meter, for some particular location on earth), but there is no fundamental reason for us to prefer that basis.

In contrast, in computational linguistics, we are automatically provided with a basis purely by the way in which vectors are formed from components. For example, if we build the vector representation of a term by looking at the words occurring in a certain window of text, the dimensionality of the resulting vector space will be the same as the number of different words, and by default we will make use of a basis which has a basis vector corresponding to each different word. This fact is so obvious that its importance has been overlooked, but in reality it has profound implications for the properties we should expect from vector spaces in computational linguistics.

Careful consideration of this fact, can, we believe, lead us to answer one of the most difficult questions in the foundations of computational linguistics, regarding the relationship between vector representations of meaning and the older ontological and logical representations of meaning, based in lattice theory. This issue is touched on by Widdows (2004), where the implication is that the solution lies in generalising vector and lattice structures by weakening the mathematical requirements. In contrast, we will argue that all the necessary structure is already present and implicit in existing representations, which can be simultaneously be considered as vector spaces and lattices — they are *vector lattices* (see Section A.4).

Any vector space together with a basis can be considered as a vector lattice: the

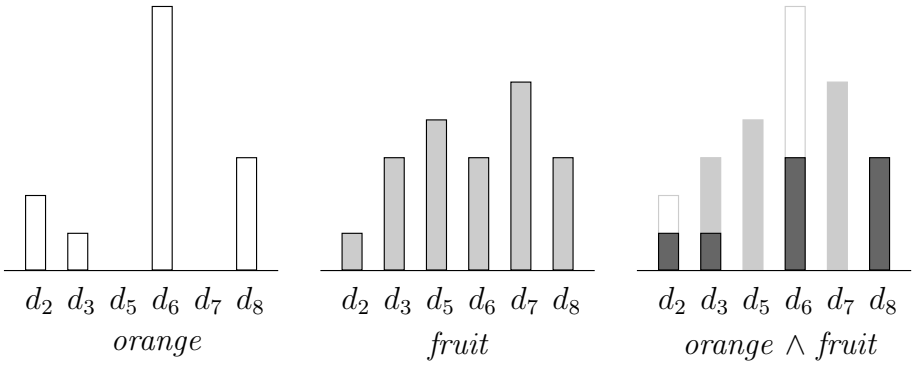


Figure 3.1: Vector representations of the terms *orange* and *fruit* based on hypothetical occurrences in six documents (see the previous chapter) and their vector lattice meet.

meet and join operations can be defined as the component-wise minimum and maximum respectively. Figure 3.1 shows two vectors representing the contexts of the terms *orange* and *fruit* based on their hypothetical occurrences in six documents, described in the previous chapter, and shows how their meet (component-wise minimum) is derived. Note that it is only because we are able to describe these vectors in terms of their components that we can define the lattice operations: the lattice operations are defined with respect to that particular basis, and if we had chosen a different basis the lattice operations would be different.

The vectors we discussed in the previous chapter were finite-dimensional; the context vector $\hat{x}$ of a string just defined is potentially infinite-dimensional. The same argument applies however: we can decompose the vector into components relating to individual contexts; for example, the basis vector corresponding to the context (u, v), for $u, v \in A^*$ is the function which takes the value 1 on (u, v) and 0 everywhere else on $A^* \times A^*$. Because we can decompose vectors in this way, we can again define lattice operations as component-wise minimum and maximum.

As with any lattice, there is an associated partial ordering; in this case, we write $\hat{x} \leq \hat{y}$ if each component of $\hat{x}$ is less than or equal to the corresponding component of $\hat{y}$. In terms of contexts, this means that the string x occurs in each context that y occurs in at least as frequently. Relating this back to the concept of distributional generality discussed in the previous chapter, we may state the following hypothesis: *a string fully entails another string if and only if the first occurs with equal or lower probability in all the contexts that the second occurs in*; or x entails y if and only if $\hat{x} \leq \hat{y}$.

In fact, we expect this situation to occur rarely; it is more likely that a string will share a proportion of its contexts with other strings. In order to be able to describe such "partial entailment" we need to have a way of measuring the size of such vectors, and this

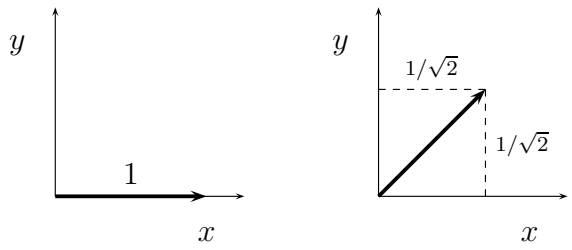


Figure 3.2: The length of a vector under the l^1 norm is not invariant under rotation.

is the topic of the next section.

3.1.3 Context-Theoretic Probability

Probability theory is central to modern techniques in computational linguistics, and it is thus important that our framework can inform us about the probabilistic aspect of language. In fact, we will show that in our model, the probability of a string is intimately connected to the "size" of its vector representation, as long as we choose a particular measure of size, the l^1 norm. We propose that this norm is the most appropriate way of measuring the size of vectors in computational linguistics. The l^1 norm of a vector is simply the sum of the absolute value of its components: if u is a vector with components u_i for $1 \leq i \leq n$, then the l^1 norm of u is given by

$$\|u\|_1 = |u_1| + |u_2| + \ldots + |u_n|.$$

There are many norms we could choose — why should the l^1 norm be special? The answer is that it has properties that make it more suitable for computational linguistics, while other norms have properties making them the most suitable in other applications of vector spaces.

For example, physical law in our three spatial dimensions has the special property that it is invariant with respect to rotation; this means that the l^2 norm occurs frequently in physical laws. The l^2 norm of a vector corresponds to the familiar Euclidean notion of its length: if u is a vector with components u_i for $1 \leq i \leq n$, then the l^2 norm of u is given by

$$\|u\|_2 = (u_1^2 + u_2^2 + \ldots + u_n^2)^{1/2}.$$

It has the special property that lengths remain the same under rotation, which is something we expect to observe in our universe. To see that the l^1 norm, for example, doesn't preserve lengths under rotation, consider a vector in the x-y plane which has a zero y

component and an x component of 1. This vector has length 1 under both the l^1 and l^2 norms. Rotating this by 45°, however we find the length under the l^1 norm is $2/\sqrt{2}$, whereas under the l^2 norm, the length remains 1.

There is however no reason for us to expect the same properties for vectors in computational linguistics. We can consider other, more exotic norms, such as the generalisations of the l^1 and l^2 norms, the l^p norms:

$$\|u\|_p = (|u_1|^p + |u_2|^p + \ldots + |u_n|^p)^{1/p},$$

where $1 \leq p < \infty$, and the l^∞ norm, where $\|u\|_\infty$ is the supremum over all components of u.

The l^1 norm, though, has a special property with regards to vectors in computational linguistics. In the previous chapter, we saw how, in practice, the vector representation of a term is built according to the frequencies of occurrence of that term in different contexts. Summing these frequencies is equivalent to summing the components of the vector representing the term; in the simplest methods of building vector representations we would expect this sum to be proportional to the frequency of occurrence of the term itself, or equivalently, proportional to its probability of occurrence.

In fact, there is a deeper connection to probability theory. Under the l^1 norm, the vector space becomes an *Abstract Lebesgue* or *AL* space (see section A.4.1). As the name suggests, the space can be considered as an abstraction of Lebesque spaces which form the foundation of measure theory, and hence the theory of probability. The key property is additivity of disjoint elements: in an AL space, if x and y are positive elements with $x \wedge y = 0$ then $\|x + y\| = \|x\| + \|y\|$. This is precisely the property we expect from probability: if we have two areas A and B in a Venn diagram which don't overlap, then we know that $P(A \cup B) = P(A) + P(B)$. In a vector lattice, we have

$$x \vee y = x + y - x \wedge y,$$

so if $x \wedge y = 0$ the above condition is the same as requiring $\|x \vee y\| = \|x\| + \|y\|$, an exact match for the Venn diagram requirement. Thus using the l^1 norm allows us to think of the vector space as simultaneously being a probability space. Although the structure is not what we normally think of as being a probability space (i.e. a set of elements which we can interpret as events) the mathematical properties are the same, and this is what is so attractive about using the l^1 norm in computational linguistics applications: we can treat the lattice with the l^1 norm *as if it's a probability space.*

There is a problem, however, when it comes to applying the l^1 norm to context vectors

as we have just defined them: they are not guaranteed to be finite. For example, consider the corpus model C on $A = \{a\}$ defined by

$$C(a^{2^n}) = 1/2^{n+1}$$

for integer $n \geq 0$, and zero otherwise, where by a^n we mean n repetitions of a, so for example, $C(a) = \frac{1}{2}$, $C(aa) = \frac{1}{4}$, $C(aaa) = 0$ and $C(aaaa) = \frac{1}{8}$. Then $\|\hat{a}\|_1$ is infinite, since each non-zero document contributes $1/2$ to the value of the norm, and there are an infinite number of non-zero documents.

To get around this, we resize the norm in such a way that the context of the empty string ϵ has a size of 1. This property will be important later, when we relate the construction to non-commutative probability. In order to get around the problem of infinities in the definition, we are forced to make a definition based on limits. Context vectors live in the infinite dimensional vector space $L^\infty(A^* \times A^*)$. To guarantee that the l^1 norm is finite, we define a projection P_n for integer n which projects context vectors to a finite dimensional vector subspace in which we only consider a finite subset of the contexts the string occurs in. If $u \in L^\infty(A^* \times A^*)$, that is u is a bounded function on $A^* \times A^*$, then we define

$$(P_n u)(x, y) = \begin{cases} 0 & \text{if } |x| + |y| > n, \\ u(x, y) & \text{otherwise,} \end{cases}$$

where $|x|$ denotes the length of string $x \in A^*$. Given this definition $\|P_n u\|_1$ is always finite; it is the sum of the components of u considering only those contexts $(x, y) \in A^* \times A^*$ for which the sum of the length of the strings x and y is less than or equal to n.

We are now in a position to define the context theoretic probability in terms of the limit of $n \to \infty$ as we consider increasingly bigger vector spaces:

Definition 3.3 (Context-theoretic Probability). The (context-theoretic) probability ϕ is a function on $L^\infty(A^* \times A^*)$ defined by

$$\phi(u) = \lim_{n \to \infty} \frac{\|P_n u\|_1}{\|P_n \hat{\epsilon}\|_1}$$

where u is a positive element of $L^\infty(A^* \times A^*)$, that is a positive bounded function on $A^* \times A^*$, as long as the limit exists and is finite, otherwise it is undefined. We extend ϕ to all elements of $L^\infty(A^* \times A^*)$ by defining $\phi(v) = \phi(v^+) - \phi(v^-)$, where v^+ and v^- are the positive and negative parts of v respectively (see section A.4).

The function ϕ is not guaranteed to be finite for all elements $u \in L^\infty(A^* \times A^*)$; however we are really interested in the value of ϕ on context vectors. The following

proposition shows that placing a small requirement on corpus models guarantees that the value of ϕ on context vectors, and vectors generated from context vectors under the vector and lattice operations is finite; it also clarifies the relationship between the context-theoretic probability of the context vector of a string and what we normally think of as the "probability of a string":

Proposition 3.4. *Let C be a corpus model such that for every sequence $x_n \in A^*$ with $|x_{n+1}| > |x_n|$, $\lim_{n\to\infty} C(x_n) = 0$. Then:*

1. *for $x \in A^*$, $\phi(\hat{x})$ is defined and satisfies $\phi(\hat{x}) \leq 1$ with $\phi(\hat{x}) = 1$ if and only if $x = \epsilon$.*
2. *$\sum_{a\in A} \phi(\hat{a}) < 1$.*
3. *If v is a vector in $L^\infty(A^* \times A^*)$ constructed from context vectors using a finite number of multiplication by a scalar, addition, meet and join operations, then $\phi(v)$ is defined.*

Proof.

1. Consider a document $d \in A^*$ with $|d| \leq n$. It contributes $(|d|+1)C(d)$ to $\|P_n\hat{\epsilon}\|_1$. The string $x \in A^* - \{\epsilon\}$ can occur at most $|d|-|x|+1$ times in d, so d can contribute at most $(|d|-|x|+1)C(d)$ to $\|P_n\hat{x}\|_1$. Thus

$$\begin{aligned}\frac{\|P_n\hat{x}\|_1}{\|P_n\hat{\epsilon}\|_1} &\leq \frac{\sum_{|d|\leq n+|x|}(|d|-|x|+1)C(d)}{\sum_{|d|\leq n}(|d|+1)C(d)} \\ &= \frac{\sum_{|d|\leq n}(|d|-|x|+1)C(d)}{\sum_{|d|\leq n}(|d|+1)C(d)} + \frac{\sum_{n<|d|\leq n+|x|}(|d|-|x|+1)C(d)}{\sum_{|d|\leq n}(|d|+1)C(d)}\end{aligned}$$

   The first term above is strictly less than 1 for $x \neq \epsilon$; the second term tends to zero as $n \to \infty$ because the number of documents d with lengths $n < |d| \leq n + |x|$ is fixed as n increases, and because of the requirement we placed on C: we can view this fixed number k of documents as k sequences of increasingly long elements in A^*, thus $C(d) \to 0$ for each sequence as $n \to \infty$. Hence $\phi(\hat{x}) < 1$; we also have $\phi(\hat{\epsilon}) = 1$.

2. Consider $s_n = \sum_{a\in A} \|P_n\hat{a}\|_1$. The document d with $|d| \leq n+1$ contributes exactly $|d|C(d)$ to s_n since there are $|d|$ symbols in d. Thus

$$\frac{s_n}{\|P_n\hat{\epsilon}\|_1} = \frac{\sum_{|d|\leq n+1} |d|C(d)}{\sum_{|d|\leq n}(|d|+1)C(d)}.$$

   By a similar argument to the one above, this is strictly less than 1 as $n \to \infty$, hence $\sum_{a\in A} \phi(\hat{a}) < 1$.

3. We say ϕ is lattice defined for $u \in L^\infty(A^* \times A^*)$ if ϕ is defined for u^+ and u^-; clearly ϕ is lattice defined for context vectors. If ϕ is lattice defined for $u, v \in L^\infty(A^* \times A^*)$ then since

$$u + v = u^+ - u^- + v^+ - v^- \leq u^+ + v^+$$

we have $(u+v)^+ \leq u^+ + v^+$, and hence $\|P_n(u+v)^+\|_1 \leq \|P_n u^+\|_1 + \|P_n v^+\|_1$ and

$$0 \leq \phi((u+v)^+) \leq \phi(u^+) + \phi(v^+).$$

Similarly, $-u - v = u^- - u^+ + v^- - v^+ \leq u^- + v^-$ and hence $(u+v)^- \leq u^- + v^-$ and $0 \leq \phi((u+v)^-) \leq \phi(u^-) + \phi(v^-)$. Thus ϕ is lattice defined for $u + v$ if it is defined for u and v. Clearly it is lattice defined for αu if it is lattice defined for u: if $\alpha > 0$ then $(\alpha u)^+ = \alpha u^+$ and $(\alpha u)^- = \alpha u^-$; if $\alpha < 0$ then $(\alpha u)^+ = |\alpha| u^-$ and $(\alpha u)^- = |\alpha| u^+$. We can define the vector operations in terms of the positive and negative parts: $|u| = u^+ + u^-$ and

$$\begin{aligned} u \wedge y &= \frac{1}{2}(u + v - |u - v|) \\ u \vee y &= \frac{1}{2}(u + v + |u - v|). \end{aligned}$$

Thus ϕ is lattice defined for vectors formed from lattice defined elements using the lattice operations. By induction, ϕ is lattice defined for any vector formed from context vectors using a finite number of vector addition, multiplication by a scalar and lattice operations; hence ϕ is defined for such vectors.

□

The condition on C for this proposition is a light one — it is almost inevitable in computational linguistics that longer strings have lower probability, thus requiring the probability of a string to tend to zero as its length goes to infinity is a very natural requirement to place on C.

In contrast to our definition, when talking about the "probability of a string" in the context of language modelling, we would expect to find the property $\sum_{a \in A} \phi(\hat{a}) = 1$. We can explain this by interpreting the value $\phi(\hat{x})$ in the following way. Consider a machine that outputs strings according to the probability distribution C, and at the end of each string outputs an additional symbol to denote the end of the document. Then $\phi(\hat{x})$ is the probability that if you stop the machine at a random point, the next $|x|$ symbols output by the machine will form the string x.

From an information-theoretic perspective, if we wished to encode the corpus model we would need an additional symbol to denote the end of a document; we can think of

this additional symbol as absorbing the lost probability. While this property might seem inconvenient, it is essential that $\phi(\epsilon)$ has the value 1 in order for us to relate the definition to non-commutative probability, which we discuss later in the chapter.

3.1.4 Degrees of Entailment

As we discussed previously, the performance of many tasks in computational linguistics rests on our ability to determine entailment between strings. Thus ultimately we are interested in being able to determine whether one string entails another based on their context vectors. We propose that rather than having a black and white measure of entailment there should be *degrees* of entailment. We are now in a position to define such a measure based on the context-theoretic probability. Like Glickman and Dagan (2005) we believe that entailment is closely connected to the nature of conditional probability. This is what we would expect from a Bayesian perspective; according to the Bayesian philosophy the correct formalism for reasoning about uncertainty is the mathematics of probability; from this perspective conditional probability can be viewed as a Bayesian implication. Because the l^1 norm together with the lattice operations define a an AL-space, the following definition of entailment has all the properties of a conditional probability:

Definition 3.5 (Degree of Entailment). The degree of entailment $\mathrm{Ent}(x, y)$ between two strings x and y is defined as

$$\mathrm{Ent}(x, y) = \frac{\phi(\hat{x} \wedge \hat{y})}{\phi(\hat{x})}$$

when $\phi(\hat{x}) \neq 0$, and is undefined otherwise. This value is a measure of the degree to which the contexts string x occurs in are shared by the contexts string y occurs in. According to this definition, complete entailment exists between x and y when $\mathrm{Ent}(x, y) = 1$, which will be true when $x \leq y$. There will be no degree of entailment when $\mathrm{Ent}(x, y) = 0$, which is true when $x \wedge y = 0$.

As we will see in the following chapters, this definition provides us with a unified measure of the degree of entailment for any implementation of the framework we will define, based on the context-theoretic philosophy.

3.1.5 Multiplication on Contexts

So far we have not got any closer to determining how we should combine vector representations of words to get vector representations of phrases and sentences. In this section we compare the properties of the representation of strings of words to the representation of the individual words, and we are able to show that our model places strong restrictions

between the two. These requirements form part of the framework and informs us as to how the meaning of strings relates to the meanings of individual words in the context-theoretic philosophy.

A crucial feature of our definition is that it applies to strings as well as to individual symbols: strings of any size are attributed with a context vector. In particular, given two strings x and y, not only do the strings have their own context vectors, but their concatenation xy has a context vector $\widehat{xy}$ associated with it. What we will show is that context vectors can be considered as elements of an *algebra over the real numbers*, that is, a real vector space with multiplication defined on it such that the multiplication is *bilinear* and *associative* (see section A.5).

Specifically, the question we are addressing is: does there exist some algebra $\mathcal{A}$ containing the context vectors of strings in A^* such that $\hat{x} \cdot \hat{y} = \widehat{xy}$ where $x, y \in A^*$ and $\cdot$ indicates multiplication in the algebra? As a first try, consider the vector space $L^1(A^* \times A^*)$ in which the context vectors live. Is it possible to define multiplication on the whole vector space such that the condition just specified holds?

Consider the corpus C on the alphabet $A = \{a, b, c, d, e, f\}$ defined by $C(abcd) = C(aecd) = C(abfd) = 1$ and $C(x) = 0$ for all other $x \in A^*$. Now if we take the shorthand notation of writing the basis vector in $L^1(A^* \times A^*)$ corresponding to a pair of strings as the pair of strings itself then

$$\begin{aligned} \hat{b} &= (a, cd) + (a, fd) \\ \hat{c} &= (ab, d) + (ae, d) \\ \widehat{bc} &= (a, d) \end{aligned}$$

It would thus seem sensible to define multiplication of contexts so that $(a, cd) \cdot (ab, d) = (a, d)$. However we then find

$$\hat{e} \cdot \hat{f} = (a, cd) \cdot (ab, d) \neq \widehat{ef} = 0$$

showing that this definition of multiplication doesn't provide us with what we are looking for. In fact, if there did exist a way to define multiplication on contexts in a satisfactory manner it would necessarily be far from intuitive, as, in this example, we would have to define $(a, cd) \cdot (ab, d) = 0$ meaning the product $\hat{b} \cdot \hat{c}$ would have to have a non-zero component derived from the products of context vectors (a, fd) and (ae, d) which don't relate at all to the contexts of bc.

3.1.6 Multiplication on the Generated Subspace

As an alternative to the approach of defining multiplication directly on contexts, we can consider instead defining multiplication on a *subspace* of $L^\infty(A^* \times A^*)$, specifically the subspace generated by all context vectors. This is in fact the subspace we are interested in, since in general we are interested in the relationships between meanings of words, described in terms of their context vectors. Because we are interested in the context theoretic probability ϕ of strings, we will extend ϕ to all vectors in this subspace by requiring it to be linear: $\phi(\alpha_1\hat{x}_1+\alpha_2\hat{x}_2) = \alpha_1\phi(x_1)+\alpha_2\phi(x_2)$ for all $\alpha \in \mathbb{R}$ and $x_1, x_2 \in A^*$. Note that this doesn't contradict the earlier definition of ϕ because of the properties of the l^1 norm which ϕ is defined with respect to. We might want to consider infinite sums of context vectors, but we will not be interested in those which have infinite context theoretic probability, so we define the subspace $\mathcal{A}$ that we are interested in as follows:

Definition 3.6 (Generated Subspace $\mathcal{A}$). The subspace $\mathcal{A}$ of $L^\infty(A^* \times A^*)$ is the set defined by

$$\mathcal{A} = \{a : a = \sum_{x \in A^*} \alpha_x \hat{x} \text{ for some } \alpha_x \in \mathbb{R} \text{ and } \phi(a) < \infty\}$$

Because of the way we define the subspace, there will always exist some basis $\mathcal{B} = \{\hat{u} : u \in B\}$ where $B \subseteq A^*$, and we can define multiplication on this basis by $\hat{u} \cdot \hat{v} = \widehat{uv}$ where $u, v \in B$. Defining multiplication on the basis defines it for the whole vector subspace, since we define multiplication to be linear, making $\mathcal{A}$ an algebra.

However there are potentially many different bases we could choose, each corresponding to a different subset of A^*, and each giving rise to a different definition of multiplication. Remarkably, this isn't a problem:

Proposition 3.7 (Context Algebra). *Multiplication on $\mathcal{A}$ is the same irrespective of the choice of basis B.*

Proof. We say $B \subseteq A^*$ defines a basis $\mathcal{B}$ for $\mathcal{A}$ when $\mathcal{B} = \{\hat{x} : x \in B\}$. Assume there are two sets $B_1, B_2 \subseteq A^*$ that define corresponding bases $\mathcal{B}_1$ and $\mathcal{B}_2$ for $\mathcal{A}$. We will show that multiplication in basis $\mathcal{B}_1$ is the same as in the basis $\mathcal{B}_2$.

We represent two basis elements $\hat{u}_1$ and $\hat{u}_2$ of $\mathcal{B}_1$ in terms of basis elements of $\mathcal{B}_2$:

$$\hat{u}_1 = \sum_i \alpha_i \hat{v}_i \quad \text{and} \quad \hat{u}_2 = \sum_j \beta_j \hat{v}_j,$$

for some $u_i \in B_1$, $v_j \in B_2$ and $\alpha_i, \beta_j \in \mathbb{R}$. First consider multiplication in the basis $\mathcal{B}_1$. Note that $\hat{u}_1 = \sum_i \alpha_i \hat{v}_i$ means that $C(xu_1y) = \sum_i \alpha_i C(xv_iy)$ for all $x, y \in A^*$. This

includes the special case where $y = u_2 y'$ so

$$C(xu_1u_2y') = \sum_i \alpha_i C(xv_iu_2y')$$

for all $x, y' \in A^*$. Similarly, we have $C(xu_2y) = \sum_j \beta_j C(xv_jy)$ for all $x, y \in A^*$ which includes the special case $x = x'v_i$, so $C(x'v_iu_2y) = \sum_j \beta_j C(x'v_iv_jy)$ for all $x', y \in A^*$. Inserting this into the above expression yields

$$C(xu_1u_2y) = \sum_{i,j} \alpha_i\beta_j C(xv_iv_jy)$$

for all $x, y \in A^*$ which we can rewrite as

$$\hat{u}_1 \cdot \hat{u}_2 = \widehat{u_1u_2} = \sum_{i,j} \alpha_i\beta_j(\hat{v}_i \cdot \hat{v}_j) = \sum_{i,j} \alpha_i\beta_j\widehat{v_iv_j}.$$

Conversely, the product of u_1 and u_2 using the basis $\mathcal{B}_2$ is

$$\hat{u}_1 \cdot \hat{u}_2 = \sum_i \alpha_i\hat{v}_i \cdot \sum_j \beta_j\hat{v}_j = \sum_{i,j} \alpha_i\beta_j(\hat{v}_i \cdot \hat{v}_j)$$

thus showing that multiplication is defined independently of what we choose as the basis. □

Returning to the previous example, we can see that in this case multiplication is in fact defined on $L^1(A^* \times A^*)$ since we can describe each basis vector in terms of context vectors:

$$\begin{aligned}
(a, fd) \cdot (ae, d) &= (\hat{b} - \hat{e}) \cdot (\hat{c} - \hat{f}) = -(a, d) \\
(a, cd) \cdot (ae, d) &= \hat{e} \cdot (\hat{c} - \hat{f}) = (a, d) \\
(a, fd) \cdot (ab, d) &= (\hat{b} - \hat{e}) \cdot \hat{f} = (a, d),
\end{aligned}$$

thus confirming what we predicted about the product of $\hat{b}$ and $\hat{c}$.

3.1.7 Discussion

The fact that context vectors live in an algebra has profound implications for the nature of meaning according to the context-theoretic philosophy. The essential property is distributivity: the vector representations of two strings can be decomposed into components such that the vector representation of the concatenation of strings is sum of the distributed product of the components.

This is in fact a strong requirement to place on the nature of meaning. It means that if two words share a component of meaning, that component will remain in common between them when they are concatenated with another string (unless the component becomes zero on concatenation).

For example, we may assume the word *square* has some component of meaning in common with the word *shape*. Then we would expect this component to be preserved in the sentences *He drew a square* and *He drew a shape*. However, in the case of the two sentences *The box is square* and **The box is shape* we would expect the second to be represented by the zero vector since it is not grammatical; *square* can be a noun and an adjective, whereas *shape* cannot. Distributivity of meaning means that the component of meaning that *square* has in common with *shape* must be disjoint with the adjectival component of the meaning of *square*.

As we will see however, this requirement does not prevent us from representing many important properties of meaning in natural language; rather it provides us with guidelines as to how best to represent meaning according to the context-theoretic philosophy.

3.1.8 Non-commutative Probability

We already stated that it is important for us that our framework is well grounded in probability theory. The context theoretic probability ϕ already defines a probability space with respect to the vector lattice, however the results of the previous section allow us to think about the algebra $\mathcal{A}$ as an entirely different probabilistic structure, a *non-commutative probability space*.

Definition 3.8 (Non-commutative Probability). A non-commutative probability space is a unital algebra (an algebra with unity 1) together with a linear functional ψ such that $\psi(1) = 1$.

In our definition, $\hat{\epsilon}$ is a unity of the algebra, and the linear functional ϕ which we called the context theoretic probability satisfies $\phi(\hat{\epsilon}) = 1$, and thus $\mathcal{A}$ together with ϕ defines a non-commutative probability space. This means that we can think of context vectors as forming a probability space in two ways: they have a measure theoretic probability structure in terms of their vector lattice properties, and a non-commutative probability structure in terms of their algebraic properties. Both of these probability spaces are defined with respect to the context theoretic probability ϕ. It is these key properties that will use to form the basis of the definition of our framework for context-theoretic semantics; they form a strong set of requirements on the nature of a mathematical structure for representing meaning. As we will see in the second part of the thesis, they provide ample room for representing meaning as we are familiar with it in computational linguistics,

while still providing strong guidelines as to how to construct representations of meaning based on the context-theoretic philosophy.

3.1.9 Further Work

There are some unanswered questions relating to the theoretical properties of the model we have just described. We have shown that lattice operations can be defined on the vector space of possible contexts, and we have also shown that multiplication can be defined on the subspace of this vector space generated by context vectors to form an algebra; we have not, however, defined multiplication on the vector space generated by the lattice operations. This would be useful for us to show since this would make the space a *lattice-ordered* algebra; all the implementations of the context-theoretic framework have this structure, thus we have included it in the framework. Proving that this structure is inherent in the model of meaning as context would give further justification for its inclusion. Instead we make the following conjecture:

Conjecture 3.9. Let $\mathcal{A}^{\wedge\vee}$ denote the vector lattice generated by a context algebra $\mathcal{A}$ under the lattice operations. There exists some multiplication on $\mathcal{A}^{\wedge\vee}$ that is an extension of the multiplication of $\mathcal{A}$ that makes it a lattice-ordered algebra.

Our attempts to prove this conjecture have not yet succeeded, nor have we been able to find a counter-example to disprove it.

Another interesting theoretical question relating to the model is the question of completeness with respect to the norm defined by the linear functional ϕ; if the vector space is complete with respect to this norm then we would have a Banach space (see Section A.2.3). What the implications of this for the nature of meaning as context would be however are unclear; as far as we can see answering this question either way would have little impact on the practical use of the framework, though of course the long term benefits of answering such theoretical questions are hard to predict.

3.2 The Context-theoretic Framework

In this section we define the context theoretic framework based on the theory of meaning as context we have just discussed; the framework is formed from the central mathematical properties of the theory. These properties are derived from the assumption that the meaning of a string is purely determined by context; because of this, we can think of implementations of the framework as describing a theory about the contexts a string can occur in — for this reason we call such implementations "context theories".

There are certain things we require of the framework: it must provide guidelines about

how to represent phrases and sentences, about determining the probability of a string and determining the degree of entailment between strings. With this in mind, looking at the theory of meaning as context we can find a set of properties of the theory that we wish to incorporate into the framework:

- Words and strings of words should be represented as vectors. We may wish to make use of techniques such as latent semantic analysis to derive vector representations of words; this ensures that such representations can be incorporated, but places the requirement that strings of words are also represented by vectors, based on our analysis in the previous chapter.

- The vector space should in addition have a lattice structure. As we have seen, it is the lattice structure that informs us about entailment between strings, it is thus essential that this structure be incorporated into the framework.

- The representation of the concatenation of strings can be viewed as a product of the representations of the individual strings for some distributive product (i.e. the vector space forms an algebra); this is a strong requirement to place on the mathematical structure. Imposing this structure is justified by the analysis of meaning as context and not only simplifies things from a mathematical perspective, but potentially opens up the vast amount of research available on these structures to be applied to computational linguistics.

- There is a linear functional ϕ (the context-theoretic probability) on the vector space such that the lattice operations together with ϕ can be used to define an AL-space. This requirement ensures that ϕ behaves like a probability with respect to the lattice operations. This is important since the degree of entailment is defined in terms of ϕ and the lattice operations. We wish the degree of entailment to have the form of a conditional probability, and placing this requirement ensures that this will be the case for any implementation of the framework.

- The algebra together with ϕ defines a non-commutative probability space. The benefits of this requirement are less clear at the moment, however there is evidence from our previous analysis that imposing it is justified. In fact, this requirement will become more relevant later in the thesis when we discuss the representation of syntactic structure in the framework. Later we will hypothesise that the syntactic and semantic aspects of the representation of a word should combine *freely* in the algebra. The study of non-commutative probability spaces has centred on this notion of *free probability*, which is similar to the idea of independence in commutative variables.

We are able to combine these properties within the following definition:

Definition 3.10 (Context Theory). A context theory for an alphabet A is a unital lattice-ordered algebra $\mathcal{A}$ together with a semigroup homomorphism from A^* to $\mathcal{A}$, denoted $a \mapsto \hat{a}$ and a positive linear functional ϕ such that $\phi(\hat{\epsilon}) = 1$.

In addition we shall also often require that the set $I = \{u : \phi(u) = 0\}$ is a sub-vector lattice of $\mathcal{A}$ — that is, a subspace of $\mathcal{A}$ that is a vector lattice under the same partial ordering. We call a context theory that satisfies this condition a *strong context theory.*

This definition incorporates all the properties we require:

- A string x is represented by the vector $\hat{x}$; requiring this to be a semigroup homomorphism ensures that we can view strings as elements of an algebra.
- We require that the algebra is lattice-ordered. While the lattice structure is essential, requiring a lattice-ordered algebra is a stronger requirement; this would be justified in our theory if the conjecture at the end of the last chapter is proven correct. We have made this requirement since in practice it is not a limitation: all the structures we will describe in the second half of this thesis are naturally lattice-ordered algebras.
- requiring $\phi(\hat{\epsilon}) = 1$ makes $\mathcal{A}$ together with ϕ a non-commutative probability space;
- we can define a norm on $\mathcal{A}$ based on ϕ that makes it an AL-space:

Proposition 3.11 (ϕ-norm). *Given a context theory $\mathcal{A}$ with positive linear functional ϕ such that $I = \{u : \phi(u) = 0\}$ is a sub-vector lattice of $\mathcal{A}$ we can define a norm $\|\cdot\|_\phi$ on $\mathcal{A}/I$ that defines an AL-space:*

$$\|u\|_\phi = \phi(u^+) + \phi(u^-)$$

Proof. The space $\mathcal{A}/I$ is the quotient space $\mathcal{A}/\equiv$ where $u \equiv v$ if $u, v \in I$, and is a vector lattice under the ordering of $\mathcal{A}$ (Aliprantis and Burkinshaw, 1985).[1] The linear functional ϕ is well defined on this quotient space and satisfies $\phi(u) = 0$ if and only if u is the zero of the quotient space. We need to show that $\|\cdot\|_\phi$ has the properties of a norm. For all $u \in \mathcal{A}/I$ we have:

- Positivity: $\|u\|_\phi \geq 0$ since ϕ is positive, and both u^+ and u^- are positive.
- Positive scalability: for $\alpha \in \mathbb{R}$, if $\alpha > 0$ then $\|\alpha u\|_\phi = \alpha\phi(u^+) + \alpha\phi(u^-)$. If $\alpha < 0$ then $(\alpha u)^+ = -\alpha u^-$ and $(\alpha u)^- = -\alpha u^+$ so $\|\alpha u\|_\phi = -\alpha\phi(u^+) - \alpha\phi(u^-)$. If $\alpha = 0$ then $\|\alpha u\|_\phi = 0$, thus for all $\alpha \in \mathbb{R}$, $\|\alpha u\|_\phi = |\alpha| \cdot \|u\|_\phi$.

[1] Effectively, it is the space formed by setting the subspace I to zero.

- Triangle inequality: we have $(u+v)^+ \leq u^+ + v^+$ and $(u+v)^- \leq u^- + v^-$. Then $\|u+v\|_\phi = \phi((u+v)^+) + \phi((u+v)^-) \leq \phi(u^+ + v^+) + \phi(u^- + v^-) = \|u\|_\phi + \|v\|_\phi$.
- Positive definiteness: it follows from $\phi(u) = 0$ if and only if $u = 0$ that $\|u\|_\phi = 0$ if and only if $u = 0$.

Finally, for $u, v \in \mathcal{A}$ with $u \geq 0$ and $v \geq 0$ we have $\|u+v\|_\phi = \phi(u+v) = \|u\|_\phi + \|v\|_\phi$ thus $\|\cdot\|_\phi$ defines an AL-space on $\mathcal{A}/I$. □

The requirements that we placed on a context theory ensured that the space is a probability space in two ways. In particular, the definition of the degree of entailment that we defined previously in the form of a conditional probability applies equally well in the case of a context theory. We restate it here for the case of a context theory:

Definition 3.12 (Degree of Entailment). The degree of entailment $\mathrm{Ent}(x, y)$ between two strings x and y is defined as

$$\mathrm{Ent}(x, y) = \frac{\phi(\hat{x} \wedge \hat{y})}{\phi(\hat{x})}$$

when $\phi(\hat{x}) \neq 0$, and is undefined otherwise.

Part II

Context-theoretic Semantics for Natural Language

Chapter 4

Textual Entailment

In this chapter we examine the task of recognising textual entailment from the context-theoretic perspective. Textual entailment recognition is the task of determining, given two sentences, whether the first sentence entails or implies the second. The task is particularly well suited to the application of the context-theoretic framework since it concerns detecting entailment between strings of words, which is what a context theory predicts. However, the task requires determining the existence or non-existence of entailment while from the context-theoretic perspective it is more accurate to talk about a *degree* of entailment between strings. Nevertheless we will show later in the chapter how several existing approaches to the task relate to the framework.

4.1 The Recognising Textual Entailment Challenge

The task of recognising textual entailment has reached prominence recently with the launch of the Recognising Textual Entailment Challenge (Dagan et al., 2005a; Bar-Haim et al., 2006), in which participants develop systems to analyse pairs of sentences to automatically determine whether entailment exists. An example from the development set of the third challenge is the pair

1. UK Foreign Secretary Jack Straw said Iraqis had "shown again their determination to defy the terrorists and take part in the democratic process".

2. Jack Straw holds the position of UK Foreign Secretary.

In this case entailment does hold, since we can deduce the content of the second sentence (called the *hypothesis*) from the first (called the *text*); see Table 4.1 for more examples.

It is immediately clear from the generality of this task that a wide range of language processing tools and resources are required to tackle this problem comprehensively; this is demonstrated in the approaches that have been attempted in the two Recognising Textual Entailment Challenges (see Table 4.2):

Task	Text	Hypothesis	Ent.
IE	A bus collision with a truck in Uganda has resulted in at least 30 fatalities and has left a further 21 injured.	30 die in a bus collision in Uganda.	Yes
IR	Chirac needed a new mandate for his government from the electorate, or a new left government was needed that could count on the support of the trade union bureaucracy and among the working class and so would encounter less resistance.	Parliamentary elections create new government in France.	No
QA	Brazilian cardinal Dom Eusbio Oscar Scheid, Archbishop of Rio de Janeiro , harshly criticized Brazilian President Luiz Incio Lula da Silva after arriving in Rome on Tuesday.	The Brazilian President is Luiz Incio Lula da Silva.	Yes
SUM	The mine would operate nonstop seven days a week and use tons of cyanide each day to leach the gold from crushed ore.	A weak cyanide solution is poured over it to pull the gold from the rock.	No

Table 4.1: Sample text and hypothesis sentences from the Third Recognising Textual Entailment Challenge and whether entailment is judged to hold between them, with examples from the sub-tasks of information extraction (IE), information retrieval (IR), question answering (QA) and summarisation (SUM).

Challenge	Corpus / web-based statistics	Lexical relation DB	Syntactic matching	World knowledge / Paraphrase templates	Logical inference	Total number of submissions
RTE-1	13	10	13	3	7	28
RTE-2	22	32	28	5	2	41
Total (%)	51%	61%	59%	12%	13%	100%

Table 4.2: Number of submitted runs using various techniques in Recognising Textual Entailment Challenges 1 and 2 (RTE-1 and RTE-2 respectively).

- **Morphological and syntactic analysis:** various levels of analysis have been performed in tackling this task, ranging from simple stemming of words to full dependency parsing of sentences. 59% of runs submitted to both challenges used some kind of syntactic analysis.

- **Lexical semantic knowledge:** An even greater proportion of runs submitted, 61% made use of a lexical relations database such as WordNet, while 51% of runs used corpus or web-based statistics such as measures of distributional similarity. Indeed it seems that the major focus of approaches to the task to date has been on analysis at the lexical level, while deep semantic analysis has received far less attention.

- **Inference and world knowledge:** Only 13% of submitted runs in both challenges, and only 2 runs in the second challenge used some kind of logical inference. This could be because of the complexity of implementing the task, with no choice but to deal with problems such as anaphora resolution and lexical ambiguity. However, we believe that deeper semantic analysis is necessary to achieve high accuracy in this task: overall accuracy is low in current approaches, with no team achieving greater than 75%. Deep semantic analysis and use of world knowledge are areas that have not been explored fully; indeed, we will argue that current systems have not achieved their full potential because of failure to deal effectively with the ambiguity and uncertainty inherent in analysis of natural language.

It does seem that in order to perform well at this task it is necessary to combine various tools and techniques: the two best performing entries to the second entailment challenge improved the accuracy of their systems in this way. One of these (Hickl et al.,

2006) achieved 75% accuracy using a system that essentially treated the task as a classification problem; in doing so however, they made use of a part-of-speech tagger, parser, named entity recogniser, semantic parser for determining dependency relations, a lexical alignment system, and a method of acquiring paraphrases from the world-wide web. It is not clear however whether all these components are essential to achieving this level of accuracy in their system. Tatu and Moldovan's (2006) system achieves 74% accuracy by combining a simple lexical alignment method with a deep semantic analysis using world knowledge and inference.

Despite the benefits of combining several techniques, part of the attraction of the task is that very simple methods perform relatively well. For example, in the second Recognising Textual Entailment Challenge, Zanzotto et al. (2006) achieve 60% accuracy purely by measuring lexical overlap between the text and hypothesis sentences.

In the following sections, we will describe some of the approaches to this task, concentrating specifically on those that we believe can benefit most from the context-theoretic approach.

4.1.1 Glickman and Dagan's Probabilistic Setting

We believe it is vital when implementing a system that it is based within a framework with a firm theoretical foundation. The framework provides guidance at each stage of construction of the system and ensures that decisions that are taken in implementing it are made in a consistent, logical manner.

It is also important for us that the theoretical foundation of a textual entailment system be *linguistic* in nature; that is, the framework provides guidance as to how to deal with language. Conversely, many approaches to the task of recognising textual entailment make use of a framework which requires abstraction of the task to a level where language is irrelevant — an example of this is systems such as that of Hickl et al. (2006) which treat the problem as one of classification of pairs of sentences. Whilst their approach is successful, it provides no insight into the linguistic nature of textual entailment because of the framework in which it is based; instead it provides engineering insight about the task itself and approaches to the task.

We also believe that such a framework should be grounded in the mathematics of probability. Statistical approaches to dealing with language have proved successful in dealing with syntax and, to a degree, lexical semantics because of the possibility of gaining wide coverage by using large amounts of data, and providing robustness, for example by allowing the representation of uncertainty of the correctness of a parse. Thus it is only natural that a framework for textual entailment should also allow for such representation of

uncertainty, and the mathematics of probability is the most established and well-founded way of doing this.

The work of Glickman and Dagan (2005) is of great interest to us because they describe a probabilistic framework which is also linguistic in nature, and deals specifically with the nature of textual entailment. As we will see however, in our opinion their framework is not ideal and leaves areas which our context-theoretic framework can improve upon.

Their framework is defined as follows: let T denote the set of possible texts and H the set of possible hypotheses. The set W denotes the set of all mappings from H to $\{0, 1\}$; it is called the set of *possible worlds*, and each element of W is interpreted as assigning truth values of true (1) or false (0) to elements of H.

The authors describe their setting as follows:

> We assume a probabilistic generative model for texts and possible worlds. In particular, we assume that texts are generated along with a concrete state of affairs represented by a possible world. Thus, whenever the source generates a text $t \in T$, it generates also corresponding hidden truth assignments that constitute a possible world $w \in W$.
>
> The probability distribution of the source, over all possible texts and truth assignments $T \times W$, is assumed to reflect inferences that are based on the generated texts. That is, we assume that the distribution of truth assignments is not bound to reflect the state of affairs in a particular "real" world, but only the inferences about the proposition's truth which are related to the text.

The term "possible world" is perhaps confusing, as it is apparently not intended to relate to any type of "world", merely assignments of truth values to elements of H. Thus the authors' setting states that for each text t, there is some conditional probability distribution $P(\mathrm{Tr}_h = 1|t)$ over truth assignments to hypotheses; $P(\mathrm{Tr}_h = 1|t)$ denotes the probability that hypothesis h is true given that the text t has been generated. The authors consider entailment to exist between t and h if this probability is greater than the prior probability of h being true, $P(\mathrm{Tr}_h = 1)$.

4.1.2 Lexical Entailment Model

Glickman and Dagan apply their probabilistic setting using a simple model of entailment based on occurrences of words in web documents. In order to do this, they allow individual words to be assigned truth values; they suggest a possible interpretation for this as the existence of a concept related to the word, so that $\mathrm{Tr}_{\mathrm{book}} = 1$ when text t is generated if "it can be inferred in t's state of affairs that a book exists". A hypothesis is considered to

be true if all its component words are true; in addition it is assumed that the probabilities of individual terms being true in a hypothesis are independent of each other:

$$P(\mathrm{Tr}_h = 1) = \prod_{u \in h} P(\mathrm{Tr}_u = 1)$$
$$P(\mathrm{Tr}_h = 1|t) = \prod_{u \in h} P(\mathrm{Tr}_u = 1|t)$$

where the product is over all words u in h, considered as a bag or multiset of words. In order to estimate $P(\mathrm{Tr}_u = 1|t)$ for a given word u in the hypothesis, they assume that "the majority of the probability mass comes from a specific entailing word in t:

$$P(\mathrm{Tr}_u = 1|t) = \max_{v \in t} P(\mathrm{Tr}_u = 1|T_v)$$

where T_v denotes the the event that a generated text contains the word v." Finally, they make the assumption that "all hypotheses stated verbatim in a document are true and all others are false and hence $P(\mathrm{Tr}_u = 1|T_v) = P(T_u|T_v)$". That is, the probability of a hypothesis (word) u being true given that a document containing the word v is generated is just the probability that a document contains word u given that it contains word v. These values can easily be estimated based on frequency counts:

$$P(T_u|T_v) \simeq \frac{n_{u,v}}{n_v}$$

where $n_{u,v}$ is the number of documents containing both u and v, and n_v is the number of documents containing v.

For the first Recognising Textual Entailment Challenge, the authors used estimates of these probabilities based on frequency counts from web search engines; their system achieved 59% accuracy, one of the best scores achieved in the first challenge.

4.1.3 Analysis of Glickman and Dagan's Approach

Glickman and Dagan's framework aims to achieve the same as we wish to achieve, namely, the incorporation of the representation of uncertainty into logical reasoning. This includes the representation of all kinds of uncertainty involved in recognising textual entailment, as they state:

> An implemented model that corresponds to our probabilistic setting is expected to produce an estimate for $P(\mathrm{Tr}_h = 1|t)$. This estimate is expected to reflect all probabilistic aspects involved in the modelling, including inherent uncertainty of the entailment inference itself..., possible uncertainty regard-

> ing the correct disambiguation of the text. . . , as well as probabilistic estimates that stem from the particular model structure.

Their framework requires combining knowledge about generation of text with reasoning about the probability of the *truth* of propositions. An example they give that illustrates this is the sentence "His father was born in Italy", which would entail with high probability the sentence "He was born in Italy". However, according to the authors, examining the texts containing the sentence "His father was born in Italy", we find that in these texts the son was more often *not* born in Italy. Hence, in their framework, the probability of entailment would be low, since the probability of the second sentence being true, given the generation of the first sentence, is low.

From our perspective, there are several problems with their framework:

- The combination of the probability of truth of propositions with generation of text is confusing. For someone implementing a textual entailment recognition system within their framework, it is unclear how these probabilities are to be obtained. For example, in the authors' implementation, they assume that "all hypotheses stated verbatim in a document are true and all others are false". If this assumption were made from the start then the framework could refer solely to the probability of generation of text, a familiar concept with clear interpretation.

- The framework requires the hypothesis to be interpretable as a logical proposition. Many textual entailment implementations do not make use of logical representations, however, including the authors' own implementation. This forces them to allow truth values to be assigned to words, which is not ideal since there is no satisfactory interpretation of what it means for a word to be "true".

- Because of the limitation just mentioned, their framework does not make predictions about the entailment of phrases or words, only sentences.

In our opinion, the authors' framework is a step in the right direction, but the insistence on interpreting hypothesis in a logical fashion is a limitation in the context of textual entailment, which our framework overcomes.

4.1.4 Logical Approaches

Textual entailment recognition systems that make use of logic rarely take the straightforward approach of translating a sentence into a logical form and seeing if the representation of the text logically entails the representation of the hypothesis; in fact no entry to either challenge took such an approach, while those that came closest to doing so (Akhmatova,

Author(s)	Parser	Inference technique	Inference system(s)	Accuracy (Coverage)
Akhmatova (2005)	Link Parser	Theorem proving to detect entailment between atomic propositions	OTTER	52%
Bayer et al. (2005)	Link Parser, MITRE dependency analyser	Probabilistic inference	EPILOG	52% (73%)
Delmonte et al. (2005)	VENSES	Score based comparison of semantic representations	VENSES	61% (62%)
Fowler et al. (2005)	Unknown	Theorem proving with scores for dropped predicates / relaxed arguments	COGEX (based on OTTER)	55%
Raina et al. (2005)	Klein and Manning (2003)	Abductive theorem proving with costs, classifier	EPILOG	56%
Bos and Markert (2006)	CCG-parser (Bos, 2005)	Theorem proving and model building, decision tree	Vampire, Paradox, Mace	61%
Tatu and Moldovan (2006)	MINIPAR	Theorem proving with scores for dropped predicates / relaxed arguments, lexical alignment, classifier	COGEX	74%

Table 4.3: Summary of logical approaches to textual entailment. Coverage indicates the proportion of pairs for which the system returned answers, if not 100%.

2005; Delmonte et al., 2005) are not robust enough to perform well at the task. It seems logical representations are inherently brittle and on their own are not suited to the flexibility of reasoning that is required to deal with natural language. To get around this problem, several strategies have been employed (see also Table 4.3):

- **Score / cost based systems:** In logical representations a single proposition in the hypothesis that is not in the text will prevent entailment from holding; this is an example of the brittleness of logical representations. Score based systems (Delmonte et al., 2005; Fowler et al., 2005; Raina et al., 2005; Tatu and Moldovan, 2006) address this, by relaxing the conditions on entailment holding, but adding a "cost" or score to such relaxations (for example the addition of an extra proposition to the hypothesis) to indicate a lack of certainty about entailment holding.

  This effectively allows "degrees" of entailment, where the degree is determined by the score. It is practically useful, since it allows more flexibility in determining the existence of entailment, however it is theoretically unfounded, and thus questions remain as to exactly how the scores are to be assigned, what values they are to take, and how they can be interpreted.

- **Classsification based systems:** Another approach (Raina et al., 2005; Tatu and Moldovan, 2006) often used in combination with scoring is to treat detection of entailment between pairs of sentences as a classification problem: the task is to classify such pairs as either showing entailment or not. The results of logical inference would then be considered as one "feature" of the pair, which together with other features, (for example word overlap), would form the input to a classifying system. The parameters of the classifier are then determined by training on the development set of pairs.

  This approach is useful since it allows different techniques to be combined by describing them as features; the weaknesses of one technique can be compensated for by strengths of another. For example, measuring word overlap is a robust technique, but not terribly accurate, so in cases where logical analysis fails, word overlap provides a good back-up measure.

  The problem with the clasification approach is that it is not tackling the problem at its source, merely compensating for failures in each technique with other, also imperfect techniques. Instead of trying to understand the nature of entailment, this is a useful way of engineering systems to do the best with the techniques at hand.

  Ultimately, it seems hard to imagine such a system performing extremely well, if each of the component "features" involved are really flawed measures of entailment.

It would always be possible to think of example pairs of sentences which fall outside the range of the development set and thus exploit flaws in each component system.

In addition, it seems unlikely that such analyses will bring us closer to understanding the nature of language or textual entailment itself, instead it will merely provide us with insight as to how best approach the *task* of recognising textual entailment using existing techniques.

- **Model building:** Another interesting approach (Bos and Markert, 2006) is to build models of T and $T \wedge H$, where T and H are the logical representations of the text and hypothesis, if they are satisfiable, and compare the sizes of the models built. If $T \wedge H$ has a model that is not much larger than T then it is reasonable to assume that a lot of the information in the hypothesis is also contained in the text, and thus that the text entails the hypothesis; the result is again a relaxing of the conditions for entailment allowing degrees of entailment based on the comparison of model sizes.

  Again this approach lacks a firm theoretical foundation however, and thus questions such as how to best measure model size are unanswered — for example, a measure that one might use is domain size, which measures the number of entities in the model. In addition to this Bos and Markert use the product of the domain size and the number of all instances of relations in the model as a measure of model size.

- **Probabilistic reasoning:** The EPILOG system used by Bayer et al. (2005) allows reasoning about probabilistic statements such as "If x is a person, then with probability ≥ 0.95, x lives in a building." (Kaplan, 2000). It is not clear however, if and how Bayer et al. incorporate this feature into their system; moreover their system performs poorly in terms of robustness and accuracy.

4.2 Context Theories for Textual Entailment

The only existing framework for textual entailment that we are aware of is that of Glickman and Dagan (2005). However this framework does not seem to be general enough to deal satisfactorily with many techniques used to tackle the problem since it requires interpreting the hypothesis as a logical statement.

Conversely, systems that use logical representations of language are often implemented without reference to any framework, and thus deal with the problems of representing the ambiguity and uncertainty that is inherent in handling natural language in an ad-hoc fashion.

Thus it seems what is needed is a framework which is general enough to satisfactorily incorporate purely statistical techniques and logical representations, and in addition provide guidance as to how to deal with ambiguity and uncertainty in natural language. It is this that we hope our context-theoretic framework will provide.

In this section we analyse approaches to the textual entailment problem, showing how they can be related to the context-theoretic framework.

4.2.1 Document Projections

Glickman and Dagan (2005) give a probabilistic definition of entailment in terms of "possible worlds" which they use to justify their lexical entailment model based on occurrences of words in web documents. They estimate the lexical entailment probability $\textsc{lep}(u, v)$ to be

$$\textsc{lep}(u, v) \simeq \frac{n_{u,v}}{n_v}$$

where n_v and $n_{u,v}$ denote the number of documents that the word v occurs in and the words u and v both occur in respectively. From the context theoretic perspective, we view the set of documents the word occurs in as its context vector. To describe this situation in terms of a context theory, consider the vector space $L^\infty(D)$ where D is the set of documents. With each word u we associate an operator P_u on this vector space by

$$P_u e_d = \begin{cases} e_d & \text{if } u \text{ occurs in document } d \\ 0 & \text{otherwise.} \end{cases}$$

where e_d is the basis element associated with document $d \in D$. P_u is a projection, that is $P_u P_u = P_u$; it projects onto the space of documents that u occurs in. These projections are clearly commutative (they are in fact *band projections*): $P_u P_v = P_v P_u = P_u \wedge P_v$ projects onto the space of documents in which both u and v occur.

In their paper, Glickman and Dagan assume that probabilities can be attached to individual words, as we do, although they interpret these as the probability that a word is "true" in a possible world. In their interpretation, a document corresponds to a possible world, and a word is true in that world if it occurs in the document.

They do not, however, determine these probabilities directly; instead they make assumptions about how the entailment probability of a sentence depends on lexical entailment probability. Although they do not state this, the reason for this is data sparseness: they assume that a sentence is true if all its lexical components are true: this will only happen if all the words occur in the same document. For any sizeable sentence this is extremely unlikely, hence their alternative approach.

It is nevertheless useful to consider this idea from a context theoretic perspective. The probability of a term being true can be estimated as the proportion of documents it occurs in. This is the same as the context theoretic probability defined by the linear functional ϕ, which we may think of as determined by a vector p in $L^\infty(D)$ given by $p(d) = 1/|D|$ for all $d \in D$. In general, for an operator U on $L^\infty(D)$ the context theoretic probability of U is defined as

$$\phi(U) = \|U^+ p\|_1 - \|U^- p\|_1$$

The probability of a term is then $\phi(P_u) = n_u/|D|$. More generally, the context theoretic representation of an expression $x = u_1 u_2 \dots u_m$ is $P_x = P_{u_1} P_{u_2} \dots P_{u_m}$. This is clearly a semigroup homomorphism (the representation of xy is the product of the representations of x and y), and thus together with the linear functional ϕ defines a context theory for the set of words.

The degree to which x entails y is then given by $\phi(P_x \wedge P_y)/\phi(P_x)$. This corresponds directly to Glickman and Dagan's entailment "confidence" without the additional assumptions they make; it is simply the proportion of documents that contain all the terms of x which also contain all the terms of y.

This formulation suggests an alternative approach to that of Glickman and Dagan to cope with the data sparseness problem. We consider the finite data available D as a sample from a corpus model D'; the vector p then becomes a probability distribution over the documents in D'. In our own experiments, we used Latent Dirichlet Allocation (Blei et al., 2003), which builds a generative model of a corpus from a finite sample, allowing us to estimate probabilities of occurrences in an infinite corpus. The results we obtained using this method on the data from the first Recognising Textual Entailment Challenge were comparable to those obtained by Glickman and Dagan.[1]

4.2.2 Subsequence Matching and Lexical Overlap

We call a sequence $x \in A^*$ a "subsequence" of $y \in A^*$ if each element of x occurs in y in the same order, but with the possibility of other elements occurring in between, so for example *abba* is a subsequence of *acabcba* in $\{a, b, c\}^*$. We denote the set of subsequences of x (including the empty string) by $\mathrm{Sub}(x)$. Subsequence matching compares the subsequences of two sequences: the more subsequences they have in common the more similar they are assumed to be. This idea has been used successfully in text classification (Lodhi et al., 2002) and also formed the basis of the author's entry to the second Recognising Textual Entailment Challenge (Clarke, 2006).

[1] In order to evaluate the technique in relation to the challenge, a cutoff value had to be chosen for the "degree" of entailment for which entailment was considered to exist.

If S is a semigroup, $L^1(S)$ is a lattice ordered algebra under the multiplication of convolution:

$$(f \cdot g)(x) = \sum_{yz=x} f(y)g(z)$$

where $x, y, z \in S$, $f, g \in L^1(S)$. For a sequence $x \in A^*$, we define $\hat{x} \in L^1(A^*)$ by

$$\hat{x} = (1/2^{|x|}) \sum_{y \in Sub(x)} e_y,$$

where e_y is the unit basis element associated with y; that is, the function that is 1 on y and 0 elsewhere. This is clearly a semigroup homomorphism and thus together with the linear functional ϕ,

$$\phi(u) = \|u^+\|_1 - \|u^-\|_1$$

defines a context theory for A. Under this context theory, a sequence x completely entails y if and only if it is a subsequence of y. In our experiments, we have shown that this type of context theory can perform significantly better than straightforward lexical overlap. Many variations on this idea are possible, for example using more complex mappings from A^* to $L^1(A^*)$.

The simplest approach to textual entailment is to measure the degree of lexical overlap: the proportion of words in the hypothesis sentence that are contained in the text sentence. Though simple, variations on this approach can perform comparably to much more complex techniques (Dagan et al., 2005a).

The free commutative semigroup on a set A is $A^*/\equiv$ where $x \equiv y$ in A^* if the symbols making up x can be reordered to make y. Following the reasoning of subsequence matching, for a sequence x we can define $\hat{x} \in L^1(A^*/\equiv)$ by

$$\hat{x} = (1/2^{|x|}) \sum_{y \in Sub(x)} e_{[y]},$$

where $[y]$ is the equivalence class of y in $A^*/\equiv$. Defining a linear functional similarly gives us a context theory in which entailment depends on the words in the sequences but not their order. Again, more complex definitions of $\hat{x}$ can be used, for example to weight different words by their probabilities.

Chapter 5

Uncertainty in Logical Semantics

The standard approach to representing meaning in natural language is to represent sentences of the language by some logical form. This is useful in situations where it is necessary to perform in-depth reasoning, however it brings with it many problems. Such systems require accurate parses of sentences in order to reason effectively, yet existing parsers do not provide sufficient accuracy or coverage; parsers will return a probability distribution over possible parses, however to our knowledge there is currently no principled way to reason with such probabilities. Similarly, these systems typically need to know which sense of a word was intended by the speaker in a particular context; the task of word sense disambiguation attempts to determine this, however current performance at this task is poor. Whilst there are many problems encountered by such systems, these are two that we will look at in this chapter; it is our hope however that it will be possible to generalise the ideas presented here to deal with other problems in a similar way.

It is possible that these problems are inherent in the nature of language: perhaps in general there is not one correct parse for a sentence, nor is there only ever one sense of a word that can apply in a particular context. In this case it is vital that representations of the meaning of natural language can incorporate such uncertainty. Alternatively, it may be that these problems will be eventually be solved satisfactorily; in the meantime we need a way to deal with the uncertainty that results from using these techniques.

To our knowledge, existing methods of representing uncertainty and ambiguity are founded in formal semantics and have no way to incorporate statistical information such as the probability of a parse or the probability of the sense of a word. In our opinion it is vital to make use of this information when dealing with natural language because there are so many opportunities for uncertainty; it makes sense to quantify the uncertainty in meaning representations since in general it can be quantified in existing techniques. In this chapter we show how the context-theoretic framework can be used to reason about uncertainty making use of statistical information, first giving a theoretical analysis of the problem, and then outlining how the ideas can be implemented practically.

The approach we take in this chapter is to formulate logical semantics within the context-theoretic framework; this gives us the flexibility of vector spaces that we need to represent statistical information about uncertainty. For example, in the representation we will present the meaning of a word can be viewed as a weighted sum of its individual senses

Instead of dealing with a specific version of model-theoretic semantics, we give a very general treatment that can deal with any system in which strings are translated into a logical form with an implication relation defined on it; thus the ideas presented here can be applied to just about any conceivable logical system.

The contributions of this chapter are as follows:

- In section 5.1 we show how logical semantics can be interpreted in a context-theoretic manner: given a way of translating natural language expressions into logical forms, we can define an algebra which represents the meaning equivalently. Given also a way of attaching probabilities to logical forms (which can be given a Bayesian interpretation), we can define a context theory allowing us to deduce degrees of entailment between expressions.

- In section 5.2 we show how the vector-based representation allows statistical information about uncertainty of meaning and ambiguity to be incorporated; the representation of an ambiguous sentence is a weighted sum over the vector representations of its unambiguous meanings. These may be the result of syntactic ambiguity, such as multiple parses returned by a statistical parser, due to ambiguous words or some other source of uncertainty.

- The presentation we give is theoretical in nature and it is not clear how the ideas can be implemented in a concrete manner. To overcome this problem, in Section 5.3 we outline how this may be done, to show how a system may be built to compute a degree of entailment between two sentences.

- Most of this chapter relates to the representation of natural language sentences that are translated into logical form, however to demonstrate the general applicability of the context-theoretic framework, we show how (in Section 5.2.4), given a logical representation of sentences, entailment can be defined between words and phrases, based on a context-theoretic analysis of the situation.

5.1 From Logical Forms to Algebra

Model-theoretic approaches generally deal with a subset of all possible strings, the language under consideration, translating sequences in the language to a logical form, ex-

pressed in another, logical language. Relationships between logical forms are expressed by an entailment relation on this logical language.

This section is about the algebraic representation of logical languages. By a *logical language* we mean a language $\Lambda \subset A'^*$ for some alphabet A', together with a relation $\vdash$ on Λ that is reflexive and transitive; this relation is interpreted as entailment on the logical language. We will show how each element $u \in \Lambda$ can be associated with a projection on a vector space; it is these projections that define the algebra. Later we will show how this can be related to strings in the natural language λ that we are interested in.

For a subset T of a set S, we define the projection P_T on $L^\infty(S)$ by

$$P_T e_s = \begin{cases} e_s & \text{if } s \in T \\ 0 & \text{otherwise} \end{cases}$$

Where e_s is the basis element of $L^\infty(S)$ corresponding to the element $s \in S$. Given $u \in \Lambda$, define $\downarrow_\vdash(u) = \{v : v \vdash u\}$. As a shorthand we write P_u for the projection $P_{\downarrow_\vdash(u)}$ on the space $L^\infty(\Lambda)$.

The projection P_u can be thought of as projecting onto the space of logical statements that entail u. This is made formal in the following proposition:

Proposition 5.1. *$P_u \leq P_v$ if and only if $u \vdash v$.*

Proof. Clearly

$$(*) \qquad P_u P_v e_w = \begin{cases} e_w & \text{if } w \vdash u \text{ and } w \vdash v \\ 0 & \text{otherwise} \end{cases},$$

so if $u \vdash v$ then since $\vdash$ is transitive, if $w \vdash u$ then $w \vdash v$, so we must have $P_u P_v = P_u$. The projections P_u and P_v are band projections, so $P_u P_v = P_u$ if and only if $P_u \leq P_v$.

Conversely, if $P_u P_v = P_u$ then it must be the case that $w \vdash u$ implies $w \vdash v$ for all $w \in \Lambda$, including $w = u$. Since $\vdash$ is reflexive, we have $u \vdash u$, so $u \vdash v$ which completes the proof. □

To help us understand this representation better, we wil show that it is closely connected to the ideal completion of partial orders (see Proposition A.28). Define a relation $\equiv$ on Λ by $u \equiv v$ if and only if $u \vdash v$ and $v \vdash u$. Clearly $\equiv$ is an equivalence relation; we denote the equivalence class of u by $[u]$. Equivalence classes are then partially ordered by $[u] \leq [v]$ if and only if $u \vdash v$. Then note that $\bigcup \downarrow([u]) = \downarrow_\vdash(u)$, thus P_u projects onto the space generated by the basis vectors corresponding to the elements $\bigcup \downarrow([u])$, the ideal completion representation of the partially ordered equivalence classes.

What we have shown here is that logical forms can be viewed as projections on a vector space. Since projections are operators on a vector space, they are themselves vectors; viewing logical representations in this way allows us to treat them as vectors, and we have all the flexibility that comes with vector spaces: we can add them, subtract them and multiply them by scalars; since the vector space is also a vector lattice, we also have the lattice operations of meet and join. As we will see in the next section, in some special cases such as that of the propositional calculus, the lattice meet and join coincide with logical conjunction and disjunction.

5.1.1 Application: Propositional Calculus

In this section we apply the ideas of the previous section to an important special case: that of the propositional calculus. We choose as our logical language Λ the language of a propositional calculus with the usual connectives $\vee$, $\wedge$ and $\neg$, the logical constants $\top$ and $\bot$ representing "true" and "false" respectively, with $u \vdash v$ meaning "infer v from u", behaving in the usual way. Then:

$$P_{u \wedge v} = P_u P_v \qquad P_{\neg u} = 1 - P_u + P_\bot$$
$$P_{u \vee v} = P_u + P_v - P_u P_v \qquad P_\top = 1$$

To see this, note that the equivalence classes of $\vdash$ form a Boolean algebra under the partial ordering induced by $\vdash$, with

$$[u \wedge v] = [u] \wedge [v] \qquad [u \vee v] = [u] \vee [v] \qquad [\neg u] = \neg [u].$$

Note that while the symbols $\wedge$, $\vee$ and $\neg$ refer to logical operations on the left hand side, on the right hand side they are the operations of the Boolean algebra of equivalence classes; they are completely determined by the partial ordering associated with $\vdash$.[1]

Since the partial ordering carries over to the ideal completion we must have

$$\downarrow [u \wedge v] = \downarrow [u] \cap \downarrow [v] \qquad \downarrow [u \vee v] = \downarrow [u] \cup \downarrow [v]$$

Since $u \vdash \top$ for all $u \in \Lambda$, it must be the case that $\downarrow [\top]$ contains all sets in the ideal completion. However the Boolean algebra of subsets in the ideal completion is larger than the Boolean algebra of equivalence classes; the latter is embedded as a Boolean subalgebra of the former. Specifically, the least element in the completion is the empty set,

[1] In the context of model theory, the Boolean algebra of equivalence classes of sentences of some theory T is called the *Lindenbaum-Tarski* algebra of T (Hinman, 2005).

whereas the least element in the equivalence class is represented as $\downarrow[\bot]$. Thus negation carries over with respect to this least element:

$$\downarrow[\neg u] = (\downarrow[\top] - \downarrow[u]) \cup \downarrow[\bot] .$$

We are now in a position to prove the original statements:

- Since $\downarrow[\top]$ contains all sets in the completion, $\bigcup \downarrow[\top] = \downarrow_{\vdash}(\top) = \Lambda$, and $P_\top$ must project onto the whole space, that is $P_\top = 1$.
- Using the above expression for $\downarrow[u \wedge v]$, taking unions of the disjoint sets in the equivalence classes we have $\downarrow_{\vdash}(u \wedge v) = \downarrow_{\vdash}(u) \cap \downarrow_{\vdash}(v)$. Making use of $(*)$ in the proof to Proposition 5.1, we have $P_{u \wedge v} = P_u P_v$.
- In the above expression for $\downarrow[\neg u]$, note that $\downarrow[\top] \supseteq \downarrow[u] \supseteq \downarrow[\bot]$. This allows us to write, after taking unions and converting to projections, $P_{\neg u} = 1 - P_u + P_\bot$, since $P_\top = 1$.
- Finally, we know that $u \vee v \equiv \neg(\neg u \wedge \neg v)$, and since equivalent elements in Λ have the same projections we have

$$\begin{aligned} P_{u \vee v} &= 1 - (P_{\neg u \wedge \neg v}) + P_\bot \\ &= 1 - (P_{\neg u} P_{\neg v}) + P_\bot \\ &= 1 - (1 - P_u + P_\bot)(1 - P_v + P_\bot) + P_\bot \\ &= P_u + P_v - P_u P_v - 2P_\bot + P_\bot P_u + P_\bot P_v \\ &= P_u + P_v - P_u P_v \end{aligned}$$

It is also worth noting that in terms of the vector lattice operations $\vee$ and $\wedge$ on the space of operators on $L^\infty(\Lambda)$, we have $P_{u \vee v} = P_u \vee P_v$ and $P_{u \wedge v} = P_u \wedge P_v$.

5.2 Representing Uncertainty

There are many situations in computational linguistics where we are inevitably faced with uncertainty. For example, given a long sentence in English, the chances of a parser choosing one of the parses that would be identified by humans as correct is minimal; with modern statistical parsers we are left instead with a probability distribution over parses. Similarly, the task of word sense disambiguation is a long way from being solved; again, the best we can do is assign a probability distribution over the senses of a word given a

particular context. Thus it seems that syntactic and lexical ambiguity are problems that can only be partially solved, at least using current techniques.

This syntactic and lexical ambiguity results in *semantic ambiguity* — we are uncertain of the intended meaning of an expression. We believe the correct approach to this problem (in the context of traditional, logical representations of language) is to incorporate ambiguity into the representation of meaning itself. This would allow us to reason about our uncertainty of the intended meaning of expressions in a logical manner, and ultimately lead to *semantic disambiguation*, in which the semantic ambiguity of an expression is reduced by analysing the possible meanings resulting from syntactic and lexical ambiguity.

We are particularly interested in incorporating statistical information about uncertainty. As far as we are aware, the problem of representing uncertainty and ambiguity in the context of statistical methods has not been tackled theoretically. We will show how the context theoretic approach, together with the ideas developed in the last section lead almost directly to a method of dealing with semantic ambiguity.

5.2.1 Requirements for Representing Ambiguity

In the context of logical representations of meaning, there are certain properties that we would expect from representations of ambiguity; we list and discuss these here.

Bayesianism

We would expect our representation to be tied closely to Bayesian reasoning, since this is the standard approach to reasoning with uncertainty. Bayesianism asserts that the correct calculus for modelling uncertainty is the mathematics of probability theory. A "probability" assigned to a sentence is then merely taken as an indication of our certainty of the truth of the sentence; it is not intended to be a scientifically measurable quantity in the sense that probabilities are often assumed to be. We would expect to be able to incorporate such probabilities into our system

Ambiguity and Logic

When dealing with ambiguity in the context of logical representations, we expect certain relationships between the representation of ambiguity and the logical representations. Specifically, if we have an ambiguous expression with two meanings, we would expect the ambiguous representation to entail the logical disjunction of two expressions. In general, we would not expect the converse, since the two are not equivalent. To see this, for example, consider the sentence s = "He saw a plant". We wish to represent the lexical

ambiguity in the word "plant" which we will consider can either mean an industrial plant or an organism. The two disambiguated meanings roughly correspond to the sentences s_1 = "He saw an industrial plant" and s_2 = "He saw a plant organism". We expect that each disambiguated sentence s_i entails the ambiguous sentence s, if not completely, then to some degree.

Statistical Features

Similarly, we would expect the ambiguous sentence to entail the disambiguated meanings to the degree that we expect the ambiguous word to carry the relevant sense. For example, if "plant" is used in the sense of industrial plant 40% of the time, then we would expect that s entails s_1 to degree 0.4.

5.2.2 Representing Bayesian Uncertainty

The projection representation of translations to logical form allows us to associate an algebra (of projections) with the logical language Λ, however it does not quite give us a context theory. For that, we need a linear functional on the algebra of projections, and we will show how this can be done if we take a Bayesian approach to reasoning.

We need to associate probabilities with (logical) sentences in a way that is compatible with their logical structure. This can be done using a probability distribution over the sentences of the logical language:

Let Λ be a logical language with entailment relation $\vdash$, a probability distribution p over elements of Λ and a distinguished element $\bot \in \Lambda$ such that

- $\bot \vdash u$, and
- $p(u) = 0$ if $u \vdash \bot$,

for all $u \in \Lambda$. We call this a probabilistic logical translation for Λ. For an arbitrary subset X of Λ, define $p(X) = \sum_{u \in X} p(u)$. For $u \in \Lambda$ define $p_\vdash(u) = p(\downarrow_\vdash(u))$. Then:

Proposition 5.2. *The function $p_\vdash$ defines a probability measure on the lattice defined by $\downarrow_\vdash$. Specifically:*

1. $p_\vdash(\bot) = 0$
2. *if* $\downarrow_\vdash(u) \cap \downarrow_\vdash(v) = \downarrow_\vdash(\bot)$ *then* $p(\downarrow_\vdash(u) \cup \downarrow_\vdash(v)) = p_\vdash(u) + p_\vdash(v)$

Proof.

1. $p_\vdash(\bot) = \sum_{u \vdash \bot} p(u) = 0$.

2. if $\downarrow_{\vdash}(u) \cap \downarrow_{\vdash}(v) = \downarrow_{\vdash}(\bot)$ then $p(\downarrow_{\vdash}(u) \cup \downarrow_{\vdash}(v)) = p_{\vdash}(u) + p_{\vdash}(v) - p_{\vdash}(\bot) = p_{\vdash}(u) + p_{\vdash}(v)$

□

That is, $p_{\vdash}$ behaves like a probability measure with respect to the partial order structure induced by $\vdash$. We can now define a linear functional on the algebra of projections:

Definition 5.3. Given a probability distribution p over a logical language Λ, we define a vector $\hat{p}$ in $L^{\infty}(\Lambda)$ by

$$\hat{p} = \sum_{u \in \Lambda} p(u) e_u$$

We define a linear functional ϕ on the space of bounded operators on $L^{\infty}(\Lambda)$ by

$$\phi(F) = \|F_{+}(\hat{p})\|_1 - \|F_{-}(\hat{p})\|_1$$

where F_{+} and F_{-} are the positive and negative parts of the bounded operator F respectively.

Proposition 5.4. *If $u \in \Lambda$ for some logical language Λ with probability distribution p, then $\phi(P_u) = p_{\vdash}(u)$*

Proof.

$$\phi(P_u) = \|P_u \hat{p}\|_1 = \sum_{v \in \downarrow_{\vdash}(u)} p(v) = p_{\vdash}(u)$$

□

Using the linear functional we can define a context theory for logical sentences. Since context theories are defined in terms of an alphabet, we have to define it in terms of a finite set A of symbols with each symbol corresponding to a sentence in Λ. We associate a bounded operator $\hat{x}$ on the space $L^{\infty}(S)$ with each element $x \in A$: we have $\hat{x} = P_u$, where u is the logical sentence corresponding to x; thus we have a context theory.

5.2.3 Representing Syntactic Ambiguity

One of the major problems facing engineers of natural language systems is how to deal with syntactic ambiguity. Most modern wide-coverage parsers will return many parses for a single sentence, together with a probability distribution over these parses. How are we to make use of this probability distribution while reasoning with the logical representations of the sentences?

We can interpret the probability distribution as an indication of the likelihood of the parse applying in different contexts: in some contexts the sentence occurs in one parse may be favoured over another. Thus we can view the context vector of a sentence as the sum of the context vectors of its senses. We can interpret the probability given to each parse by the parser as contributing to its context theoretic probability, thus the representation of a sentence s is a weighted sum of the representation of its individual senses s_i:

$$\hat{s} = \sum_i p(s_i)\hat{s}_i,$$

where $p(s_i)$ is the probability of parse s_i. If we use the logical representation described in the previous section, the probability given by the linear functional ϕ will be

$$\phi(\hat{s}) = \sum_i p(s_i)p_{\vdash}(u_i)$$

where u_i is the logical representation of sense s_i; that is the probability of the sentence is the weighted sum of the probability of the meaning of its senses. Note that because of the vector lattice based framework, we are able to take probabilistic sums of the representations of sentences, and the lattice operations are still well defined, enabling us to calculate the degree of entailment between two sentences represented in this way.

This recipe can of course be applied more generally to deal with other forms of uncertainty: for example, any uncertainty about lexical ambiguity, anaphora resolution, part of speech tagging etc. can be incorporated into a probabilistic sum of the resulting semantic representations of the different analyses.

5.2.4 A Context Theoretic Analysis of Logical Representations

The algebraic description of logic given in the previous section is useful for giving us an intuition of how logic can be interpreted geometrically as projections, however it can only deal with descriptions of logic at the sentence level. It would be useful in addition to have a description of the logical representation of language in terms of vectors that also told us how words and phrases should be represented. Such a description would allow us to examine representations of phrases and sentences and compute entailment between them. In this section, we show how such a description can be constructed. This will allow us to represent a word in terms of a sum of its senses, where the logical behaviour of each individual sense is well defined, and provide us with a deeper insight into the relationship between model-theoretic and context-theoretic descriptions of meaning.

In order to represent words however, we are going to need a more comprehensive

representation. There are potentially many ways to do this, for example, we could attempt to construct an algebra in terms of semigroups that contains the properties we are looking for. The approach we will describe, however, is highly context-theoretic in nature, bearing many similarities to the context vectors of a string defined previously.

Let us assume we have some language $\lambda \subseteq A^*$ and a probabilistic logical translation from λ to Λ with probability distribution p and entailment relation $\vdash$.

Definition 5.5. Let x be a string in A^*. Then we define the function $\tilde{x}$ from $A^* \times A^*$ to $L^1(\Lambda)$ by

$$\tilde{x}(a,b) = \begin{cases} \sum_{u \in \downarrow(\mu(axb))} p(u) e_u & \text{if } axb \in \lambda \\ 0 & \text{otherwise.} \end{cases}$$

The function $\tilde{x}$ maps a context (a, b) to a vector representing the sum of all the logical representations that entail the logical translation of axb. Note that since $\tilde{x}$ is a function to a vector lattice, it can be viewed itself as a vector lattice, with the vector and lattice operations defined point-wise: for example $(\tilde{x} + \tilde{y})(a,b) = \tilde{x}(a,b) + \tilde{y}(a,b)$, $(\alpha\tilde{x})(a,b) = \alpha\tilde{x}(a,b)$ and $(\tilde{x} \wedge \tilde{y})(a,b) = \tilde{x}(a,b) \wedge \tilde{y}(a,b)$.

We also define a linear functional φ on the vector space by

$$\varphi(u) = \|u_+(\epsilon,\epsilon)\|_1 - \|u_-(\epsilon,\epsilon)\|_1$$

This description in terms of functions incorporates information about entailments between sentences, whilst remaining entirely context-theoretic in nature. The next proposition shows how logical and probabilistic properties of sentences of λ are preserved in the vector representation.

Proposition 5.6. *If $x, y \in \lambda$ and $\mu(x) \vdash \mu(y)$ then $\mathrm{Ent}(x,y) = 1$; if p is non-zero everywhere on Λ except $\bot$ then the converse also holds. Moreover, if $x \in \lambda$, then $\varphi(\tilde{x}) = p_\vdash(\mu(x))$*

Proof. If $x, y \in \lambda$ and $\mu(x) \vdash \mu(y)$ then clearly $\tilde{x}(\epsilon,\epsilon) \leq \tilde{y}(\epsilon,\epsilon)$, so $\mathrm{Ent}(x,y) = 1$. Conversely, if $\mathrm{Ent}(x,y) = 1$ then by the definition of φ it must be the case that $0 < \tilde{x}(\epsilon,\epsilon) \leq \tilde{y}(\epsilon,\epsilon)$, hence $x, y \in \lambda$. If p is non-zero everywhere on Λ then the only way this can be true is if $\mu(x) \vdash \mu(y)$. To see this, assume that $\mu(x) \nvdash \mu(y)$; then there exists an element of $\downarrow(\mu(x))$ that is not in $\downarrow(\mu(y))$, hence $\tilde{x}(\epsilon,\epsilon)$ will be non-zero in a component for which the corresponding component of $\tilde{y}(\epsilon,\epsilon)$ will be zero; hence $\tilde{x}(\epsilon,\epsilon) \not\leq \tilde{y}(\epsilon,\epsilon)$, thus by contradiction, $\mu(x) \vdash \mu(y)$. □

Thus we have a vector based description of the language which preserves the logical and probabilistic nature of the translation, however we have not yet shown that this description

is context-theoretic in nature. To do this, we will show a very close relationship between the description and the definition of meaning in terms of context we used in the discussion on corpus models in chapter 3. We will need a more general definition than that used previously however: we define a *general corpus model* on an alphabet A to be a positive real-valued function over A^*. The definition of the context vector $\hat{x}$ of a string $x \in A^*$ still holds with a general corpus model; and again the vector space $\mathcal{A}$ generated by all such vectors is an algebra under the multiplication defined by concatenation of strings. What we will show is that a general corpus model can be associated with every probabilistic logical translation of a language.

Proposition 5.7. *Given a probabilistic logical translation T of a language $\lambda \subseteq A^*$ to $\Lambda \subseteq A'^*$ there exists a general corpus model C_T over an alphabet B and a one-to-one function ψ from the space V of functions from $A^* \times A^*$ to $L^1(\Lambda)$ to $L^\infty(B^* \times B^*)$ such that $\psi(\tilde{x}) = \hat{x}$.*

Proof. Let $B = A \cup A' \cup \{\diamond\}$ where $\diamond$ is an additional symbol, $\diamond \notin A \cup A'$. Define C_T by:

$$C_T(x \diamond m) = p(m)$$

for all $x \in \lambda$ and all $m \in \downarrow(\mu(x))$, and C is zero for all other elements of B^*. Let $u(a, b, m)$ be the basis element of V which maps $(a, b) \in A^* \times A^*$ to e_m in $L^1(\Lambda)$ and maps all other elements of $A^* \times A^*$ to 0. Then we define ψ by its operation on these basis elements:

$$\psi(u(a, b, m)) = e_{(a,b \diamond m)}.$$

Because $\diamond$ is not in A or A' this function must be one to one. Then

$$\psi(\tilde{x}) = \sum_{a,b\,:\,axb \in \lambda} \left[\sum_{m \in \downarrow(\mu(axb))} p(m) e_{(a,b \diamond m)} \right] = \hat{x}.$$

□

This is an important result since it means that given a logical description of a language, we can construct a general corpus model incorporating this logical description, allowing us to make a strong link between logical and context-theoretic approaches. It also means that since the vector space $\mathcal{A}$ generated by the context vectors of C is an algebra, the vector space $\mathcal{A}'$ generated by the vectors $\{\tilde{x} : x \in A^*\}$ is also an algebra, again with multiplication defined by concatenation: $\tilde{x} \cdot \tilde{y} = \widetilde{xy}$. This is guaranteed to be well defined since it is well defined in the vector space $\mathcal{A}$ defined by C_T.

5.2.5 Semantic Corpus Models

According to the context theoretic framework we have developed, the linear functional φ when applied to the vector representation of a string is supposed to give the "probability" of that string. Clearly there is no such concept in model-theoretic semantics — we can attach probabilities to logical forms giving them a Bayesian interpretation, but the concept of a probability of a string itself is foreign to model theoretic semantics. In fact the linear functional φ we have defined behaves exactly like this: when x is a sentence of the language λ, $\varphi(x)$ is the probability of the logical representation of x; if x is not in the language, $\varphi(x)$ is zero; thus φ does not conform to the context-theoretic ideal.

Yet if we are to truly find a way to combine context-theoretic techniques with model-theoretic approaches, we must find a way to link the concept of a probability of a string with these logical approaches; we should look for a linear functional that behaves more like a probability while still not ignoring the model theoretic nature of the representation.

In the linear functional φ we are only using one context (ϵ, ϵ); one way to give a non-zero probability to phrases that aren't sentences would be to consider other contexts. However here we face a practical problem; if we use all contexts, the value is not guaranteed to be finite. One solution is to think of the probability of a string as being composed of two parts: the probability of the meaning of the string, and the probability that the meaning is expressed in that particular way. We can describe this by a probability distribution $q(x)$ over elements of λ. We interpret this value as the conditional probability of observing string x given that a string with a logical translation that entails the logical translation of string x has been observed. Thus q satisfies the requirement

$$\sum_{x \in A^*} \left(\sum_{u \in \downarrow(\mu(x))} p(u) \right) q(x) = 1.$$

We then give a new definition of the representation of a string in the vector space: it is still a function from $A^* \times A^*$ to $L^1(\Lambda)$; we define

$$\tilde{x}_q(a, b) = \begin{cases} \sum_{u \in \downarrow(\mu(axb))} q(axb)p(u)e_u & \text{if } axb \in \lambda \\ 0 & \text{otherwise.} \end{cases}$$

When we use the general corpus model translation we defined in the previous section, we must define

$$C(x \diamond m) = q(x)p(m)$$

for all $x \in \lambda$ and all $m \in \downarrow(\mu(x))$, with C zero for all other elements of B^*. Because

of the requirement we placed on q, we must have $\|C\|_1 = \sum_{u \in B^*} C(u) = 1$, so C is a corpus model. Having a corpus model allows us to use the original linear functional ϕ defined for corpus models to measure the probability of a string. How are we to interpret this probability? We can think of C as a "semantic" corpus model: it generates strings according to the probability p of their meaning as well as the probability q that this meaning is expressed in that particular way.

5.2.6 Representing Lexical Ambiguity

The work of the previous section gives us the tools with which to describe lexical ambiguity within the context-theoretic framework. We are interested in descriptions of word sense ambiguity that allow us to incorporate statistical information about the probabilities of different senses and reason about these in a way that is consistent with the context-theoretic philosophy.

Let us take a simple model of word sense ambiguity in which each word w takes a finite number n of senses $S(w) = \{w_1, w_2, \ldots w_n\}$. We assume that given a particular context (a, b), we know which sense of the word is intended: each context is associated with exactly one sense in $S(w)$ so that the context completely disambiguates w. We can associate with each sense w_i a set $[w_i]$ of contexts in which the word w takes sense w_i. Similarly, given a corpus model C we can define a context vector $\hat{w}_i$ with each sense w_i which represents the contexts that that particular sense of w occurs in:

$$\hat{w}_i(a, b) = \begin{cases} \hat{w}(a, b) = C(awb) & \text{if } (a, b) \in [w_i] \\ 0 & \text{otherwise.} \end{cases}$$

Given this definition, we see that the context vectors of the senses of a word are disjoint in the vector lattice, and the context vector of a word is equal to the sum of the context vector of its senses: $\hat{w} = \sum_i \hat{w}_i$. Note that, just as we would expect, the context-theoretic probability of a word is the sum of the probability of its individual senses. Note also that the senses are disjoint because we assumed that each context completely disambiguated the word; if we relax this condition this will no longer hold, thus this is not necessarily an essential feature, indeed it may not be useful in cases where a word has senses that are closely related.

We can define multiplication of senses with context vectors in a very straightforward way:

$$(\hat{w}_i \cdot u)(a, b) = \begin{cases} (\hat{w} \cdot u)(a, b) & \text{if } (a, b) \in [w_i] \\ 0 & \text{otherwise,} \end{cases}$$

for $u \in \mathcal{A}$, and similarly for left-hand multiplication. This allows us to see how ambiguous words are partially disambiguated as they are concatenated with other words: we have

$$\hat{w} \cdot \hat{x} = \sum_i \hat{w}_i \cdot \hat{x}.$$

Thus as w is concatenated with a string x, its representation remains the sum of its senses multiplied by $\hat{x}$, however since each sense only occurs in a subset of the possible contexts, x has the effect of partially disambiguating w, and the left hand side of the equation becomes more similar to one of the summands.

This analysis provides us with a simple formula for representing a word in terms of its senses, given the methods of the previous sections: we treat each sense exactly as if it were an unambiguous word; build a context-theoretic representation using the senses, then represent the ambiguous word as the sum of the representation of its individual senses. For example, using the ideas of the previous section, we can define a semantic corpus model based on a probabilistic logical translation in which we assume we only ever deal with senses, for which the logical translation can be well defined. We can then represent the word as the sum of the representation of its individual senses. The probabilistic logical translation and the function q can be interpreted as disambiguating the word. There are several kinds of disambiguation that can occur:

- Note that we do not distinguish between different parts of speech when we talk about senses; for example the representation of a word like "book" will include both noun and verb parts. As words are concatenated with this word, only the senses that can make the phrase grammatical (i.e. that occur as a substring of λ) will remain, disambiguating parts of speech.

- The entailment relation and p provide *semantic disambiguation*: in a particular context those senses which lead to sentences which are meaningless and thus whose meaning is assigned a value of 0 by p will be eliminated so that only senses which are meaningful in the context remain. Similarly, senses which produce a meaning in the given context which is very unlikely will be assigned a low probability by p.

- The function q provides *statistical disambiguation* — it reduces emphasis on senses of words which are statistically unlikely based on the context, although the resulting meaning may not be unlikely; thus this function has a rôle similar to current word sense disambiguation techniques.

Thus the framework provides ample room for the representation of word sense ambiguity and its disambiguation.

5.3 Outline of Possible Implementations

We chose to describe the models of the preceding sections in a manner which was extremely general and also mathematically simple. This allowed us to present the concepts clearly without concern for how we could represent and compute with such models. Clearly it is impractical to explicitly represent a sentence as a sum over a (potentially infinite) number of dimensions. Instead, we imagine that in practice, systems that make use of the mathematics we have presented here will make use of standard representations for the logical aspect of the representation; the statistical or algebraic aspects can then be computed separately, while making use of the existing algorithms for computing with logic.

To make this clearer, we will outline how such a system may be constructed. We assume we have at our disposal a method for computing entailments between sentences of the logical language Λ. In most cases, Λ will include the propositional calculus as a subset, and thus the equivalence classes of $\vdash$ will form a Boolean algebra. The main problem facing us is the function p which is defined on sentences of Λ. In fact, we can do without p itself, and assume we have at our disposal the function $p_\vdash$ which will be a probability measure on the Boolean algebra of equivalence classes of $\vdash$. This means we will not have to compute sums of p over sentences of Λ, a potentially impossible task. The function $p_\vdash$ can be assigned in many ways, for example:

- A simple heuristic could be used. For example, this could be an information-theory inspired measure based on the length of the logical expression: $p_\vdash(u) = k^{-|u|}$ where k is a constant and $|u|$ is the length of the shortest member of the equivalence class of $u \in \Lambda$. This would have the advantage of being simple to compute yet fairly consistent with the requirements of $p_\vdash$: in general it is likely that if $u \vdash v$ then $p_\vdash(u) \leq p_\vdash(v)$ will hold since v can be expressed at least as simply as u.

- A value for $p_\vdash$ could be assigned based on a probabilistic logic: this may be a fuzzy logic such as Łukasiewicz logic (Kundu and Chen, 1994) or Basic Fuzzy Logic (Hájek, 1998), or a first order logic such as that of Nilsson (1986) and later variations.

Both these approaches could make use of techniques that assign probabilities to concepts in an ontology; these are described in Chapter 6.

We assume for now that we are only interested in the representation of sentences; all uncertainty is described by a weighted sum over representations of sentences. Given a natural language sentence the system may for example:

- Parse the sentence. A statistical parser such as RASP (Briscoe et al., 2006) can return the top n parses with their probabilities. It includes a statistical part of

speech tagger that retains the most probable parses for each word for input to the parser.

- Perform word sense disambiguation and anaphora resolution. Both of these can result in probabilistic information about which senses and referents are intended.
- Combine probabilistic information from parsing, word sense disambiguation and anaphora resolution information. This would result in a list of interpretations of sentences, each with a specific parse, a sense for each word and a resolved referent for each anaphor, together with a probability. Each interpretation is completely unambiguous and thus ready to be translated into logic. For efficiency purposes, these could be sorted by probability, and only the most probable interpretations retained.
- Translate each interpretation into a logical form.
- Compute the probability $p_{\vdash}$ for each logical translation.

At the end of this process, a sentence s is represented as a list of pairs $\langle u_i, \alpha_i \rangle$, each specifying a logical translation u_i and the probability α_i of the combined statistical information. At this point we can compute probabilities for sentences using the linear functional ϕ:

$$\phi(\tilde{s}) = \sum_i \alpha_i p_{\vdash}(u_i),$$

where $p_{\vdash}(u_i)$ is the probability of the logical sentence u_i. However these probabilities are unlikely to coincide with our normal conception of the probability of a string, since they are the combination of probabilities assigned by the parser and probabilities of logical expressions, which need not necessarily coincide with the probabilities of strings. We can compensate for this however, by making use of a function $q'(u_i|s_i)$ which specifies the probability of the logical expression u_i given that it is translated from the specific interpretation s_i of the sentence under consideration, similar to the function q we discussed earlier — however this is clearly a difficult value to estimate directly. On the other hand, the problem of estimating the probability of a string is well understood; we can make use of one of the many language modelling techniques to do this, for example we could use an n-gram. This value of the probability of a string then provides a renormalising condition which allows us make the vector representing the string fit the expected probability of the string; define a constant $c_s = l(s)/\phi(\tilde{s})$, where $l(s)$ is the probability assigned to the string s by the language model. The renormalised string is then represented by the list of pairs $\langle u_i, \beta_i \rangle$ where $\beta_i = c_s \alpha_i$. The renormalising constant c_s thus plays the role of the function q'.

Computing the degree of entailment between the representations of strings causes some difficulties. This is because we have represented strings as sums over the vector representations of logical sentences which are not disjoint in the vector lattice. Given two such representations $\tilde{s}_1 = \sum_i \beta_i^{(1)} \tilde{u}_i^{(1)}$ and $\tilde{s}_2 = \sum_i \beta_i^{(2)} \tilde{u}_i^{(2)}$, where $\tilde{u}_i^{(k)}$ is the vector representation of the logical expression $u_i^{(k)}$, we need to compute $\phi(\tilde{s}_1 \wedge \tilde{s}_2)$ in order to obtain the degree of entailment. However addition does not distribute with respect to the lattice meet operation except when the addition is between disjoint elements of the vector lattice; since the vector representations of the logical sentences are not in general disjoint, there is no way to find $\tilde{s}_1 \wedge \tilde{s}_2$ in terms of the meets of their summands. The solution to this is to find a set S of disjoint logical sentences such all the u_i can be written as a disjunction of elements of S. This is possible using the canonical form of a Boolean algebra in which each element is written as a join of *minterms* (Birkhoff, 1973). (This could potentially be computationally expensive — given n sentences there could be 2^n minterms.) For disjoint positive elements a and b of a vector lattice, $a \vee b = a + b$, so given the set S each sentence u_i can be written as a sum of disjoint elements, the meet operation becomes trivial and the degree of entailment can easily be computed.

An alternative approach is to compute a lower bound on the degree of entailment. Since $\beta_i^{(k)} u_i^{(k)} \leq \tilde{s}_k$ for each i and s_k, we have $\beta_i^{(1)} \tilde{u}_i^{(1)} \wedge \beta_j^{(2)} \tilde{u}_j^{(2)} \leq \tilde{s}_1 \wedge \tilde{s}_2$ for all i and j, and hence

$$\phi_{\min}(\tilde{s}_1 \wedge \tilde{s}_2) = \max_{i,j} \left[\phi(\beta_i^{(1)} \tilde{u}_i^{(1)} \wedge \beta_j^{(2)} \tilde{u}_j^{(2)}) \right] \leq \phi(\tilde{s}_1 \wedge \tilde{s}_2).$$

The left hand side thus provides us with a lower bound $\phi_{\min}(\tilde{s}_1 \wedge \tilde{s}_2)$ on the context-theoretic probability of the meet of the representations of the two sentences, which can be used to calculate the degree of entailment. This lower bound can be thought of as the greatest probability obtained by taking meets between individual interpretations of the two sentences. It is straightforward to calculate:

$$\phi_{\min}(\tilde{s}_1 \wedge \tilde{s}_2) = \max_{i,j} \left[(\min\{\beta_i^{(1)}, \beta_j^{(2)}\}) p_{\vdash}(u_i^{(1)} \wedge u_j^{(2)}) \right].$$

Note that $\wedge$ here is logical conjunction from the language Λ. This is possible since in a logic with the propositional calculus as a subset, the vector representation of the logical conjunction of two sentences will be the same as the vector lattice meet of the representations of the individual sentences, for the reason discussed in Section 5.1.1. The lower bound on the degree of entailment $\mathrm{Ent}(s_1, s_2)$ is then given by $\phi_{\min}(\tilde{s}_1 \wedge \tilde{s}_2)/\phi(\tilde{s}_1)$.

5.3.1 Entailment between words and phrases

Computing entailment between words and phrases using the ideas of Section 5.2.4 and subsequent sections is clearly more challenging than computing entailment between sentences, since we need to calculate a sum over all contexts $(a, b) \in A^*$. One approach to this problem would be to use a Monte-Carlo technique to estimate the entailment by taking a sample of contexts. In fact, only those contexts which give a sentence in λ will contribute to the sum, and heuristics could be used to skew the sample towards those contexts which are likely to be important for the string under consideration.

5.4 Conclusion

We have presented an context-theoretic analysis of logical semantics for natural language, and shown how the flexibility of the vector representation that comes with the context-theoretic framework allows the incorporation of statistical information about uncertainty into the representation. This provides us with a principled way of reasoning with uncertainty and ambiguity in meaning.

We discussed some requirements that we may expect of a system that represents ambiguity and uncertainty in natural language, namely:

- that the system be able to reason with uncertainty in a probabilistic fashion, following a Bayesian philosophy. The mathematics we have described allows for the incorporation of information about the probability of meaning, in the Bayesian sense.

- that the system deals with ambiguity in a way that agrees with our intuition and that incorporates statistical information about this ambiguity. This is true of the ideas presented here: an ambiguous word or phrase is represented as a weighted sum of its unambiguous meanings. It is the weights given to these meanings that allow statistical information to be incorporated.

While the presentation we have given is mathematical in nature, we have shown how a system may be implemented using the ideas presented here, and outlined how the computational problems involved may be solved.

Finally it should be noted that the approaches presented in this chapter are just a few ways of dealing with the problems of ambiguity and uncertainty in logical semantics within the context-theoretic framework. It is entirely possible that future work within the framework will bring to light new approaches and computational techniques.

Chapter 6

Taxonomies and Vector Lattices

A crucial feature that we require of the context-theoretic framework is that we are able to make use of logical representations of meaning within the framework. Ontologies form an important part of many systems that deal with logical representations of natural language, thus it is important to examine the relationship between ontological representations of meaning and vector based ones. In this chapter we show how an important part of an ontology, a taxonomy, can be represented in terms of vectors in a vector lattice, by means of *vector lattice completions*, a concept that we define. The ideas presented in this chapter marry the vector-based representations of meaning with the ontological ones by considering both from the unifying perspective of vector lattice theory.

The constructions presented here may have several practical benefits:

- The constructions provide a link with statistical representations of meaning such as latent semantic analysis (Deerwester et al., 1990) and distributional similarity measures (Lee, 1999). Eventually the ideas presented here in combination with such techniques may lead to new methods of automatic ontology construction. For example, by relating semantic analysis vectors to the taxonomy vectors it may be possible to place a new concept in the vector space of the taxonomy based on its latent semantic analysis vector.

- The vector-based representation of a taxonomy can be used to build context theories that make use of the taxonomy whilst remaining entirely vector-based, allowing the use of techniques to combine vectors such as tensor or free products, discussed in the next chapter. These could lead to new, more robust representations of natural language semantics.

- Vector spaces give us a lot of flexibility: vectors can be scaled, rotated, translated, and the dimensionality of the vector space can be reduced. These properties may lead to new techniques for the efficient representation of meaning. For example,

there may be a way to use a dimensionality reduction to efficiently represent a taxonomy in terms of vectors.

Perhaps more importantly though, the subject of this chapter is the nature of meaning itself, and the techniques we present here show that the vector lattice representations of meaning generalise ontological ones. In addition to the lattice structure of ontologies, however, vector lattices allow a more subtle description of meaning that allows the quantification of nearness of meaning that cannot be described fully in the lattice structure of ontologies. Part of the success of techniques such as latent semantic analysis is due to the ability to quantify nearness of meaning in this way; it thus seems only natural that vector lattices will eventually become the structure of choice for representing meaning in natural language.

The contributions of this chapter are as follows:

- We give a definition for a vector lattice completion as a way of representing a taxonomy in terms of a vector lattice.
- We describe several vector lattice completions with different properties:
  - A *probabilistic completion* (see Section 6.1.2) allows the incorporation of the "probability of a concept" into the vector-based description.
  - We describe a *distance preserving completion* (see Section 6.1.3) in which the distance between vectors in the vector lattice representation is the same as the distance in the ontology using a measure of **?**.
  - The vector space representation typically uses a large number of dimensions. In Section 6.1.4 we describe a vector lattice completion that in many cases uses a smaller number of dimensions than the probabilistic completion, and discuss its application to two real world ontologies.
- The constructions we present allow the description of concepts; in Section 6.2 we discuss the representation of words which may be ambiguous, requiring the representations of the individual senses of a word to be combined.

6.1 Taxonomies

Ontologies describe relationships between concepts. They are considered to be of importance in a wide range of areas within artificial intelligence and computational linguistics. For example, WordNet (Fellbaum, 1989) is an ontology that describes relations between word senses.

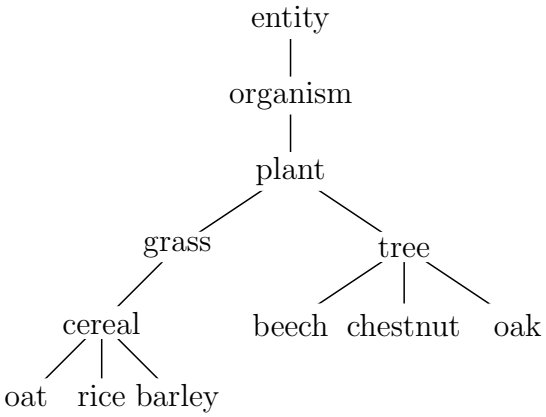


Figure 6.1: A small example taxonomy extracted from WordNet (Fellbaum, 1989).

Arguably the most important relation described in an ontology is the **is-a** relation (also called subsumption), which describes inclusion between classes of objects. When applied to meanings of words, the relation is called *hypernymy*. For example, a *tree* is a type of *plant* (the concept *plant* subsumes *tree*), thus the word "plant" is a hypernym of "tree". The converse relationship between words is called hyponymy, so "tree" is a hyponym of "plant". A system of classification that only deals with the **is-a** relation is referred to as a *taxonomy*. An example taxonomy is shown in figure 6.1, with the most general concept at the top, and the most specific concepts at the bottom.

The **is-a** relation is in general a partial ordering, since

- it is always the case that an a is an a (reflexivity);
- if an a is a b and a b is an a then a and b are the same (anti-symmetry).
- if an a is a b and a b is a c then an a is necessarily a c (transitivity).

One may wish to argue that the second of these is not justified, however, real life taxonomies rarely conflict with this requirement.

The taxonomy described by figure 6.1 has a special property: it is a tree, i.e. no concept subsumed by one concept is subsumed by any other concept. This type of taxonomy will be studied in section 6.1.3.

6.1.1 Vector Lattice Embeddings of Taxonomies

Vector representations of meaning do not seem to sit nicely with ontological representations of meaning — the former make use of vector spaces and the latter make use of lattices. In fact, what we will show in this chapter is that the two types of representation

can be combined within the structure of a vector lattice, a space that is simultaneously a vector space and a lattice. Taxonomies can be embedded in a vector lattice in such a way that the lattice structure is preserved, and existing vector representations of meaning can be considered as implicitly carrying a lattice structure.

The relationship between concepts in a taxonomy is expressed by means of a partial order, and we wish to embed the partial ordering representation in a vector lattice; we call such an embedding a *vector lattice completion*. The partial ordering of the vector lattice representation must still therefore contain the partial ordering of the taxonomy, but in addition, we provide each meaning with a concrete position in some n-dimensional space. We define this formally as follows:

Definition 6.1 (Vector Lattice Completion). Let S be a partially ordered set. A *vector lattice completion* of S is a vector lattice V and a function ϕ from S to V such that $\phi(s_1) \leq \phi(s_2)$ if and only if $s_1 \leq s_2$, for all $s_1, s_2 \in S$.

Because the embedding will necessarily be a *lattice completion*, it will introduce new operations of *meet* and *join* on elements (see section A.3). Many taxonomies may already have some of the properties of a lattice, for example, most taxonomies are *join semilattices*. However the existing join operation is not usually directly useful since it does not correspond with our usual idea of logical disjunction. For example, in figure 6.1, the join of the concepts *beech* and *oak* is *tree*. If we say that something is a *beech* or an *oak*, it is definitely a *tree*, however the converse provides problems: something being a *tree* does not imply that that thing is necessarily a *beech* or an *oak*—since it could also be a *chestnut*. Thus the logical disjunction of the concepts *beech* and *oak* should sit somewhere between these two concepts and *tree*.

6.1.2 Probabilistic Completion

We are also concerned with the *probability* of concepts. This is an idea that has come about through the introduction of "distance measures" on taxonomies. Since words can be ascribed probabilities according to their occurrences in corpora, the concepts they refer to can similarly be assigned probabilities. By the *probability of a concept* we mean the probability of encountering an *instance* of that concept in the ontology, that is, a word whose meaning in that context is subsumed by that particular concept. The following definition clarifies this idea:

Definition 6.2 (Real Valued Taxonomy). A real valued taxonomy is a finite set S of *concepts* with a partial ordering $\leq$ and a positive real function p over S. The *measure* of

a concept is then defined in terms of p as

$$\hat{p}(x) = \sum_{y \in \downarrow(x)} p(y).$$

The taxonomy is called *probabilistic* if $\sum_{x \in S} p(s) = 1$. In this case $\hat{p}$ refers to the *probability of a concept*.

Thus in a probabilistic taxonomy, the function p corresponds to the probability that a term is observed whose meaning corresponds (in that context) to that concept. The function $\hat{p}$ denotes the probability that a term is observed whose meaning in that context is subsumed by the concept.

Note that if S has a top element I then in the probabilistic case, clearly $\hat{p}(I) = 1$. In studies of distance measures on ontologies, the concepts in S often correspond to senses of words, in this case the function p represents the (normalised) probability that a given word will occur with the sense indicated by the concept. The top-most concept often exists, and may be something with the meaning "entity"—intended to include the meaning of all concepts below it.

The most simple completion we consider is into the vector lattice $\mathbb{R}^S$, the real vector space of dimensionality $|S|$, with basis elements $\{e_x : x \in S\}$.

Proposition 6.3 (Ideal Vector Completion). *Let S be a probabilistic taxonomy with probability distribution function p that is non-zero everywhere on S. The function ϕ from S to $\mathbb{R}^S$ defined by*

$$\phi(x) = \sum_{y \in \downarrow(x)} p(y) e_y$$

is a completion of the partial ordering of S under the vector lattice order of $\mathbb{R}^S$, satisfying $\|\phi(x)\|_1 = \hat{p}(x)$.

Proof. The function ϕ is clearly order-preserving: if $x \leq y$ in S then since $\downarrow(x) \subseteq \downarrow(y)$, necessarily $\phi(x) \leq \phi(y)$. Conversely, the only way that $\phi(x) \leq \phi(y)$ can be true is if $\downarrow(x) \subseteq \downarrow(y)$ since p is non-zero everywhere. If this is the case, then $x \leq y$ by the nature of the ideal completion. Thus ϕ is an order-embedding, and since $\mathbb{R}^S$ is a complete lattice, it is also a completion. Finally, note that $\|\phi(x)\|_1 = \sum_{y \in \downarrow(x)} p(y) = \hat{p}(x)$. □

This close connection with the ideal completion is what leads us to call it the *ideal vector completion*. The completion allows us to represent concepts as elements within a vector lattice so that not only the partial ordering of the taxonomy is preserved, but the probability of concepts is also preserved as the size of the vector under the L^1 norm.

6.1.3 Distance Preserving Completion

Some attempts have been made to link ontological representations with statistical techniques. These centre around measures of *semantic distance* which attempt to put a value on semantic relatedness between concepts.

Jiang and Conrath (1998) defined a distance measure that has been shown to perform best out of five different measures in a spelling correction task (Budanitsky and Hirst, 2006). The measure is based on *information content* of concepts, which can be derived from their probabilities. We will show that this measure has the following special property: concepts can be embedded in a vector lattice in such a way that the distance between concepts in the vector lattice is equal to the Jiang-Conrath distance measure.

We are able to show that the distances are preserved in certain types of taxonomy: the concepts must form a *tree*:

Definition 6.4 (Trees). A partially ordered set S is called a *tree* if every element x in S has at most one element $y \in S$ such that $x \prec y$ and there is an element I such that $z \leq I$ for all $z \in S$. The unique element preceding x is called the *parent* of x, it is denoted $\mathrm{Par}(x)$ if it exists.

Note that in a tree only the topmost element I has no parent.

The Jiang-Conrath measure makes use of a particular property of trees. It is easy to see that a tree forms a *semilattice*: for each pair of elements x and y there is an element $x \vee y$ that is the *least common subsumer* of x and y. For example, in figure 6.1, the least common subsumer of *oat* and *barley* is *cereal*; the least common subsumer of *oat* and *beech* is *plant*.

The measure also makes use of the information content of a concept; this is simply the negative logarithm of its probability. In our formulation, the information content $IC(x)$ of a concept x is defined by

$$IC(x) = -\log \hat{p}(x).$$

The information content thus decreases as we move up the taxonomy; if there is a most general element I, it will have an information content of zero.

The Jiang-Conrath distance measure $d(x, y)$ between two concepts x and y is then defined as

$$d(x, y) = IC(x) + IC(y) - 2IC(x \vee y).$$

There is a notable similarity between this expression, and a relation that holds in vector lattices:

$$(*) \qquad |u - v| = u + v - 2(u \wedge v),$$

for all u and v in the vector lattice. This formula provides the starting point for preserving distances in the vector lattice completion.

In its current form, in building a vector lattice we cannot simply replace the function $\hat{p}$ with the information content, since $\hat{p}$ must increase as we move up the taxonomy; instead we must invert the direction of the lattice. This allows us to embed concepts in the lattice while retaining the information content as the norm, and changes joins into meets, so that distances correspond to the Jiang-Conrath measure.

Proposition 6.5 (Distance Preserving Completion). *Let S be a probabilistic taxonomy which forms a tree with partial ordering $\leq$. The function IC defines a positive real-valued function f_{IC} by*

$$f_{IC}(x) = IC(x) - IC(\mathrm{Par}(x)).$$

for $x \in S - \{I\}$, and $f_{IC}(I) = 0$. We define a new partial ordering $\leq'$ on S by $x \leq' y$ iff $y \leq x$ (thus $\leq'$ is the dual *of $\leq$). Then f_{IC} together with the new partial ordering defines a real-valued taxonomy on S. Call the function that maps an element of S to its completion in the new taxonomy ψ. The vector lattice completion of the new taxonomy satisfies $\|\psi(x)\|_1 = IC(x)$ and $\|\psi(x) - \psi(y)\|_1 = d(x, y)$.*

Proof. For the results about vector lattices used here see section A.4. By the tree nature of the taxonomy, f_{IC} is clearly a positive function satisfying $\|\psi(x)\|_1 = IC(x)$. To see the second part, we need to know that the vector lattice $\mathbb{R}^S$ with the L_1 norm is an *AL-space*; this means that $\|s + t\| = \|s\| + \|t\|$ whenever $s \wedge t = 0$. We have here

$$(u - u \wedge v) \wedge (v - u \wedge v) = \tfrac{1}{2}(u + v - 2(u \wedge v) - |u - v|) = 0,$$

where we have used the above identity twice. Thus, using the same identity, we have

$$\begin{aligned} \|u - v\|_1 = \||u - v|\|_1 &= \|u + v - 2(u \wedge v)\|_1 \\ &= \|(u - u \wedge v) + (v - u \wedge v)\|_1 \\ &= \|u - u \wedge v\|_1 + \|v - u \wedge v\|_1 \\ &= \|u\|_1 + \|v\|_1 - 2\|u \wedge v\|_1 \end{aligned}$$

For the last step, we used the fact that we are dealing with positive elements, with $u - u \wedge v \geq 0$ and thus, using the additive property of the L_1 norm, $\|u\| = \|u - u \wedge v + u \wedge v\| = \|u - u \wedge v\| + \|u \wedge v\|$.

Finally, note that the lattice completion is built from the dual of a tree, which is a join semilattice. Joins are preserved as meets in the completion since $\downarrow\!(x) \cap \downarrow\!(y) = \downarrow\!(x \wedge y)$,

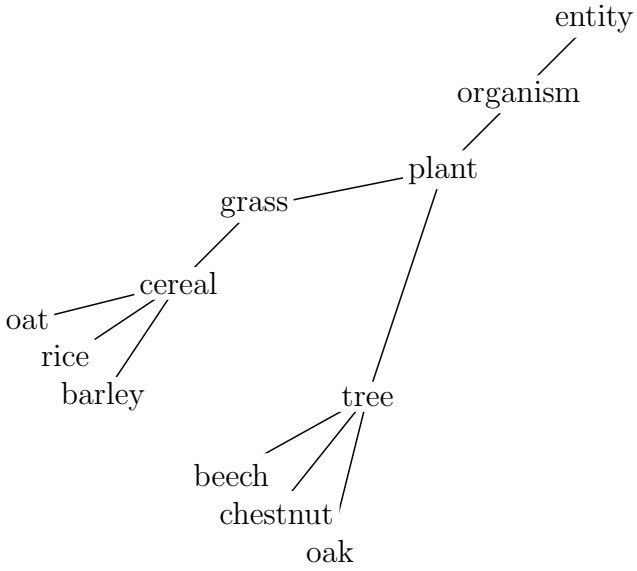


Figure 6.2: It is possible to embed a tree into a two dimensional vector lattice in such a way that the partial ordering is preserved. Two concepts s_1 and s_2 satisfy $s_1 \leq s_2$ if s_1 is to the left or level with and below or level with s_2.

and thus we have $\psi(x) \wedge \psi(y) = \psi(x \vee y)$. This completes the proof. □

Thus we have shown an important link between ontologies and vector lattice representations of meaning: it is possible to simultaneously preserve the partial ordering and distance measure of an ontology within a vector lattice representation. We believe this particular result opens up the potential for a wide range of techniques combining statistical methods of determining meaning with ontological representations. It has been suggested that distributional similarity measures can be used as a predictor of semantic similarity, originating in the distributional hypothesis of Harris (1968). There has not yet to our knowledge been a thorough analysis of the degree to which distributional similarity can be used to predict semantic distance, however our own preliminary investigations reveal that there is definitely some correlation between the two. If this correlation is strong enough then distributional similarity could in theory be used to place concepts in the vector lattice of meanings based on their similarity with surrounding concepts, opening up the field of determining meaning automatically, and making use of the fine-grained structure allowed by the vector lattice representations.

6.1.4 Efficient Completions

In this section we discuss the question of how many dimensions are necessary to maintain the lattice structure in the vector lattice completion. The representations discussed

previously use a very large number of dimensions: one for each node in the ontology. To see that this is more than is generally needed, consider an ontology whose Hasse diagram is planar: that is it can be rearranged so that no lines cross (see figure 6.2). If we then position the nodes in the diagram such the lines between nodes are at an angle of less than 45° to the vertical (this can always be done by stretching the diagram out vertically), then if we rotate the diagram by 45° to the right, and set an origin, the position of each node in the two dimensional diagram can be considered as a representation of the concept in the vector lattice $\mathbb{R}^2$. It is easy to see that the partial ordering is preserved — if $x \leq y$ in the partial ordering, then it will be the case in the two-dimensional vector lattice, although care has to be taken in the positioning of concepts to ensure that other unwanted relations can't be derived in the new space.

One problem with this simplistic vector lattice representation is that there is no obvious way to interpret the two dimensions. Another, more serious problem is that it is not unique: in general there are many ways we can draw the Hasse diagram, and each will correspond to a different representation. Concepts will necessarily be positioned arbitrarily according to which way the diagram is drawn, leaving us in doubt as to whether the vector aspect of the representation is meaningful. This arbitrary positioning means that distances between nodes are dependent on how we draw the Hasse diagram: in one representation a pair of nodes may be close together, while in another they may be far apart. For example, in the Hasse diagram of a tree, we can swap leaf nodes any way we wish to make a pair of nodes arbitrarily close or far apart.

We call representations that don't have this property *symmetric*: a representation is symmetric if the distances $\|x - y\|$ between the representation of a pair of nodes is only dependent on the lattice properties of the nodes represented by x and y. Clearly symmetry goes along with uniqueness: if there is only one representation of a given lattice, the vector properties must be determined by the lattice.

Instead of this two dimensional representation then, we propose an efficient symmetric representation suitable for any partial ordering in which dimensions correspond to *chains* or *totally ordered subsets* of the partial order. This representation is unique up to isomorphism, given a certain requirement on the partial order which we expect to hold for most real world ontologies. This more efficient representation can generally be used in place of the vector ideal completion.

Definition 6.6 (Chains). Let S be a partially ordered set. A *chain* C of S is a totally ordered subset of S, that is a subset of S which is a partially ordered set under the the partial ordering of S such that $x \leq y$ or $y \leq x$ for all $x, y \in C$.

A collection of chains is called *covering* if $\bigcup \mathcal{C} = S$. Clearly every partially ordered set has at least one covering collection of chains: that collection consisting of all chains

containing just one node of S.

Proposition 6.7 (Chain completion). *Let S be a real valued taxonomy and $\mathcal{C} = \{C_1, C_2, \ldots C_n\}$ be a covering collection of chains for S. Let $\mathrm{Ch}_{\mathcal{C}}(x) = \{i : x \in C_i\}$. Then define the function ξ_0 from S to $\mathbb{R}^n$ by*

$$\xi_0(x) = \sum_{i \in \mathrm{Ch}_{\mathcal{C}}(x)} \frac{p(x)}{|\mathrm{Ch}_{\mathcal{C}}(x)|} e_i,$$

where e_i are the basis elements of $\mathbb{R}^n$. Then the chain completion ξ is defined by:

$$\xi(x) = \sum_{y \leq x} \xi_0(y).$$

The function ξ defines a vector lattice completion of S satisfying $\|\xi(x)\|_1 = \hat{p}(x)$.

Proof. By the definition of ξ it is clear that $u \leq v$ in S implies $\xi(u) \leq \xi(v)$ in $\mathbb{R}^n$. Conversely, if it is not true that $u \leq v$ then there will be some chain in $\mathcal{C}$ containing v but not u, so it will never be true that $\xi(u) \leq \xi(v)$, showing that ξ defines an embedding of the partial ordering of S, and since $\mathbb{R}^n$ is a vector lattice, it also defines a vector lattice completion. Finally, note that

$$\|\xi(x)\|_1 = \sum_{y \leq x} \|\xi_0(x)\|_1 = \sum_{y \leq x} p(x) = \hat{p}(x)$$

since all the vectors are positive, which completes the proof. □

Providing we can find a covering with a low number of chains n, the previous proposition gives us an efficient vector lattice representation using n dimensions. The representation as it stands is not unique, since there are in general many ways we can cover a partially ordered set with chains. The task then, is to find an efficient, unique way of determining a covering collection of chains. We achieve this by considering *maximal chains*, chains containing all elements of S that they can whilst remaining a totally ordered set.:

Definition 6.8 (Maximal chains). A maximal chain C for S is a chain such that there is no element x of $S - C$ such that if x were added to C then C would remain a chain. Let $\mathcal{C}$ be the covering collection of chains consisting of all maximal chains of S. S is said to be *uniquely minimally covered* by $\mathcal{C}$ if for each $C \in \mathcal{C}$ there is at least one element $x \in C$ such that x is not in any other chain of $\mathcal{C}$; in this case, S is said to possess a *unique minimal covering* $\mathcal{C}$.

Proposition 6.9. *If S has a unique minimal covering $\mathcal{C}$, then $\mathcal{C}$ is the unique covering for S with the least number of chains.*

Proof. By definition $\mathcal{C}$ is unique, since it consists of all maximal chains. To see that $\mathcal{C}$ has the least possible number of chains, note that each chain contains as many elements of S as possible, while removing any of these chains and maintaining a covering collection is impossible, since each chain contains an element not in any other chain. □

Thus if S has a unique minimal covering, we can represent it uniquely and efficiently using the number of dimensions corresponding to the number of chains in this covering. Any taxonomy that is a tree has a unique minimal covering: each maximal chain will have a leaf node that is not in any chain; in fact there will be a chain corresponding to each leaf node. Thus the chain completion gives a unique efficient representation for any taxonomy that is a tree, and we would expect taxonomies that are very tree-like to also have efficient representations.

6.1.5 Analysis of Application to Ontologies

While we know that the chain completion is a relatively efficient for trees, we don't know how useful it is likely to be in real-world applications. To find out, we analysed two real world ontologies. The first is the Semantic Network used in the Unified Medical Language System (National Library of Medicine, 1998), whose taxonomy consists of just 135 nodes representing broad categories of meanings related to medical concepts. In this case, the taxonomy has a simple tree structure, so each dimension corresponds to a leaf node. There are 90 leaf nodes, thus we can represent the 135 nodes using only 90 dimensions, a saving of a third.

It is also instructive here to consider a simple theoretical situation: a regular tree of depth n with each node having r branches. In this case, the total number of nodes is

$$\sum_{i=1\ldots n} r^i = \frac{r^{n+1}-1}{r-1} - r \simeq \frac{r^{n+1}}{r-1}$$

where the approximation is for large n and $r > 1$. The number of leaf nodes is r^n, thus in this approximation the ratio of leaf nodes to the total number of nodes will be $r^n(r-1)/r^{n+1} = (r-1)/r$. Thus the saving in the chain completion is greatest for low r: in a binary tree, half the nodes will be concentrated in the leaf nodes. The semantic network we considered above has a saving corresponding to $r = 3$.

The second taxonomy we considered was that of WordNet (Fellbaum, 1989). This is a very different situation to that just considered, having a much greater number of nodes, and no tree structure — quite a large number of nodes have more than one parent. We looked at a subset of around 43,000 nodes using the hypernymy relation of nouns only; each node corresponds to a "synset" or concept corresponding to senses of words in WordNet.

We found a covering collection of chains using around 35,000 chains: a saving in terms of dimensionality of around 20%. This does not give a unique representation however, and thus potentially suffers from some of the same problems as the two dimensional representations. The total number of maximal chains was around 60,000, meaning the unique chain-based representation would be less efficient than the straightforward vector lattice completion in which each dimension corresponds to a node.

It seems that chain-based representations are able to provide modest improvements in the efficiency of vector lattice representations, especially in the case of taxonomies with a tree structure. It is our hope, however, that techniques such as dimensionality reduction will eventually provide a means to find much more efficient representations, using good quality approximations which retain as much structure as possible of the original vector lattice.

6.1.6 Context-theoretic Taxonomies

The unifying mathematics of vector lattices allows us to view ontological representations and vector based representations based on the analysis of contexts from the same perspective. Just as we have endowed the partial ordering of taxonomies with a vector structure, we can view the vectors of context-based methods as having a lattice structure. We can view such a structure as a "context-theoretic taxonomy". It satisfies the mathematical requirements of a taxonomy, yet it is not considered as representing concepts that are necessarily related to the real world: it merely represents contexts that terms appear in. Thus the representations discussed in the first chapter — the vectors of latent semantic analysis and the feature vectors of distributional similarity — can be considered as describing context-theoretic taxonomies.

It is this unifying perspective that we hope will lead the way to new methods combining ontological (model-theoretic) and context-theoretic representations and techniques.

6.2 Representing Words

So far we have only really considered representing concepts, or *senses* of words; we have not been concerned yet with how to represent words themselves, which may be ambiguous with meanings covering many senses. For example, we view the structure of WordNet, which describes senses of words, as a partial ordering, or as elements of a vector lattice. If we want to combine the vector lattice representations of the senses of a word to form something representing the ambiguous meaning, what is the correct way to do this?

Context-theoretic techniques provide an answer: if we look at the most straightforward model of context, the representation of a word is given by the vector sum of the

representations of its contexts. This can easily be seen by considering the model of context discussed in the first chapter: if we add sense tags to words in a corpus, then look at the vector representations of the individual senses, since the vector representation is formed linearly, summing these representations will give us the same vector as that arrived at by looking at occurrences of the word without sense tags. This also makes sense from a probabilistic perspective; the probability of the occurrence of a word in a corpus is the sum of the probability of the occurrences of its senses, and this property is carried over in the L^1 norm of the corresponding vector representations. Looking at the lattice structure, this construction behaves as we would expect: each sense of a word entails the word itself. Thus if word w has n senses $s_1, s_2, \ldots s_n \in S$, then the context vector of w would be

$$\hat{w} = \sum_{i=1}^{n} \hat{s}_i$$

where $\hat{s}_i$ is the context vector of sense s_i.

When it comes to making use of vector representations of taxonomies, however, we run into a problem. We have constructed our vectors so that the L^1 norm corresponds to the probability of the *concept*, which depends on the taxonomic structure. According to the context-theoretic philosophy, the representation of a word should be constructed linearly from the representations of its senses, however the probability of the occurrence of a *sense* does not coincide with the probability of a concept. For example, the meaning of the word "entity" corresponds to the most general concept in some taxonomies, and thus the probability of the concept *entity* is 1. However the word itself occurs fairly rarely in corpora, and we would expect it to have a fairly low probability even with respect to words representing much more general concepts.

Looking at the situation from a context-theoretic perspective helps us to find an answer. We can view each node in the taxonomy as a context that words can occur in. In the ideal vector completion a concept s is represented as a sum over basis vectors corresponding to the nodes representing concepts at least as general as s. When s is the sense of a word, we view the word as occurring in sense s in contexts corresponding to the concepts at least as general as s. We may know the probability of the sense, but we have no way to distribute this probability over the hypothetical contexts.

One way of getting around this problem is to renormalise the vectors representing the individual senses s_i and scale them according to the probability π_i that the word w occurs in sense s_i (so that $\sum_i \pi_i$ is equal to the probability of word w occurring):

$$\bar{w} = \sum_{i=1}^{n} \frac{\pi_i}{\|\bar{s}_i\|_1} \bar{s}_i$$

Thus we have a plausible way of representing words in terms of vectors. If we are to make use of these representations as part of a context theory, however, we have to be able to consider them as elements of an algebra. We have already seen the use of projections to represent lattice structures in the previous chapter, and again it is an algebra formed from projections that we will use to represent meanings of words within the setting of a context theory. In fact, as we will show, work in measures of distributional similarity supports the idea of representing meanings as projections.

6.2.1 Distributional Similarity and Projections

The work of Lee (1999) analyses distributional similarity measures with respect to the *support* of the underlying distribution. Let $f_t(c)$ denote the observed frequency of term t occurring in context c. The support $S(t)$ of f_t is the set of contexts c for which $f_t(c)$ is non-zero;

$$S(t) = \{c \in C : f_t(c) > 0\}$$

where C denotes the set of possible contexts that terms may occur in, or the feature space. According to our previous analysis, we consider the function f_t as a vector in the space $\mathbb{R}^C$.

Lee considers measures of the degree of similarity between two terms u and v. She shows that the three best performing measures (which include the L^1 norm, $\|f_u - f_v\|_1$) all depend only on the behaviour of the functions f_u and f_v on the intersection of the supports of the two terms, $S(u, v) = S(u) \cap S(v)$. Those measures which placed emphasis on the behaviour of the functions outside of this set, such as the L^2 norm, generally performed poorly in comparison.

Weeds (2003) takes this analysis further, considering different functions $D(t, c)$ measuring the degree of association between a term t and context c. The support with respect to D is defined as $S_D(t) = \{c \in C : D(t, c) > 0\}$. She then considers the *precision* according to an "additive model" defined in terms of D:

$$\mathcal{P}^{\text{add}}(u, v) = \frac{\sum_{c \in S_D(u,v)} D(u, c)}{\sum_{c \in S_D(u)} D(u, c)};$$

recall can then be defined as the dual of precision, $\mathcal{R}^{\text{add}}(u, v) = \mathcal{P}^{\text{add}}(v, u)$. Weeds goes on to show how a general framework to describe distributional similarity measures can be described in terms of measures of precision and recall, and evaluates a range of measures within her framework. The best performing measure made use of the additive model of precision and recall together with a mutual information based function for D.

The details of Weeds' analysis are not so relevant for us; what is important to note

is that in Weeds' additive model there is a move away from considering terms merely as vectors, and that this move is experimentally successful. What we will show is that we can view the additive model as representing terms as *projections*, special kinds of operators on a vector space.

The vector space we are considering is given by the set C of contexts that terms may occur in; we denote it $\mathbb{R}^C$ since each element c of C has a corresponding basis element $e_c \in \mathbb{R}^C$ which determines a dimension in the vector space. Given a subset of contexts X, $X \subseteq C$, we can view the vector space $\mathbb{R}^X$ as a subspace of $\mathbb{R}^C$. This subspace defines a projection P_X on $\mathbb{R}^C$.

To specify this in more detail, consider a vector f defined on $\mathbb{R}^C$ in terms of its components α_c, where $f = \sum_{c \in C} \alpha_c e_c$. The effect of the projection P_X is then defined as follows:

$$P_X f = \sum_{c \in X} \alpha_c e_c.$$

Given two subsets X and Y of C, it is easy to see that $P_X P_Y = P_{X \cap Y}$, thus the projection encodes set-theoretic behaviour. Since the definitions of precision and recall depend on the intersection of supports, we can translate these definitions into ones based on projections:

$$\mathcal{P}^{\text{add}}(u, v) = \|P_u P_v \Omega_D(u)\|_1,$$

where $P_t = P_{S_D(t)}$ and $\Omega_D(u)$ is a vector in $\mathbb{R}^C$ given in terms of its components by

$$\Omega_D(u) = \frac{1}{\sum_{c \in C} D(u, c)} \sum_{c \in C} D(u, c) e_c.$$

This representation comes close to providing us with a context theory; words can be represented as operators on a vector lattice and thus are elements of an algebra; the difference is that there is not a unique linear functional under consideration, the linear functional (which depends on $\Omega_D(u)$) is different depending on what element we are considering precision with respect to. The preceding analysis does however, point to the representation of meanings as projections on a vector lattice; we will show how such representations allow us to combine representations of concepts to form representations of the meanings of words.

6.2.2 Combining Concept Projections

First we show how concepts in a taxonomy can be represented in terms of projections together with a linear functional.

Definition 6.10 (Ideal Projection Completion). If S is a probabilistic taxonomy with probability distribution function $p \in \mathbb{R}^S$, then the *ideal projection* P_x associated with $x \in S$ is the projection $P_{\downarrow(x)}$ on the space $\mathbb{R}^S$. We define a linear functional ψ on $L^1(\mathbb{R}^S)$ by

$$\psi(A) = \|(Ap)^+\|_1 - \|(Ap)^-\|_1,$$

Proposition 6.11. *The ideal projection completion defines a vector lattice completion for S, such that $\psi(P_x) = \hat{p}(x)$.*

Proof. There is clearly a lattice isomorphism between the ideal completion representation $\downarrow(x)$ of $x \in S$ and the projection P_x; for example

$$P_x P_y = P_{\downarrow(x) \cap \downarrow(y)}.$$

Then note that $\psi(P_x) = \|P_x p\|_1 = \sum_{y \in \downarrow(x)} p(y) = \hat{p}(x)$. □

The new representation encodes probabilities in the linear functional rather than directly in the representation of individual concepts, in contrast to the ideal vector completion introduced earlier. This gives us additional flexibility to combine concept representations in a way which preserves the partial ordering relation as we would expect from a context-theoretic perspective.

The ideal projection completion can in fact be used to define a context theory for an alphabet A if we have a way of associating elements of A with concepts in S; for example A may be a set of words and S a taxonomy of their meanings. If the words are unambiguous they will be associated with just one concept in S. Thus we can associate with each word a projection on $L^1(\mathbb{R}^S)$.

The new flexibility comes in being able to add these projections to create representations of words. If a word w has n senses $s_1, s_2, \ldots s_n \in S$, and the word w occurs in the sense s_i with probability π_i, then we can represent w as a probabilistic sum of the projection representation of its senses:

$$\bar{w} = \sum_{i=1}^{n} \frac{\pi_i}{\psi(P_{s_i})} P_{s_i},$$

where $\bar{w}$ is the representation of w in $L^1(\mathbb{R}^S)$. The factor $\pi_i / \psi(P_{s_i})$ ensures that $\psi(\bar{w})$ is equal to the probability of word w; it can be interpreted as the conditional probability that word w occurs in sense s_i given that some word has occurred in some sense t at least as general as s_i, that is $s_i \leq t$.

Because we represent words as operators, in addition to the usual lattice operations, which work in a similar way to the ideal vector completion, multiplication is also defined

on the representations. We can think of the probabilistic sum of senses as representing our uncertainty about the meaning of a word. The product of two words then, would represent our uncertainty about the conjunction of their meanings. For example, if we approximate the meaning[1] of the word *line* by

$$\bar{w_l} = \tfrac{3}{10}P_{l_1} + \tfrac{1}{10}P_{l_2}$$

where l_1 represents the sense "a formation of people or things one beside another" and l_2 represents the sense "a mark that is long relative to its width", and the word *mark* by

$$\bar{w_m} = \tfrac{1}{5}P_{m_1} + \tfrac{1}{10}P_{m_2}$$

where m_1 represents the sense "grade or score" and m_2 represents the sense "a visible indication made on a surface", then the product is given by

$$\hat{w_l}\hat{w_m} = \tfrac{3}{50}P_{l_1}P_{m_1} + \tfrac{3}{100}P_{l_1}P_{m_2} + \tfrac{1}{50}P_{l_2}P_{m_1} + \tfrac{1}{100}P_{l_2}P_{m_2}.$$

If we further assume that the meanings of senses are disjoint, except for those referring to the sense "a mark that is long relative to its width" and the sense "a visible indication made on a surface"; that is we assume $P_xP_y = 0$ unless $x = l_2$ and $y = m_2$ or vice versa, in which case $P_{l_2}P_{m_2} = P_{l_2}$ since a line is a type of mark. Then $\hat{w_l}\hat{w_m} = \frac{1}{100}P_{l_2}$; the product has disambiguated the meaning of both words.

[1]Meanings are based on Wordnet definitions (Fellbaum, 1989), probabilities are invented.

Chapter 7

Context Theories and Syntax

In this chapter we look at ways of describing syntactic properties of language in terms of vector space operators and algebra. This will allow us to incorporate such properties into context theories for natural language. The ability to view syntax from an algebraic perspective has many potential benefits, for example, we describe a method to represent syntax in terms of matrices that may lead to fast computational methods for statistical parsing, and at the end of the chapter we describe some ideas for how separate context theories for syntax and semantics may be combined using a generalisation of the notion of independence to create a new form of natural language semantics in which both the semantic and syntactic aspects of a word may be represented as a single element of an algebra.

7.1 Background

This chapter places special emphasis on one syntactic formalism; namely that of *link grammars*. In order to see why these grammars are particularly suited to the context-theoretic approach, it is useful to consider other grammars however. The methods for describing syntax in natural language are numerous; we concentrate here on those that we have found to be closest to the algebraic approach, namely variations on categorial grammar (Bar-Hillel, 1950; Lambek, 1958) and later variations on this formalism. We discuss the problems involved in expressing these within the context-theoretic framework, then discuss link grammars, and show two alternative ways of describing these algebraically within the framework.

7.1.1 Bar-Hillel Categorial Grammar

The simplest form of categorial grammar is due to Bar-Hillel (1950; 1964) (based on earlier work of Ajdukiewicz) and is described as a deductive system with the following rewrite

rules:

$$\begin{aligned}(A/B)\ B &\rightarrow A\\ B\ (B\backslash A) &\rightarrow A\end{aligned}$$

In a categorial grammar, words in a language are assigned one or more *categories*, built up out of a number of *basic types* and the operations / and \. For example, a transitive verb might be assigned the category $(NP\backslash S)/NP$, where NP and S are a basic types representing the categories of noun phrases and sentences respectively. The category $(NP\backslash S)/NP$ can be thought of as describing those strings which form a sentence when they are both preceded and followed by a noun phrase.

7.1.2 Lambek Calculus

Based on Bar-Hillel's categorial grammar, Lambek (1958) developed a calculus specifically for describing natural language. In its original form, it is defined as a deductive system, whose axioms[1] are:

$$\begin{aligned}A &\rightarrow A\\ (AB)C &\leftrightarrow A(BC),\end{aligned}$$

where $A \leftrightarrow B$ is shorthand for $A \rightarrow B$ and $B \rightarrow A$, with the following rules of inference:

$$\begin{aligned}AB \rightarrow C &\quad \text{iff} \quad A \rightarrow C/B\\ AB \rightarrow C &\quad \text{iff} \quad B \rightarrow A\backslash C,\end{aligned}$$

and

$$\text{if } A \rightarrow B \text{ and } B \rightarrow C \text{ then } A \rightarrow C$$

Using these rules, it possible to deduce many theorems of the calculus, for example

$$\begin{aligned}(A/B)\ B &\rightarrow A && \text{(Ajdukiewicz's law)}\\ A &\rightarrow (B/A)\backslash B && \text{(Type raising)}\\ (A/B)(B/C) &\rightarrow A/C && \text{(Composition)}\end{aligned}$$

and their equivalents with / exchanged with \; many of these are useful in describing features of natural language.

[1] See also Wood (1993).

One way of modeling the Lambek calculus is with free semigroups (also called *L-models*) — the completeness of the Lambek calculus with respect to such models is described in Pentus (1995). The calculus can be viewed as operations on subsets of a monoid M, with

$$\begin{aligned} XY &= \{xy : a \in X, b \in Y\} \\ X\backslash Y &= \{m \in M : Xm \subseteq Y\} \\ Y/X &= \{m \in M : mX \subseteq Y\} \end{aligned}$$

where $X, Y \subseteq M$ and we also use m as a shorthand for $\{m\}$.

More generally, the operations / and \ can be defined for certain semigroups called *residuated lattices* (Birkhoff, 1973). The connection between the Lambek calculus and residuated lattices was noted in Lambek's original paper (Lambek, 1958).

Definition 7.1 (Partially Ordered Semigroup). A semigroup S together with a partial ordering $\leq$ is called *partially ordered* if $x \leq y$ implies $xz \leq yz$ for all $x, y, z \in S$.

Definition 7.2 (Lattice Ordered Semigroup). A lattice ordered semigroup is a partially ordered semigroup S in which the partial ordering defines a lattice with operations $\vee$ and $\wedge$ such that

$$\begin{aligned} x \cdot (y \vee z) &= x \cdot y \ \vee \ x \cdot z \\ (y \vee z) \cdot x &= y \cdot x \ \vee \ z \cdot x \end{aligned}$$

Definition 7.3 (Residuated Lattice). A lattice ordered semigroup S is called a residuated lattice, if for each $x, y \in S$ there exists a greatest element x/y such that

$$x/y \cdot y \leq x$$

and a greatest element $x\backslash y$ such that

$$y \cdot y\backslash x \leq x.$$

The elements x/y and $y\backslash x$ are called the right and left *residuals* or *quotients*.

As Birkhoff (1973) notes, if S has a zero which is also the least element of the lattice then the residuation operations / and \ can be defined by

$$\begin{aligned} x/y &= \bigvee\{z : zy \leq x\} \\ y\backslash x &= \bigvee\{z : yz \leq x\} \end{aligned}$$

The notion of residuated lattice is useful for our purposes because it allows us to think of categorial grammar in purely algebraic terms, allowing us to see how it relates to the context theoretic framework, and how it compares to other algebraic approaches.

7.1.3 Bilinear Logic

Lambek (1993) and Abrusci (1991), based on earlier work of Girard (1987), developed a new version of Lambek's calculus called *(classical) bilinear logic*. This adds two constants, 1 (introduced at an earlier stage by Lambek) and 0, to Lambek's original definition, which satisfy

$$1A \leftrightarrow A \leftrightarrow A1$$
$$(0/A)\backslash 0 \leftrightarrow A \leftrightarrow 0/(A\backslash 0)$$

As a shorthand notation, $A\backslash 0$ is written A^r and $0/A$ is written A^l. It can be shown that

$$(B^r A^r)^l \leftrightarrow (B^l A^l)^r$$

which is written as $(A \oplus B)$. Some theorems of bilinear logic (Casadio and Lambek, 2002) are

$$1^r \leftrightarrow 0 \leftrightarrow 1^l$$
$$A \oplus 0 \leftrightarrow A \leftrightarrow 0 \oplus A$$
$$(A \oplus B) \oplus C \leftrightarrow A \oplus (B \oplus C)$$
$$A^l A \to 0 \qquad AA^r \to 0$$
$$1 \to A \oplus A^l \qquad 1 \to A^r \oplus A$$
$$A/B \leftrightarrow A \oplus B^l \qquad B\backslash A \leftrightarrow B^r \oplus A$$
$$(A \oplus B)C \to A \oplus BC \qquad C(A \oplus B) \to CA \oplus B$$

7.1.4 Pregroups

Pregroups (Lambek, 2001) arose as a simplification of bilinear logic called *compact bilinear logic*, in which it is additionally assumed that $0 \leftrightarrow 1$ and $AB \leftrightarrow A \oplus B$. In this case there is a simpler description in terms of partially ordered monoids:

Definition 7.4 (Pregroup). Let S be a partially ordered monoid. Then S is called a

pregroup if for each $x \in S$ there are elements x^l and x^r in S such that

$$\begin{aligned} x^l x &\leq 1 \leq x x^l \\ x x^r &\leq 1 \leq x^r x \end{aligned}$$

7.1.5 Categorial Grammar and Context Theories

We would like to be able to describe the syntactic formalisms we have discussed within the context-theoretic framework; firstly to demonstrate the generality of the framework, and secondly, because we hope new techniques in parsing and semantic representation to arise by doing so. When it comes to categorial grammars, we seem to be well-placed since there are algebraic interpretations of many versions of the formalism. However, on closer inspection, making direct use of these formalisms within the context theoretical framework appears difficult.

For example, if we want to make use of a residuated lattice S, we could try and represent the structure within a lattice ordered algebra. Like any semigroup, the vector space $L^1(S)$ can be considered as a lattice ordered algebra (see Section A.5). However, the lattice ordering of $L^1(S)$ is not connected to the lattice ordering of S. If we wished to connect them, we may try to use one of the constructions described in the previous chapter to embed partial orderings within vector lattices. However, then it is not clear how we are to define multiplication on the vector lattice in a way that is consistent with multiplication in S.

We face similar problems with pregroups: it is not clear how we can incorporate the pregroup partial order into a vector lattice partial order whilst maintaining the multiplication defined in the pregroup.

Bilinear logic appears closer to being a vector space with an "addition" operation, $\oplus$, however, this operation is not defined to be commutative, something which is essential for a vector space. Requiring $\oplus$ to be commutative results in multiplication also being commutative, something not generally desirable for describing natural language syntax.

There is one way to represent categorial grammars within the framework however: we can make use of free semigroup models to describe the Lambek calculus. Instead of using subsets of a free monoid A^*, however, we use elements of the algebra $L^\infty(A^*)$. A set $X \subset A^*$ is represented as the element $\tilde{X} \in L^\infty(A^*)$:

$$\tilde{X}(z) = \begin{cases} 1 & \text{if } z \in X \\ 0 & \text{otherwise,} \end{cases}$$

for $z \in A^*$. Multiplication in this algebra is defined by multiplication of the underlying free monoid, while vector space and lattice operations are defined since $L^\infty(A^*)$ is a vector lattice. We are thus able to represent the syntactic properties of a word by taking weighted sums of the representation of its syntactic categories, with weights corresponding to the probability that a word will take the respective category.

We can use this idea to make a context theory if we define a linear functional ϕ on $L^\infty(A^*)$ by

$$\phi(u) = \sum_{x \in A^*} p(x)u(x)$$

where p is a probability distribution over elements of A^*. In this way, the context-theoretic probability of a category is the sum of the probabilities of all the strings in that category.

This representation raises computational issues similar to the ones that arose in dealing with logical semantics in Section 5.3; and a similar solution can be used. The problem again is that a word may be represented as a sum of categories whose vector representations are not disjoint in the vector lattice. The same method for computing a lower bound for the degree of entailment between sentences can be used to estimate a degree of entailment between a desired parse and the syntactic representation of a sentence, or to estimate a syntactic "entailment" between sentences.

Note that the algebra $L^1(A^*)$ is *not* a residuated lattice under the vector lattice ordering, since it is not a lattice ordered semigroup under this ordering. The subsemigroup of elements of this algebra generated by the representation of categories does, however, form a lattice ordered semigroup under this ordering, and is also a residuated lattice, since it is isomorphic as a lattice ordered semigroup to the semigroup of subsets of the free monoid A^*. This means, that while we can represent categories within the algebra and take weighted sums of them, we cannot form new categories from these weighted sums — something that is not a limitation for representing natural language syntax.

7.1.6 Link Grammar

Link grammar (Sleator and Temperley, 1991) is a lexicalised syntactic formalism which describes properties of words in terms of *links* formed between them, and which is context-free in terms of its generative power. Apart from determining which sequences are grammatical, the links also encapsulate the nature of the relationships between words.

As an example, a transitive verb in English may link (simultaneously) to a subject on the left and an object on the right. This is represented in link grammar as the *disjunct* $|s\rangle\langle o|$ where s and o stand for 'subject' and 'object' respectively.[2]

[2]We are introducing our own, quantum mechanical, notation for link grammars from the beginning so as to be consistent, however we will describe the intended interpretation of this notation later.

Table 7.1: A small link grammar.

word	*disjuncts*			
they	$\langle s\vert$			
mashed	$\vert s\rangle\langle o\vert$	$\vert s\rangle\langle m\vert\langle o\vert$		
way, mud	$\vert d\rangle\vert o\rangle$	$\vert d\rangle\vert j\rangle$	$\vert a\rangle\vert d\rangle\vert o\rangle$	$\vert a\rangle\vert d\rangle\vert j\rangle$
their, the	$\langle d\vert$			
through	$\vert m\rangle\langle j\vert$			
thick	$\langle a\vert$			
Link types:	s: subject, o: object, m: modifying phrases, a: adjective, j: preposition.			

Definition 7.5 (Link Grammar). Let L be a set of *link types*. Then we define a set of *left connectors* $D_l(L) = \{\vert x\rangle : x \in L\}$ and a set of *right connectors* $D_r(L) = \{\langle x\vert : x \in L\}$.

A disjunct is an element of $D_l(L)^* D_r(L)^*$. That is, a disjunct consists of a string of left connectors $\vert x_1\rangle\vert x_2\rangle \ldots \vert x_n\rangle$ followed by a string of right connectors $\langle y_1\vert\langle y_2\vert \ldots \langle y_m\vert$.

The syntactic representation of a word is a set of disjuncts, each one corresponding to a different syntactic rôle played by the word. A sequence of words is in the language generated by the grammar if there is a corresponding sequence of disjuncts and a set of arcs, or *links* drawn above the disjuncts such that:

- each disjunct in the sequence is a disjunct of the corresponding word in the sequence of words;
- each left connector is connected to a right connector of the same type at any position to the right of it by drawing a link from one to the other;
- each connector in each disjunct in the sequence is connected to exactly one other connector;
- no links cross.

Table 7.3 shows a fragment of a link grammar. The grammar is clearly highly simplified, and is presented merely to explain the concept; for example in our fragment, *way* and *mud* can only occur as objects. Link grammars generally include a special symbol called the 'wall' to indicate the beginning of the sequence (Sleator and Temperley, 1991), which is then included in the grammar, but again we have omitted this for simplicity.

A parse for a sentence is drawn as a set of links above the sentence, as in Figure 7.1 for the sentence 'they mashed their way through the thick mud'. The disjuncts that are

used in the parse are not generally drawn, but can be inferred from the links drawn above the sentence.

An efficient parsing algorithm for link grammar based on dynamic programming is described by Sleator and Temperley (1991). Their link grammar for English can handle transitive, ditransitive and modal verbs; prepositions, adverbs, complex noun phrases and relative clauses; questions and question inversion; number agreement is also taken into account.

7.2 Operator Formulation of Link Grammar

In this section we describe link grammar in terms of operators on a vector space. The mathematics we will make use of is in fact derived from that of quantum mechanics: links are described as combinations of 'creation' and 'annihilation' operators referring to the creation and annihilation of a particle in a quantum mechanical system.

The mathematics of quantum mechanics has proved useful in retrieval applications for removing unwanted components of meaning in a search query (Widdows, 2003) on latent semantic analysis vectors. Quantum mechanics deals with a kind of vector space that is particularly well behaved and frequently occurring, so called *Hilbert space*. We make use of a special kind of infinite dimensional Hilbert space called *Fock space*. As we will show, we can describe syntactic properties of words in terms of link grammars as operators on such a space.

One immediate benefit of this discovery is an entirely new perspective on link grammars, which may open up research on this type of grammar. For example, we will show how this view of link grammars can be used to describe the grammar in terms of matrix operations, opening up the possibility of (potentially very efficient) computational procedures for statistical parsing using matrices.

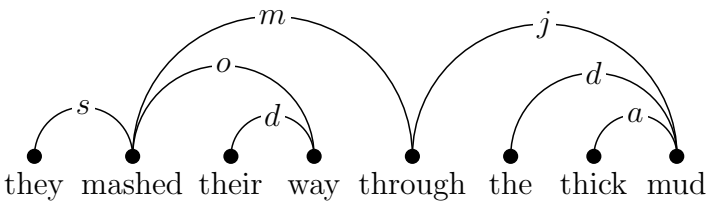


Figure 7.1: A link grammar parse.

7.2.1 Quantum Mechanics and Syntax

We now begin our algebraic formulation of link grammar in terms of operators on Hilbert space. The required definitions are given in the appendix (section A.2); a more complete description, is given for example by Kreyszig (1989) which also applies the mathematics to quantum mechanics.

Our exposition is inspired by the study of *free probability* (Voiculescu, 1997), wherein the study of non-crossing diagrams is very closely connected to link grammars; our main result in this section is more or less a direct translation of a standard result in free probability theory.

Our syntactic vectors will reside in *Fock Space*, a Hilbert space which is like the sum of an infinite series of Hilbert spaces.

Let H be a finite dimensional complex Hilbert space and Ω a distinguished vector in H with norm 1. The Fock space $\mathcal{F}$ of H is then defined as

$$\mathcal{F} = \mathbb{C}\Omega \oplus H \oplus (H \otimes H) \oplus (H \otimes H \otimes H) \oplus \cdots$$

i.e. it is the direct sum of all finite tensor product powers of H, where $\oplus$ denotes the direct sum and $\otimes$ the tensor product (see section A.2.5), and $\mathbb{C}\Omega$ is a one dimensional Hilbert space which is viewed as the zeroth power of H.

We are now able to form the connection between quantum mechanics and syntax. In the physical interpretation of Fock space, different powers of the Hilbert space H correspond to states of different numbers of particles. Special operators called *creation operators* map states in n powers of H to states in $n+1$ powers of H, effectively 'creating' an additional particle. Similarly, *annihilation operators* reduce the number of powers of H in a state by one, 'annihilating' a particle. It is these operators that we will use to represent syntax.

Let u be a vector in H. The creation operator $|u\rangle$ on $\mathcal{F}$ is defined such that

$$|u\rangle v_1 \otimes v_2 \otimes \cdots \otimes v_n = u \otimes v_1 \otimes v_2 \otimes \cdots \otimes v_n.$$

The *dual* of $|u\rangle$ is the annihilation operator $\langle u|$ and maps vectors according to:

$$\langle u|v_1 \otimes v_2 \otimes \cdots \otimes v_n = \langle u, v_1\rangle v_2 \otimes \cdots \otimes v_n$$

and $\langle u|\Omega = 0$. The action of the operators on sums of tensor products can be deduced from their linearity.

The effect of 'creating' and then 'annihilating' is just a scalar product times the identity

operator, 1:

$$\langle u|v\rangle = \langle u, v\rangle 1;$$

the notation $\langle u|v\rangle$ is used whenever a creation operator follows an annihilation operator.

7.2.2 Syntactic Interpretation

In the syntactic interpretation of Fock space, the set of links L are represented as a set of vectors L_H which are assumed to form an *orthonormal basis* for H. Disjuncts for words are then formed by concatenating creation and annihilation operators, in exactly the same way that left and right connectors are concatenated in link grammar. The representation of the syntactic characteristics of a word can then be represented by taking the sum of its disjuncts. For example the word *mashed* in our simple link grammar in Table 7.3 can be represented as the operator

$$\hat{mashed} = |s\rangle\langle o| + |s\rangle\langle m|\langle o|,$$

where we assume the vectors $s, o, m, a, j \in H$ form an orthonormal basis for H.

Our formulation will require that the link grammar parses are "strict" in the following sense: there must not be any connectors left unlinked; thus the parse must start with a right connector and end with a left connector.

In order to determine whether a sequence of words is in the language determined by the link grammar, we define a linear functional ϕ on $B(\mathcal{F})$ (the set of bounded linear operators on $\mathcal{F}$) by

$$\phi(\hat{a}) = \langle \Omega, \hat{a}\Omega\rangle,$$

where $\hat{a} \in B(\mathcal{F})$. We then have the following:

Proposition 7.6. *Let W be a set of words, and Γ a function that assigns a set of link grammar disjuncts to every word in W, with link types from a set L.*

For every $w \in W$ we denote its corresponding Fock space operator $\hat{w}$ on the Fock space generated by the Hilbert space with basis vectors L_H corresponding to the link types in L. Then $w_1 w_2 \dots w_n$ is in the link grammar language defined by Γ if and only if $\phi(\hat{s}) \geq 1$, where $s = \hat{w}_1 \hat{w}_2 \dots \hat{w}_n$. $\phi(\hat{s})$ indicates the number of valid link grammar parses.

Proof. Let us first assume each word has only one disjunct.

The product of an annihilation operator with a creation operator satisfies

$$\langle x|y\rangle = \begin{cases} 0 & \text{if } x \neq y \\ 1 & \text{if } x = y \end{cases},$$

where $x, y \in L_H$. Thus any operator $\hat{s}$ which is given by a product of creation and annihilation operators reduces either to 0, 1, or a product of a (possibly empty) sequence of creation operators followed by a (possibly empty) sequence of annihilation operators. In the latter case, as in the case of 0, $\phi(\hat{s})$ will be zero since if there are annihilation operators in the sequence their operation on Ω will give zero (they operate on Ω first as they are on the right), and if there are no annihilation operators the creation operators will operate on Ω to give a vector disjoint with Ω.

If the sequence satisfies any of the following the product will be zero and the sentence will not parse:

- A left connector is not matched by a right connector; in this case the product of the corresponding operator will map Ω to a different dimension in the Fock space and $\phi(\hat{s})$ will be zero.
- The left connector is matched by a right connector of a different type; in this case the product of the corresponding operators will be zero since
- The connectors match but the corresponding links cross; in the case there will again be a product of the form $\langle x|y\rangle$ where $x \neq y$ and the product will be zero.

Conversely, $\phi(\hat{s})$ will be zero just in case one of the above conditions holds and thus the sentence will not parse.

On the other hand, if none of the above conditions are met the sentence must parse and if the parse is strict the corresponding operator must map Ω to itself, so $\phi(\hat{s}) = 1$.

If words are now allowed more than one disjunct, then since these are added as operators and distribute with respect to multiplication each possible parse will be a term in the resulting sum of disjunts, and thus $\phi(\hat{s})$ will indicate the number of valid link grammar parses. □

Note that this representation defines a strong context theory: the original Hilbert space H is a vector lattice under the ordering induced by the basis associated with the set of link types, and thus $\mathcal{F}$ is also a vector lattice since we can define a basis for it using the basis of H. Thus the space of operators on this space also form a vector lattice, as well as an algebra; specifically we are interested in the algebra $\mathcal{A}$ generated by creation and annihilation operators. Together with the linear functional ϕ and the translation from strings to operators, where we assume that the empty string translates to the identity operator, we have a context theory. Moreover, the subspace $I = \{u \in \mathcal{A} : \phi(u) = 0\}$ is a sub-vector lattice of $\mathcal{A}$ since it is the space formed from all linear combinations of sequences of creation and annihilation operators which do not map Ω onto itself, thus we have a strong context theory.

7.2.3 Stochastic Link Grammar

In applications requiring robust parsing of natural language stochastic grammars are vital in order to help in dealing with the large number of parses, which in general for wide coverage parsers increases exponentially with sentence length (Manning and Schütze, 1999).

In the case of our implementation of link grammar we are not restricted to using sums of the basis vectors L_H, but can take any linear combination of these vectors when constructing the grammar, enabling us to form a type of stochastic link grammar. The representation of a word would be a weighted sum of the representation of its disjuncts; the weight attached to each disjunct can be interpreted as the probability that the word occurs in that syntactic rôle. For products of words, the weights attached to disjuncts will in general sum to less than 1 since; it is thus necessary to renormalise the weights after taking the product to account for disjuncts whose product is zero in order to interpret them as probabilities.

Probabilistic link grammars were described by Lafferty et al. (1992), where the probability of each link occurring with a word is conditioned on several factors, including the words occurring on either side. Such a model provides a probability distribution over the language generated by the grammar. They showed their formalism to be a generalisation of *trigrams* which have proved very successful in language modelling. Our formalism does not allow conditioning of the probability directly, as Lafferty et al's does, however this information can be incorporated by including extra links describing the features one wishes to condition the probability on, and weighting these links accordingly.

An advantage of this simpler formulation of stochastic link grammar in comparison to that of Lafferty et al. (1992) is that it allows an entirely lexicalised description of syntax: the grammar can be described by assigning each word its disjuncts and corresponding probabilities. The ultimate advantage however, we believe, will be in opening up new computational procedures for statistical parsing using matrices.

7.2.4 Link Grammar and Matrices

The operators described in the previous section operate on an infinite-dimensional vector space — something that is clearly difficult to implement. In practice, it may be possible to consider a finite-dimensional subspace of this vector space. This can be done by placing a limit on the number of left or right links that can be concatenated together. For example, we could use the subspace

$$\mathcal{F}_3 = \mathbb{C}\Omega \oplus H \oplus (H \otimes H) \oplus (H \otimes H \otimes H)$$

of the Fock space which is made up of $1 + n + n^2 + n^3$ dimensions, where n is the number of dimensions of H. This would allow up to three left links and up to three right links to be concatenated. In general, allowing the concatenation of k links would need $\sum_{i=0}^{k} n^i = \frac{n^{k+1}-1}{n-1}$ dimensions, thus the number of dimensions required increases exponentially with the number of concatenations required. An important question for this method of representation is what the maximum number of concatenations is likely to be for a particular grammar and application; if this number is high the technique may become impractical.

The matrix representation of a link grammar can be built up using the standard definitions of tensor product and direct sum for matrices. For example, for a two dimensional vector space with basis vectors a and b, for $k = 2$ we can assign the seven dimensions the following interpretations:

$$[\Omega, a, b, a \otimes a, a \otimes b, b \otimes a, b \otimes b]$$

The creation operator (left link) $|a\rangle$ would then have the matrix representation

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

since it maps Ω to a, a to $a \otimes a$ and b to $a \otimes b$. The corresponding annihilation operator $\langle a|$ is represented by the matrix transpose of the representation of $|a\rangle$.

One way to get around the problem of exponential increase in the number of required dimensions is to make use of a dimensionality reduction, such as that of random projections (Papadimitriou et al., 1998; Sahlgren and Karlgren, 2002). In this technique, each basis vector in the original vector space is represented as a random vector in a new vector space of much lower dimensionality; this defines a transformation (a random projection) from the old vector space to the new. If the dimensionality of the new vector space is sufficiently high, it is highly likely that distances and scalar products between vectors will be preserved to within some threshold, however some further work is required to investigate the suitability of this technique for representing syntax.

7.2.5 Parsing with Operators

So far we have only really treated the problem of acceptance of a context-free language with operators: we can tell if a sentence is in the language, but we are left with no record of the parse itself. This is not very useful in applications, since we are normally interested in finding out the structure of the sentence. In order to determine this structure as we multiply the operator representations, we need to be able to keep a record of which disjunct was used with each word. This can be done by defining a new vector space H_d of dimensionality d, where d is the greatest number of disjuncts that any word has in the grammar. We then form the Fock space $\mathcal{F}_d$ of this vector space and take the tensor product with the original Fock space in which the link grammar is represented. We now alter our original operators so that they operate on the new space $\mathcal{F} \otimes \mathcal{F}_d$. If a word has the original representation $x_1 + x_2 + \ldots x_d$ where the x_i are the representations of the individual disjuncts, then in the new representation it becomes

$$x_1 \otimes |e_1\rangle + x_2 \otimes |e_2\rangle + \ldots x_d \otimes |e_d\rangle,$$

where the e_i are basis vectors for H_d.

As these representations are multiplied, the product will be a sum of disjuncts; the right hand side of each disjunct will be a product of creation operators, each specifying the number of the disjunct used in the corresponding word. Those disjuncts of a word which cannot be used to form sentences will have a product of zero, and thus will not feature in the sum; nor will their tensor product with $\mathcal{F}_d$, thus only those disjuncts that can be used to form valid sentences will be represented in the product.

7.3 Algebraic Formulation of Link Grammars

The vector space formulation of link grammar we have just described is useful because it gives us insight into the relationship between syntax and vector space operators, and may also lead to new computational techniques. However we are interested in combining representations of syntax with representations of meaning, and the formulation just described does not seem to be ideally suited to this. Describing words as operators on Fock space would allow to be built up using tensor products only in a limited fashion: Fock space vectors work like a stack, and vectors can only be "pushed" or "popped" on this stack.

If we can describe syntax in algebraic terms, specifically in terms of semigroups, then we will be on much stronger ground because of the tools available for combining such representations. In particular, free inverse semigroups allow the representation of trees in

algebraic terms. As we will see, we will not lose the flexibility of vector space representations; the vector space nature will be regained by considering the algebra $L^1(S)$ that can be associated with each semigroup S.

First we will describe a semigroup to represent link grammar in terms of strings of left and right connectors.

Definition 7.7 (Bracket Semigroup). We define

$$D(L) = D_l(L) \cup D_r(L) \cup \{0\},$$

and let $\equiv$ be the minimal congruence on $D(L)^*$ satisfying

$$\langle x|y\rangle \equiv \begin{cases} 0 & \text{if } x \neq y \\ 1 & \text{if } x = y \end{cases},$$

for all $x, y \in L$ and $0x \equiv x0 \equiv 0$ for all $x \in D(L)^*$, where 1 is the empty string. Then the *bracket semigroup on* L is defined as $D(L)^*/\equiv$. We identify the equivalence classes of the bracket semigroup by their shortest elements.

Note that the identities that form the congruence are similar to those satisfied by the creation and annihilation operators; in fact, the bracket semigroup is not more than an algebraic description of these operators. By combining this representation with the one we are about to describe we will have a description of syntax that combines the best of both the representations.

7.3.1 Inverse Semigroups

The bracket semigroup defined previously falls within a more general category of semigroups: that of inverse semigroups.

Definition 7.8 (Inverse Semigroup). An inverse semigroup S is a semigroup such that each element $x \in S$ has a unique element $x^{-1} \in S$ such that $xx^{-1}x = x$ and $x^{-1}xx^{-1} = x^{-1}$.

Proposition 7.9. *A bracket semigroup is an inverse semigroup.*

Proof. Define $\langle x|^{-1} = |x\rangle$ and $|x\rangle^{-1} = \langle x|$. Let $x_1x_2\ldots x_n$ be a representative element of an equivalence class of a bracket algebra, then define

$$(x_1x_2\ldots x_n)^{-1} = x_n^{-1}x_{n-1}^{-1}\ldots x_1^{-1}.$$

Then the operation as given defines a unique inverse satisfying the requirements of an inverse semigroup. □

The identification of link grammars as a type of inverse semigroup has led us to consider other kinds of inverse semigroup as a possible means of incorporating semantics into the formalism. We recount some basic properties of inverse semigroups (Howie, 1976).

Let S be an inverse semigroup with set of idempotents $E(S)$. Then:

- $(a^{-1})^{-1} = a$ for all $a \in S$.
- $aa^{-1} \in E(S)$ for all $a \in S$.
- $aea^{-1} \in E(S)$ for all $a \in S$, $e \in E(S)$.
- $e^{-1} = e$ for all $e \in E(S)$.
- $ef = fe$ for all $e, f \in E(S)$, i.e. idempotents commute, and thus form a subsemigroup of S.
- A partial order $\leq$ can be defined on S by $a \leq b$ if there exists $e \in E(S)$ such that $a = eb$. If $a \leq b$ then:
  - $aa^{-1} = ba^{-1}$
  - $a = ab^{-1}a$
  - There exists $e \in E(S)$ such that $a = be$
- The partial order is easily seen to be a generalisation of the semilattice order on a commutative semigroup of idempotents, defined by $e \leq f$ if $ef = e$, and $e \wedge f = ef$.

7.3.2 Free Inverse Semigroups

The bracket semigroup does not store the 'parse' of a sentence, it merely informs us whether a sentence parses or not. An alternative construction that is of great importance for our studies is the notion of a *free* inverse semigroup. As we will see, we can use this structure to represent syntax; as we will see, a link grammar parse of a sentence corresponds to an idempotent in a corresponding free inverse semigroup. In this representation, the parse can be deduced from the idempotent itself; the semigroup effectively stores information about the parse of the sentence.

The crucial work on free inverse semigroups was done by Munn (1974) in which he proves that free inverse semigroups are isomorphic to *birooted word-trees*, also called Munn trees.

Informally, the free inverse semigroup on a set A is formed from elements of A and their inverses, $A^{-1} = \{a^{-1} : a \in A\}$, satisfying no other condition than those of an inverse semigroup. Formally, the free inverse semigroup is defined in terms of a congruence relation on $(A \cup A^{-1})^*$ specifying the inverse property and commutativity of idempotents — see Munn (1974) for details. We denote the free inverse semigroup on A by FIS(A).

7.3.3 Equivalence to Birooted Word-Trees

A birooted word-tree on a set A is a directed acyclic graph whose edges are labelled by elements of A which does not contain any subgraphs of the form $\bullet \xrightarrow{a} \bullet \xleftarrow{a} \bullet$ or $\bullet \xleftarrow{a} \bullet \xrightarrow{a} \bullet$, together with two distinguished nodes, called the start node, $\square$ and finish node, $\circ$.

A element in the free semigroup FIS(A) is denoted as a sequence $x_1^{d_1} x_2^{d_2} \dots x_n^{d_n}$ where $x_i \in A$ and $d_i \in \{1, -1\}$.

We construct the birooted word tree by starting with a single node as the start node, and for each i from 1 to n:

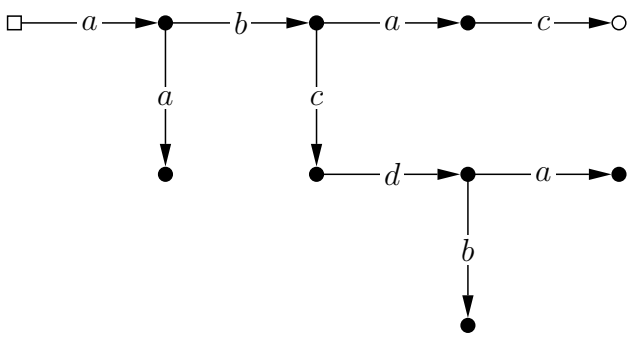
- Determine if there is an edge labelled x_i leaving the current node if $d_i = 1$, or arriving at the current node if $d_i = -1$.
- If so, follow this edge and make the resulting node the current node.
- If not, create a new node and join it with an edge labelled x_i in the appropriate direction, and make this node the current node.

The finish node is the current node after the n iterations.

As an example consider the set $A = \{a, b, c, d\}$, and the element in FIS(A) given by the sequence

$$aaa^{-1}bcdbb^{-1}aa^{-1}d^{-1}c^{-1}ac.$$

This has the following graph:



The product of two elements x and y in the free inverse semigroup can be computed by finding the birooted word-tree of x and that of y, joining the graphs by equating the start node of y with the finish node of x (and making it a normal node), and merging any other nodes and edges necessary to remove any subgraphs of the form $\bullet \xrightarrow{a} \bullet \xleftarrow{a} \bullet$ or $\bullet \xleftarrow{a} \bullet \xrightarrow{a} \bullet$.

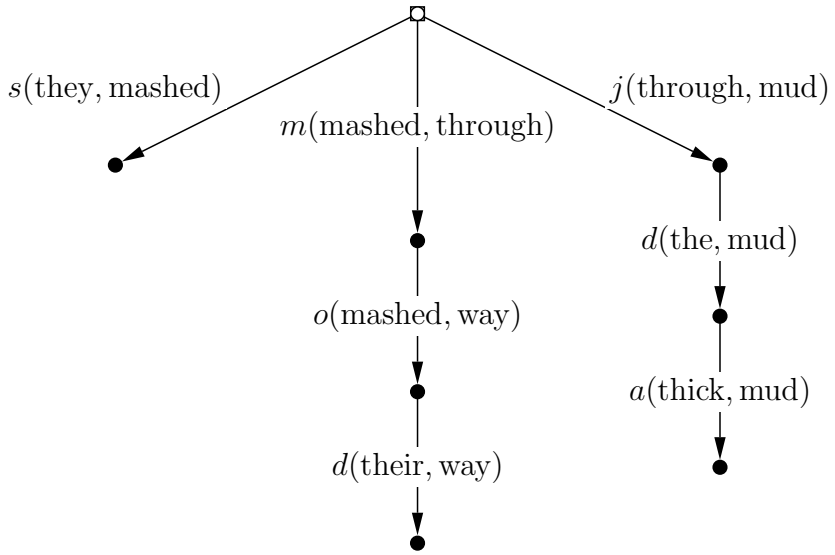
The inverse of an element has the same graph with start and finish nodes exchanged.

7.3.4 Syntactic Equivalence

We can represent parses of sentences in link grammar by translating words to syntactic categories in the *free inverse semigroup* instead of the bracket algebra. In this case sentences are represented as idempotents. For example, the parse shown earlier for "they mashed their way through the thick mud" can be represented in the inverse semigroup on $A = \{s, m, o, d, j, a\}$ as

$$ss^{-1}moo^{-1}dd^{-1}m^{-1}jdaa^{-1}d^{-1}j^{-1}$$

which has the following birooted word-tree:



In this graph, the fact that start and finish nodes overlap indicates that the element is idempotent. The nodes linked by the grammar are indicated in brackets; later we will be able to attach the meanings of these words to the links in the grammar.

We formalise the equivalence with the following proposition:

Proposition 7.10. *Let S be the free inverse semigroup on the set of link types. The inverse semigroup representation of a disjunct is the element of S formed by replacing*

each left and right link of type a with elements $a \in S$ and $a^{-1} \in S$ respectively. Then if a sequence of disjuncts is a link grammar parse the product of the inverse semigroup representation of the disjuncts is idempotent.

Proof. Let x be the concatenation of disjuncts (we can also interpret x as an element of S), and let a be the first (leftmost) element of the sequence x. If the sequence of disjuncts is a link grammar parse then for each left connector there is a corresponding right connector on its right, and each connector is connected to exactly one other connector, so the first connector must be a left connector and there must be a corresponding a^{-1} to represent its right connector on the right. Let y be the subsequence of x such that $x = aya^{-1}z$ for some sequence z. If y and z are both the empty string then x is idempotent since aa^{-1} is idempotent. Since no links cross, both y and z must satisfy the same conditions as x, and hence by induction, x is idempotent, since aea^{-1} and $aa^{-1}e$ are both idempotent for any idempotent e in the inverse semigroup. □

Note that the converse implication does not hold in general since $a^{-1}a$ is also idempotent; thus this formulation allows right connectors to precede left connectors just as well as succeed them. In practice this should not be a problem since it is likely that the grammar can be redesigned in such a way that unwanted idempotents do not occur.

7.3.5 A Semigroup for Syntax

Both the bracket semigroup and the free inverse semigroup accurately represent syntax according to link grammar, however both have advantages and disadvantages for practical application in representing syntax. The free inverse semigroup stores information about the parse in a Munn tree, however combinations which don't parse will be 'left over'. In the bracket semigroup, combinations which don't parse have a product of zero, so are ignored, but there is no memory of the parse.

For example, suppose nouns may optionally be preceded by an adjective (a) before taking a determiner (d) which we represent as $n_f = a^{-1}d^{-1} + d^{-1}$ in the L^1 algebra of the free inverse semigroup, and as $n_b = |a\rangle|d\rangle + |d\rangle$ in the L^1 algebra of the bracket semigroup. If the noun is now preceded by a determiner, d or $\langle d|$ respectively, then in the free inverse semigroup we have

$$dn_f = da^{-1}d^{-1} + dd^{-1}$$

while in the bracket semigroup we have $\langle d|n_b = 1$ since $\langle d|a\rangle = 0$. Thus the free inverse semigroup correctly stores the idempotent dd^{-1} but leaves the non-syntactic construction $da^{-1}d^{-1}$, while the bracket semigroup correctly cancels out this construction, but has no memory of the parse.

To get around this problem we combine the two structures; to do this we will need the *direct product*. Given two semigroups S_1 and S_2 the direct product is the cartesian product $S_1 \times S_2$ with the semigroup product defined by

$$(x_1, y_1) \cdot (x_2, y_2) = (x_1 x_2, y_1 y_2).$$

If S_1 and S_2 are inverse semigroups, then $S_1 \times S_2$ is an inverse semigroup, with inverse $(x, y)^{-1} = (x^{-1}, y^{-1})$.

Given a set A of links, we take the direct product of the free inverse semigroup on A and the bracket semigroup on A, modulo an equivalence which makes elements zero in the bracket semigroup zero in the product. That is, the semigroup for syntax is defined as

$$\mathrm{FIS}(A) \times B(A) \; / \equiv,$$

where $\equiv$ is defined by $(x, 0) \equiv (y, 0)$ for all $x, y \in \mathrm{FIS}(A)$. We are actually interested in the subsemigroup $S_s(A)$ generated by elements of the form $(a, \langle a|)$ and $(a^{-1}, |a\rangle)$ for all elements $a \in A$. We denote these elements $\langle a\|$ and $\|a\rangle$ respectively, and the idempotent $(aa^{-1}, 1)$ as $\langle a\|a\rangle$.

Our example then becomes

$$\langle d\|n_s = \langle d\|\Big(\|a\rangle\|d\rangle + \|d\rangle\Big) = \langle d\|d\rangle$$

where $n_s = \|a\rangle\|d\rangle + \|d\rangle$ is the representation of the noun in $S_s(A)$.

7.3.6 From Semigroups to Context Theories

If S is a semigroup, we can construct an algebra $L^1(S)$ of functions on S with multiplication defined by convolution (see section A.5).

If, however, S possesses a zero θ, then we would want this to be represented in the algebra by the zero of the algebra, rather than a function of θ. Let $\boldsymbol{\theta}$ denote the ideal generated by θ,

$$\boldsymbol{\theta} = \{\alpha\theta : \alpha \in \mathbb{R}\},$$

(assuming a real vector space). Then we are interested in the vector space $L^1(S)/\boldsymbol{\theta}$, that is the vector space of equivalence classes $x + \boldsymbol{\theta} = \{x + y : y \in \theta\}$. Addition and scalar multiplication in this space is defined by

$$(x + \boldsymbol{\theta}) + (y + \boldsymbol{\theta}) = x + y + \boldsymbol{\theta}$$
$$\alpha(x + \boldsymbol{\theta}) = \alpha x + \boldsymbol{\theta}$$

Since $L^1(S)$ is also an algebra, we can define multiplication on $L^1(S)/\boldsymbol{\theta}$ by

$$(x + \boldsymbol{\theta})(y + \boldsymbol{\theta}) = xy + \boldsymbol{\theta}.$$

The equivalence class $0+\boldsymbol{\theta}$ is now a zero of the vector space and the algebra; when there is no ambiguity, we shall simply denote it by 0. If $xy = \theta$ in S, then in the algebra $L^1(S)/\boldsymbol{\theta}$ we have $e_x e_y = 0$, where e_x denotes the function that is 1 on x and zero elsewhere.

Since $\boldsymbol{\theta}$ is a vector lattice supspace of $L^1(S)$, the space $L^1(S)/\boldsymbol{\theta}$ is a vector lattice; clearly it is also a lattice ordered algebra under the multiplication of S.

Since the L^1 norm is finite in the space $L^1(S)$ we can use it to define a linear functional:

$$\phi(u) = \|u^+\|_1 - \|u^-\|_1$$

Thus together with an assignment from words to elements of $L^1(S)$, we have a context theory.

7.3.7 Relating Link and Categorial Grammars

The inventors of link grammar describe a translation from Bar-Hillel type categorial grammars to link grammar (Sleator and Temperley, 1993). They describe it recursively in terms of a function E that takes a categorial expression and returns a link grammar expression. In our notation, it can be expressed as follows:

- The set of link types L is the set of categorial expressions.
- If a word has a set $\{x_1, x_2, \ldots, x_n\}$ of categorial expressions, then it is represented by the sum
$$E(x_1) + E(x_2) + \ldots E(x_n).$$
- The representation of a basic type A is
$$E(A) = |A\rangle + \langle A|.$$
- The representation of other categories is given by
$$\begin{aligned} E(x/y) &= |x/y\rangle + \langle x/y| + E(x)\langle y| \\ E(y\backslash x) &= |y\backslash x\rangle + \langle y\backslash x| + |y\rangle E(x) \end{aligned}$$

As Sleator and Temperley note, the size of the link grammar representation is linear in the size of the categorial grammar representation, thus they expect that translating

to link grammar would be an effective method of parsing categorial grammars. From our perspective, there is an additional potential use for the translation: the connection enables a new way of implementing statistical categorial grammars, using the statistical link grammar formalisms.

7.4 Discussion and Further work

Using the constructions of the previous section, we are able to describe a formalism that parses sentences in a purely algebraic fashion. The advantage of this algebraic description over the operator-based description is that the parse itself is stored in algebraic form and does not need to be reconstructed from information about which disjunct was used with each word. This is due to the extra structure provided by the free inverse semigroup which allows tree-like structures to be represented. It is this structure that we believe will also useful for constructing representations of meaning directly within the context theoretic framework. For example, it may be possible to find link grammars for natural language such that the Munn tree of a sentence describes relationships between the words in the sentence. This can already be seen to be true to some degree: for example, in the tree we showed for the sentence "They mashed their way through the thick mud", the branch relating to "thick" comes off the branches relating to "mud", in terms of idempotents, the idempotent representing "the thick mud" is more specific than that representing "the mud". The trees still bear little resemblance to the dependency trees that we are familiar with, however.

We have now described methods for representing meaning and syntax in algebra. The question arises how one may combine such methods to produce lexicalised algebraic representations of language incorporating both meaning and syntax. One may wish to choose a particular vector-based semantic formalism and a particular syntactic formalism and combine them. One way of doing this may be the mathematics of *free probability* (Voiculescu, 1997). The concept of freeness generalises the concept of independence to the case of non-commutative variables. Two sub-algebras A_1 and A_2 of an algebra are said to combine freely with respect to a linear functional ϕ if $\phi(x_1 x_2 \dots x_n) = 0$ whenever all the x_i satisfy $\phi(x_i) = 0$ and no two adjacent x_i are in the same sub-algebra. Given two non-commutative probability spaces, one can construct their free product as an algebra which has sub-algebras isomorphic to the original algebras and satisfying the condition of freeness. Thus one could choose a context-theory to represent meaning and a context-theory to represent syntax and build a combined context-theory using the free product, in which each word would map to a product of its original syntactic and semantic representations. The idea that meaning and syntax combine freely is appealing since we are

used to thinking of these two aspects of language separately; the concept of freeness may encapsulate this idea well, however exactly how it would work in practice remains to be seen.

We have left the question of how to compute with these new representations largely unanswered, however we are representing existing formalisms for which computational procedures already exist. Thus it may be possible to make use of existing algorithms with small adjustments to compensate for the differences that the context-theoretic perspective requires. It is our hope however that new and more efficient computational procedures will be brought to light by considering the algebraic approach, particularly in the area of statistical parsing. One area that seems particularly worthy of further investigation is the use of matrices to approximate elements of algebras, along the lines of the description we gave for Fock space operators in terms of matrices.

Chapter 8

Conclusion

We have presented a context-theoretic framework for natural language semantics. The framework is founded on the idea that meaning in natural language can be determined by context, and is inspired by techniques that make use of statistical properties of language by analysing large text corpora. Such techniques can generally be viewed as representing language in terms of vectors. These techniques are currently used in applications such textual entailment recognition, however the lack of a theory of meaning that incorporates these techniques means that they are often used in a somewhat ad-hoc manner. The purpose behind the framework is to provide a unified theoretical foundation for such techniques so that they may used in a principled manner.

Another major aim of the framework is to provide insight into how to best deal with the problems of uncertainty and ambiguity in natural language, especially when making use of logical representations of language. Because logical systems on their own are generally brittle, in natural language applications such as that of recognising textual entailment ways have to be found to make the systems more robust, and reasoning about uncertainty is one way in which this can be done. We have shown how logical semantics can be viewed from the context-theoretic perspective; this provided us with guidance as to how to represent uncertainty and ambiguity within the framework, incorporating statistical information about this uncertainty into the representations. We also outlined how a system making use of our ideas could be implemented. We then discussed in detail the relationship between ontological and vector-based representations of lexical semantics, presenting several ways of constructing vector based representations of a taxonomy.

Finally, we discussed the representation of syntactic structure within the framework, discussing categorial grammars and link grammars, and showing that the latter are particularly well suited to the context-theoretic framework.

There are many potential areas for future work. On the theoretical side, we believe that proving Conjecture 3.9 may provide further insight into the nature of meaning as context, as well as giving evidence for the nature of the context-theoretic framework. A

further interesting question is which algebras are isomorphic to the context algebra of some corpus model. It may be that stronger conditions can be placed on context theories to restrict the formalism to such algebras — such implementations would truly deserve to be called "context theories". A more general area of theoretical interest is the use of free probability to combine context theories — this seems to us a very promising area for future work that may lead to entirely new representations of language.

A recurring problem when considering the implementation of some of the ideas we have discussed is the issue of very high or infinite dimensionality of the vector spaces under consideration. A possible solution that we have suggested is the use of dimensionality reductions such as random projections. This is an area that requires much further work to determine the usefulness of such approaches, and the best way to apply them. The potential rewards however, should be great, allowing the representation of meaning and syntax using (relatively) low dimensionality vectors and matrices.

Of course it is hard to predict the benefits that may result from what we have presented — we have given a way of thinking about meaning in natural language that in many respects is new, although it consolidates the thinking inherent in many modern techniques within computational linguistics. What is new is the insistence on representing phrases and sentences using vectors in the same way that words often are, and it will undoubtedly be a while before we find the best methods of doing this, and the best representations and algorithms to compute with them.

Appendix A

Mathematical Methods for Computational Linguistics

This appendix provides a reference for foundational mathematical concepts that are necessary for an understanding of the thesis.

A.1 Semigroups, Groups and Fields

Definition A.1 (Semigroup). A *binary operation* on a set S is a function from $S \times S$ to S. The value of the binary operation $\cdot$ on two elements x and y in S is denoted $x \cdot y$. A semigroup $(S, \cdot)$ is a set S with a binary operation $\cdot$ which is *associative*:

$$(x \cdot y) \cdot z = x \cdot (y \cdot z).$$

This product is often denoted $x \cdot y \cdot z$ or simply xyz.

An element e of S is called *unity* if $es = se = s$ for all $s \in S$. There can only ever be at most one unity in S: if e_1 and e_2 are unities then $e_1 e_2 = e_1 = e_2$. A semigroup with unity is often called a *monoid*.

Definition A.2 (Free Semigroup). Let A be a set. The set A^* is the set of all finite sequences of symbols of elements of A. Then A^* is a semigroup (called the *free semigroup* on A) under concatenation of sequences, $x \cdot y = xy$ for $x, y \in A^*$.

Definition A.3 (Group). A group G is a monoid with unity e such that for each element $x \in G$ there is an element x^{-1}, called the *inverse* of x, such that $xx^{-1} = x^{-1}x = e$. A group is called *abelian* or *commutative* if $xy = yx$ for all $x, y \in G$.

Definition A.4 (Field). A field is a set F together with two operations $+$ and $\cdot$ called addition and multiplication such that F is an abelian group under addition with (additive) identity $0 \in F$, and a commutative monoid under multiplication, with (multiplicative) identity $1 \in F$, with $1 \neq 0$, such that every element $x \in F$ except 0 has a multiplicative

inverse x^{-1} (that is, $F - \{0\}$ is an abelian group under multiplication) and multiplication distributes over addition:

$$x \cdot (y + z) = x \cdot y + x \cdot z$$

Definition A.5 (Congruence). A congruence on a semigroup S is an equivalence relation R on S that is preserved under multiplication, so that if aRb then $xaRxb$ and $axRbx$. Let aR denote the set $\{x : aRx\}$, called the equivalence class of a. We can define a product on equivalence classes by $aR \circ bR = abR$. This semigroup is denoted S/R, and is called the quotient of S with respect to R.

If R_1 and R_2 are congruences on S then $R_1 \cap R_2$ is also a congruence relation. Since the universal relation U defined by xUy for all $x, y \in S$ is a congruence, we can find for every relation R on S the smallest congruence containing R as the intersection of all congruences R' with $R' \supseteq R$.

A.2 Vector Spaces

Definition A.6 (Vector Space). A vector space over a field F is a set V with two operations: addition, $V \times V \to V$, denoted $u+v$ where $u, v \in V$, and scalar multiplication: $F \times V \to V$, denoted αv where $\alpha \in F$ and $v \in V$, satisfying the following conditions:

- V is closed under addition and scalar multiplication;
- the vector space under addition forms an *abelian group*: addition is associative and commutative and there is an additive identity $0 \in V$ such that for every element $v \in V$ there is an element $-v$ such that[1] $v + (-v) = 0$;
- scalar multiplication is associative: $\alpha(\beta v) = (\alpha\beta)v$ for $\alpha, \beta \in F$ and $v \in V$;
- $1v = v$ where 1 is the multiplicative identity of F;
- scalar multiplication is distributive with respect to vector and scalar addition:

$$\begin{aligned} \alpha(u + v) &= \alpha u + \alpha v \\ (\alpha + \beta)v &= \alpha v + \beta v \end{aligned}$$

When the field F is that of the real or complex numbers $\mathbb{R}$ or $\mathbb{C}$, the vector space is called 'real' or 'complex' respectively. Unless otherwise stated, we shall be dealing exclusively with real vector spaces.

[1] In general we write $x - y$ for $x + (-y)$.

Definition A.7 (Finite-dimensional Real Vector Spaces). The most important examples for computational linguists are the n-dimensional real vector spaces, denoted $\mathbb{R}^n$. An element of $\mathbb{R}^n$ is denoted

$$x = (x_1, x_2, \ldots x_n),$$

where the x_i are the real valued *components* of x. The operations on $\mathbb{R}^n$ are defined as follows:

$$\begin{aligned} x + y &= (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n) \\ \alpha x &= (\alpha x_1, \alpha x_2, \ldots, \alpha x_n) \\ -x &= (-x_1, -x_2, \ldots, -x_n); \end{aligned}$$

the zero element is the element all of whose components are zero. Given a finite set S, we write $\mathbb{R}^S$ for the vector space $\mathbb{R}^{|S|}$; then each element of S corresponds to a dimension in $\mathbb{R}^S$.

A.2.1 Notions of Distance

The following sequence of definitions are to do with the notion of "distance" and "size" of objects. These concepts are of key importance in computational linguistics because we are often interested in "distances" between words—for example semantic distance. The types of space, in order of generality, are *metric space*, *normed space* and *inner product space*.

Definition A.8 (Metric). A metric d is a function on a set X satisfying:

$d(x, y) \geq 0$	(non-negativity)
$d(x, y) = 0$ if and only if $x = y$	(identity of indiscernibles)
$d(x, y) = d(y, x)$	(symmetry)
$d(x, z) \leq d(x, y) + d(y, z)$	(triangle inequality)

for all $x, y, z \in X$. A metric space is a set X together with a metric d.

The definition of metric is very general: it does not require the set X to be a vector space. In contrast, a more common way of defining distances on a vector space is via a *norm*:

Definition A.9 (Norm). If V is a vector space over a field F which is a subfield of the complex numbers, a norm $\| \cdot \|$ is a function from V to the real numbers satisfying:

$\|x\| \geq 0$	(positivity)
$\|\alpha x\| = \|\alpha\| \cdot \|x\|$	(positive scalability)
$\|x + y\| \leq \|x\| + \|y\|$	(triangle inequality)
$\|x\| = 0$ if and only if $x = 0$	(positive definiteness)

A *normed vector space* is a vector space together with a norm.

It is fairly straightforward to see that a norm $\|\cdot\|$ on a vector space V defines a metric d on V by $d(x, y) = \|x - y\|$.

Definition A.10 (l^p Norms). The most important examples are given by the l^p norms, for p a real number ≥ 1. For the vector space $\mathbb{R}^n$, the l^p norm of an element $x = (x_1, x_2, \ldots, x_n)$ is given by

$$\|x\|_p = (|x_1|^p + |x_2|^p + \ldots + |x_n|^p)^{1/p}$$

The l^∞ norm of x is defined as the supremum of $|x_i|$ over all components x_i of x.

Some of the most important instances of vector spaces, namely the Hilbert spaces, are those with an *inner product*, which corresponds to the familiar dot product on finite dimensional vector spaces. We give the definition here in terms of complex numbers for generality; we shall only ever need real vector spaces.

Definition A.11 (Inner Product). An inner product on a complex vector space is a function $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{C}$ satisfying for all $u, v, w \in V$ and $\alpha \in F$:

Additivity:	$\langle u, v + w \rangle = \langle u, v \rangle + \langle u, w \rangle$ $\langle u + v, w \rangle = \langle u, w \rangle + \langle v, w \rangle$
Nonnegativity:	$\langle v, v \rangle \geq 0$
Nondegeneracy:	$\langle v, v \rangle = 0 \quad \text{iff} \quad v = 0$
Conjugate symmetry:	$\langle u, v \rangle = \overline{\langle v, u \rangle}$
Sesquilinearity:	$\langle u, \alpha v \rangle = \alpha \langle u, v \rangle$

where $\overline{\alpha}$ denotes the complex conjugate of α. The definition clearly also holds when V is a real vector space. A vector space with an inner product defined is called an *inner product space*.

Note that conjugate symmetry implies that $\langle x, x \rangle$ is real for all x, and that conjugate symmetry and sesquilinearity together imply that

$$\langle \alpha x, y \rangle = \overline{\alpha} \langle x, y \rangle.$$

An inner product naturally defines a norm $\|\cdot\|$ on a vector space, by $\|x\| = \sqrt{\langle x, x \rangle}$.

Example A.12 (Dot Product). The inner product or dot product on $\mathbb{R}^n$ is defined by

$$\langle x, y \rangle = \sum_{i=1\ldots n} x_i y_i.$$

The norm of a vector in $\mathbb{R}^n$ corresponds to its length: $\|x\| = \sqrt{\sum_{i=1\ldots n} x_i^2}$.

A.2.2 Bases

Almost every vector space considered in computational linguistics comes with some basis, which can usually be conceptually linked to the notion of context. The notion of a *basis* in a vector space is also very important in relation to *vector lattices* (see section A.4).

Definition A.13 (Basis). A basis is a set B of elements of a vector space V over a field F, such that the elements are *independant*, i.e., if

$$\sum_{b_i \in B} \alpha_i b_i = 0$$

for some set of $\alpha_i \in F$, then necessarily $\alpha_i = 0$ for all i; and B *spans* V, i.e., for each element $x \in V$,

$$x = \sum_{b_i \in B} \beta_i b_i$$

for some set of values $\beta_i \in F$.

Two elements x, y in an inner product space V are called *orthogonal* if $\langle x, y \rangle = 0$. An orthonormal basis for V is a basis B such that any two distinct elements of B are orthogonal and the magnitude of each element in B is 1, i.e. $\langle b, b \rangle = 1$ for all $b \in B$.

Example A.14 (Orthonormal Basis for $\mathbb{R}^n$). An orthonormal basis for the vector space $\mathbb{R}^n$ is given by the set $\{e_1, e_2, \ldots e_n\}$ where $e_1 = (1, 0, 0, \ldots 0)$, $e_2 = (0, 1, 0, \ldots 0)$, $\ldots$, $e_n = (0, 0, 0, \ldots 1)$. In this way, for the vector space $\mathbb{R}^S$, we can associate a basis element e_s with each element $s \in S$.

A.2.3 Completeness

Completeness is a property of vector spaces which is difficult to grasp conceptually, and is not that important to understand in relation to applications in computational linguistics. However, it is a property that is possessed by a lot of interesting vector spaces, and is often required of vector spaces since it leads to things being mathematically very "well behaved".

Definition A.15 (Limit). Let $a_1, a_2 \ldots$ be an infinite sequence of real numbers. A real number a is said to be the limit of the sequence if and only if for every real number $\epsilon > 0$, there is a natural number n_0 such that for all $n > n_0$, $|a_n - a| < \epsilon$.

Definition A.16 (Completeness). Given a metric space X with metric function d, a sequence $x_1, x_2, \ldots$ is called *Cauchy* if for every positive real number a, there is an integer n_0 such that for all integers $m, n > n_0$, $d(x_m, x_n) < a$. If every Cauchy sequence has a limit in X, the metric space is called *complete*.

A *Banach space* is a normed vector space which is complete with respect to the metric d defined by $d(x, y) = \|x - y\|$. A *Hilbert space* is a vector space with an inner product which is complete with respect to the metric defined by the inner product norm, $d(x, y) = \sqrt{\langle x - y, x - y \rangle}$. A Hilbert space is thus a special kind of Banach space.

A.2.4 l^p and L^p Spaces

We shall often need to deal with infinite dimensional vector spaces, for example, we shall often want to associate a dimension with each sequence in a set of sequences A^*. When we do this, not all vectors will have finite norm, and precisely which ones do depends on which norm we use. We can thus categorise subspaces according to which norms are guaranteed to be finite. For $p \geq 1$ we define the l^p space to be the vector space of all infinite sequences x of real numbers $x = (x_1, x_2, \ldots)$ such that $\sum_i |x_i|^p$ is finite, together with the l^p norm. The l^∞ space is the set of all vectors with finite components, together with the l^∞ norm. All the l^p spaces are Banach spaces, and only the l^2 space is a Hilbert space.

If S is a countable set, we shall often want to consider real valued functions f on S as vectors. We can consider such functions as sequences of real numbers: writing $S = \{s_1, s_2, \ldots\}$, we can think of f as a sequence $(f(s_1), f(s_2), \ldots)$. We denote by $L^p(S)$ the set of functions on S which are in the corresponding l^p space when viewed as sequences.

A.2.5 New vector spaces from old

Definition A.17 (Direct Sum). Given two vector spaces U and V we can construct a vector space $U \oplus V$ called the *direct sum* of U and V. The direct sum is simply the cartesian product $U \times V$ with vector operations defined component-wise:

$$\begin{aligned} (u_1, v_1) + (u_2, v_2) &= (u_1 + u_2, v_1 + v_2) \\ \alpha(u, v) &= (\alpha u, \alpha v) \end{aligned}$$

where $u_i \in U, v_i \in V, \alpha \in F$. If U and V are Hilbert spaces, then $U \oplus V$ denotes the Hilbert space with the inner product defined by

$$\langle (u_1, v_1), (u_2, v_2) \rangle = \langle u_1, u_2 \rangle + \langle v_1, v_2 \rangle$$

The dimension of $U \oplus V$ is equal to the sum of the dimensions of U and V.

Definition A.18 (Tensor Product). The *tensor product* $U \otimes V$ of two vector spaces U and V is constructed by taking the vector space generated by the cartesian product $U \times V$ and factoring out the subspace generated by the equations:

$$\begin{aligned} (u_1 + u_2) \otimes v &= u_1 \otimes v + u_2 \otimes v \\ u \otimes (v_1 + v_2) &= u \otimes v_1 + u \otimes v_2 \\ \alpha u \otimes v &= u \otimes \alpha v = \alpha(u \otimes v) \end{aligned}$$

where $u_i, u \in U$, $v_i, v \in V$ and $\alpha \in F$.

If U and V are Hilbert spaces, the tensor product is again a Hilbert space, with inner product defined by

$$\langle u_1 \otimes v_1, u_2 \otimes v_2 \rangle = \langle u_1, u_2 \rangle \langle v_1, v_2 \rangle.$$

The dimension of $U \otimes V$ is equal to the product of the dimensions of U and V.

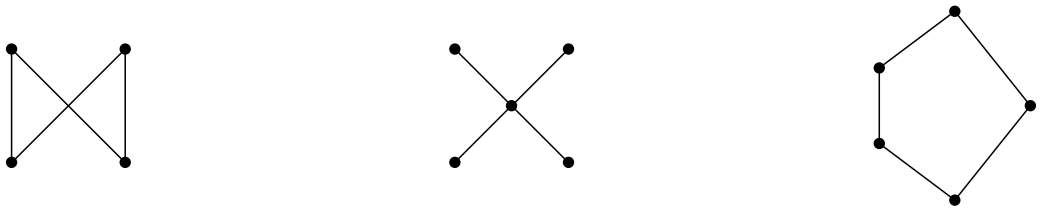
A.3 Lattice Theory

The concepts described in this section deal with relationships between objects. One of the most important types of relationship that we consider on sets of objects is that of a *partial ordering*. An example of this is the hypernymy relation between words (or equivalently the **is-a** or subsumption relation between concepts), discussed in section **??**. Another example is the subset relation on a set of sets.

These relations often satisfy much stronger conditions, which we classify in sequence: semilattices, lattices, modular lattices, distributive lattices and Boolean algebras. All of these have important characteristics which may also be expressed in algebraic terms.

Definition A.19 (Partial Ordering). A partial ordering on a set S is a relation $\leq$ that satisfies, for all $x, y, z \in S$:

$x \leq x$	(reflexivity)
if $x \leq y$ and $y \leq x$ then $x = y$	(antisymmetry)
if $x \leq y$ and $y \leq z$ then $x \leq z$	(transitivity)



(a) A partial ordering that is not a lattice

(b) An embedding of the partial ordering in a lattice

(c) The five element non-modular lattice

Figure A.1: Hasse diagrams

If $a \leq b$ then we say a is *contained in* or *is less than* b. An example of a partial ordering is the set inclusion relation, $\subseteq$ on a set of subsets of a set, or the 'less than or equal' relation on the natural numbers.

The following definition is useful for describing properties of partial orderings, and drawing diagrams of them:

Definition A.20 (Preceding elements). Write $x < y$ if $x \leq y$ and $x \neq y$ in L. We say that x *precedes* y and write $x \prec y$ if $x < y$ and there is no element z such that $x < z < y$.

Partial orderings are often depicted using *Hasse diagrams*. Some examples are shown in figure A.1. Elements of the lattice are shown as nodes, while the relation $\prec$ between elements is shown by connecting nodes with an edge, such that the lesser element is below the greater element in the diagram. For example, figure A.1(a) shows a four element set with a partial ordering which may be described by the relation $\leq$ on the set $\{a, b, c, d\}$ defined by $a \leq c$, $b \leq c$, $a \leq d$, $b \leq d$. Hasse diagrams such as these are used to show partial orderings *up to isomorphism*, that is, when we are not interested in the labeling of the nodes, only the nature of the partial order itself.

Definition A.21 (Semilattice and Lattice). An *upper bound* of a subset T of a partially ordered set S is an element s such that $t \leq s$ for all $t \in T$. The *least upper bound* of T if it exists (also called *supremum* or *join*) is the upper bound which contains every upper bound. The join of a set T is denoted $\bigvee T$, or if T consists of two elements x and y their join is denoted $x \vee y$.

Similarly a *lower bound* of T is an element s' such that $s' \leq t$ for all $t \in T$. The *greatest lower bound* if it exists (also called the *infimum* or *meet* of T) is the lower bound which is contained in every other lower bound. The meet of T is denoted $\bigwedge T$; the meet of two elements x and y is denoted $x \wedge y$.

A *meet semilattice* (or simply *semilattice*) is a partially ordered set in which every pair of elements has a greatest lower bound. Similarly, a *join semilattice* is a partially ordered set in which every pair of elements has a greatest lower bound.

A *lattice* is a partially ordered set in which any two elements have a least upper bound and a greatest lower bound; a lattice is thus both a join and a meet semilattice. A lattice is called *complete* if every subset of S has a least upper bound and greatest lower bound; all finite lattices are complete.

Figure A.1(a) shows a partial ordering that is *not* a lattice: the join of the two lesser elements is not well defined, similarly, the meet of the two greater elements is not defined. Figure A.1(b) does show a lattice: the new element acts as the missing join and meet.

A semilattice can be characterised as a semigroup S with the binary operation $\wedge$ satisfying *idempotence* and *commutativity*:

$$\begin{aligned} x \wedge x &= x \\ x \wedge y &= y \wedge x \end{aligned}$$

respectively. The partial ordering can be recovered by defining $x \leq y$ iff $x \wedge y = x$. Similarly, a lattice can be characterised as a set S together with two operations $\wedge$ and $\vee$ such that $(S, \wedge)$ and $(S, \vee)$ are semilattices (according to the above characterisation), satisfying the *absorption* laws:

$$\begin{aligned} x \vee (x \wedge y) &= x \\ x \wedge (x \vee y) &= x \end{aligned}$$

Definition A.22 (Modularity). A modular lattice is a lattice L satisfying the *modular identity*: if $x \leq z$ then

$$x \vee (y \wedge z) = (x \vee y) \wedge z,$$

for all $x, y, z \in L$.

Figure A.1(b) shows a five element modular lattice, while A.1(c) shows a lattice that is not modular; it is the only five element non-modular lattice (up to isomorphism).

The proof of the following proposition is in Birkhoff (1973):

Proposition A.23. *The modular lattices are those which do not have the five element non-modular lattice of figure A.1(c) as a sub-lattice.*

Definition A.24 (Distributivity, Complement and Boolean Algebra). A lattice

is called *distributive* if it satisfies

$$\begin{aligned} x \vee (y \wedge z) &= (x \vee y) \wedge (x \vee z) \\ x \wedge (y \vee z) &= (x \wedge y) \vee (x \wedge z) \end{aligned}$$

A lattice is *complemented* if for every element a there is an element a' such that $a \vee a' = 1$ and $a \wedge a' = 0$. A complemented distributive lattice is called a *Boolean algebra*.

A.3.1 Functions between partial orders

It is very important for our work to characterise the nature of functions between partial orderings. Of special importance are those that preserve the partial ordering, and in the case of lattices, preserve meets and joins. We define some important types of functions, and give examples.

Definition A.25 (Order Embeddings). A function f from one partially ordered set S to another T is called *monotone* or *order-preserving* if $a \leq b$ in S implies $f(a) \leq f(b)$ in T. Conversely, if $f(a) \leq f(b)$ implies $a \leq b$ then f is called *order-reflecting*. An *order embedding* is a function that is both order-preserving and order-reflecting. A *completion* of a partially ordered set S is an order embedding of S into a complete lattice.

Definition A.26 (Lattice Homomorphisms). If S and T are semilattices, a function f from S to T is a *semilattice homomorphism* if $f(a \wedge b) = f(a) \wedge f(b)$ (where $\wedge$ can represent the meet or the join operation). If S and T are lattices, a *lattice homomorphism* is a function that is both a meet semilattice and join semilattice homomorphism, i.e. $f(a \wedge b) = f(a) \wedge f(b)$, and $f(a \vee b) = f(a) \vee f(b)$. A *lattice isomorphism* is a bijective lattice homomorphism, i.e. for each element b in T there is exactly one element a in S such that $f(a) = b$. If a lattice isomorphism exists between two lattices they are said to be *isomorphic*.

Often we may be dealing with partial orders but require something with more structure than that relation provides. For example, we may like to be able to define meets and joins to make the partial ordering into a lattice. The concepts of *principal ideals* and their duals, *principal filters*, allow us to do this:

Definition A.27 (Ideals and Filters). A *lower set* in a partially ordered set S is a set T such that for all $x, y \in T$, if $y \leq x$ then $y \in T$. Similarly, an upper set in S is a set T' such that for all $x, y \in T'$, if $y \geq x$ then $y \in T$.

The *principal ideal generated by an element* x in a partially ordered set S is defined to be the lower set $\downarrow(x) = \{y \in S : y \leq x\}$. Similarly, the *principal filter generated by* x is the upper set $\uparrow(x) = \{y \in S : y \geq x\}$.

Proposition A.28 (Ideal Completion). *If S is a partially ordered set, then $\downarrow(\cdot)$ can be considered as a function from S to the powerset 2^S. Under the partial ordering defined by set inclusion, the set of lower sets form a complete lattice, and $\downarrow(\cdot)$ is a completion of S, the* ideal completion. *Similarly, the function $\uparrow(\cdot)$ is the* filter completion *of S: it is an embedding into the complete lattice of upper sets, again ordered by inclusion.*

A.4 Riesz Spaces and Positive Operators

The previous sections have described formalisms commonly used to describe meaning: broadly speaking, that of vector spaces and that of lattices. Until now, little attention within computational linguistics has been paid to how to combine these two areas. There is a large body of research within mathematical analysis into an area which merges the two formalisms: the study of *partially ordered vector spaces*, *vector lattices* (or *Riesz spaces*), and *Banach lattices*, and a special class of operators on these spaces called *positive operators*.

The definitions and propositions of this section can be found in Abramovich and Aliprantis (2002) and Aliprantis and Burkinshaw (1985).

Definition A.29 (Partially ordered vector space). A partially ordered vector space V is a real vector space together with a partial ordering $\leq$ such that:

if $x \leq y$ then $x + z \leq y + z$
if $x \leq y$ then $\alpha x \leq \alpha y$

for all $x, y, z \in V$. Such a partial ordering is called a *vector space order* on V. If $\leq$ defines a lattice on V then the space is called a *vector lattice* or *Riesz space*.

Example A.30 (Lattice Structure of l^p Spaces). The l^p spaces defined earlier are vector lattices under the component-wise partial ordering defined by $x \leq y$ if and only if $x_i \leq y_i$ for all i, where $x = (x_1, x_2, \ldots)$ and $y = (y_1, y_2, \ldots)$.

A vector x in V is called *positive* if $x \geq 0$. The *positive cone* of a partially ordered vector space V is the set $V^+ = \{x \in V : x \geq 0\}$

The positive cone has the following properties:

$$\begin{aligned} X^+ + X^+ &\subseteq X^+ \\ \alpha X^+ &\subseteq X^+ \\ X^+ \cap (-X^+) &= \{0\} \end{aligned}$$

Any subset C of V satisfying the above three properties is called a *cone* of V.

Proposition A.31. *If C is a cone in a real vector space V, then the relation $\leq$ defined by $x \leq y$ iff $y - x \in C$ is a vector space order on V, with $X^+ = C$.*

Operators which map positive elements to positive elements are called *positive*; there is a large body of work studying such operators. This idea leads to some useful definitions of particular positive elements of a vector lattice corresponding to an arbitrary element x. The *positive part* of x is denoted x^+ and is defined by $x^+ = x \vee 0$. Similarly the *negative part* is $x^- = (-x) \vee 0$, and the *absolute value* is $|x| = x \vee -x$. There are a number of useful identities concerning these definitions:

Proposition A.32. *The following identities hold for elements x, y in a vector lattice:*

(a). $x = x^+ - x^-$

(b). $|x| = x^+ + x^-$

(c). $x \wedge y = \frac{1}{2}(x + y - |x - y|)$

(d). $x \vee y = \frac{1}{2}(x + y + |x - y|)$

A.4.1 Abstract Lebesgue Spaces

A Riesz space together with a norm is called a *normed Riesz space*. If the space is complete with respect to the norm (that is, it is also a Banach space) it is called a *Banach lattice*.

Definition A.33 (Abstract Lebesgue Space). An Abstract Lebesgue (or AL) space is a Banach lattice V such that

$$\|x + y\| = \|x\| + \|y\|$$

for all x, y in V with $x \geq 0$, $y \geq 0$ and $x \wedge y = 0$.

A.5 Algebras

The concept of an *algebra over a field* (or often simply "an algebra") is of importance in abstract analysis, and for example providing (in the case of a special type of algebra called a C*-algebra) an alternative formulation of the mathematics of quantum mechanics. They also provide the foundation for the theory of non-commutative probability.

Definition A.34. An algebra is a vector space A over a field K together with a binary

operation $(a, b) \mapsto ab$ on A that is bilinear, i.e.

$$a(\alpha b + \beta c) = \alpha ab + \beta ac$$
$$(\alpha a + \beta b)c = \alpha ac + \beta bc$$

and associative, i.e. $(ab)c = a(bc)$ for all $a, b, c \in A$ and all $\alpha, \beta \in K$.[2]

Definition A.35 (Multiplication on $L^1(S)$). If S is a semigroup then $L^1(S)$ is an algebra under multiplication defined by convolution:

$$(u \cdot v)(x) = \sum_{y,z:yz=x} u(y)v(z),$$

where $u, v \in L^1(S)$ and $x, y, z \in S$.

A.5.1 Linear Operators

The concept of an algebra arose through abstraction of concrete examples of algebras, in particular the algebra of *linear operators* on a vector space. These are special kinds of functions on the vector space that agree with the vector space structure. They are defined as follows:

Definition A.36 (Linear Operator). A linear operator from a vector space U to a vector space V both over a field F is a function A from U to V satisfying

$$A(\alpha x + \beta y) = \alpha Ax + \beta Ay$$

for all $x, y \in U$ and $\alpha, \beta \in F$.

Note that the operation of A on an element x is denoted simply Ax (without brackets). In addition, we shall often refer to a linear operator simply as an "operator"—in this case linearity is assumed.

Linear operators themselves form a vector space, with vector space operations defined by

$$\begin{aligned} (A + B)x &= Ax + Bx \\ (\alpha A)x &= \alpha Ax \\ 0x &= 0 \end{aligned}$$

[2]Some authors do not place the requirement that an algebra is associative, in which case our definition would refer to an *associative algebra*.

Since the operation of functions is necessarily associative, it is easy to see that the linear operators form an algebra under function composition.

A.5.2 Positive operators

Definition A.37 (Positive Operators). An operator A on a vector space V is called *positive* if $x \geq 0$ implies $Ax \geq 0$. It is called *regular* if it can be denoted as the difference between two positive operators.

Surprisingly, the set of regular operators on a vector lattice themselves form a vector lattice:

Proposition A.38 (Riesz-Kantarovič). *The positive cone defines a vector space order on the vector space of operators on V. This order makes the space of regular operators a vector lattice. Specifically the meet and join of two regular operators A and B are given by*

$$\begin{aligned}(A \wedge B)x &= \inf\{Ay + Bz : y, z \in V^+ \text{and } y + z = x\} \\ (A \vee B)x &= \sup\{Ay + Bz : y, z \in V^+ \text{and } y + z = x\}.\end{aligned}$$

Definition A.39 (Lattice Homomorphism). A positive operator A between two vector lattices is called *lattice homomorphism* if $A(x \vee y) = Ax \vee Ay$. An lattice homomorphism that is a one-to-one function is called a *lattice isomorphism*.

The following proposition shows the importance of lattice homomorphisms:

Proposition A.40. *For a positive operator A between two Riesz spaces U and V, the following statements are equivalent:*

(a). A is a lattice homomorphism.

(b). $A(x^+) = (Ax)^+$ for each $x \in U$.

(c). $A(x \wedge y) = Ax \wedge Ay$ for all $x, y \in U$.

(d). $|Ax| = A|x|$ for each $x \in U$.

(e). $x \wedge y = 0$ in U implies $Ax \wedge Ay = 0$ in V.

Bibliography

Y. A. Abramovich and Charalambos D. Aliprantis. *An Invitation to Operator Theory*. American Mathematical Society, 2002.

V. M. Abrusci. Phase semantics and sequent calculus for pure noncommutative classical linear propositional logic. *The Journal of Symbolic Logic*, 56:1403–1451, 1991.

Elena Akhmatova. Textual entailment resolution via atomic propositions. In Dagan et al. (2005b).

Charalambos D. Aliprantis and Owen Burkinshaw. *Positive Operators*. Academic Press, 1985.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

Yehoshua Bar-Hillel. On syntactical categories. *Journal of Symbolic Logic*, 15(1):1–16, 1950.

Yehoshua Bar-Hillel. *Language and Information: Selected Essays on their Theory and Application*. Addison-Wesley Publishing Co., Reading, MA., 1964.

Samuel Bayer, John Burger, Lisa Ferro, John Henderson, and Alexander Yeh. MITRE's submissions to the EU pascal RTE challenge. In Dagan et al. (2005b).

Garrett Birkhoff. *Lattice Theory*. Amer. Math. Soc. Colloquium Publications, New York, 1973.

Patrick Blackburn and Johan Bos. *Representation and Inference for Natural Language*. CSLI, 2005.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

Johan Bos. Towards wide-coverage semantic interpretation. In *Proceedings of Sixth International Workshop on Computational Semantics IWCS-6*, page 42?53, 2005.

Johan Bos and Katja Markert. When logical inference helps determining textual entailment (and when it doesnt). In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

Jordan Boyd-Graber, David Blei, and Xiaojin Zhu. A topic model for word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1024–1033, 2007. URL `http://www.aclweb.org/anthology/D/D07/D07-1109`.

E. Briscoe, John Carroll, and R. Watson. The second release of the rasp system. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions, Sydney, Australia*, 2006.

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

Junfu Cai, Wee Sun Lee, and Yee Whye Teh. Improving word sense disambiguation using topic features. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1015–1023, 2007. URL `http://www.aclweb.org/anthology/D/D07/D07-1108`.

Claudia Casadio and Joachim Lambek. A tale of four grammars. *Studia Logica*, 71(3): 315–329, 2002.

Daoud Clarke. Meaning as context and subsequence analysis for textual entailment. In *Proceedings of the Second PASCAL Recognising Textual Entailment Challenge*, 2006.

James R. Curran and Marc Moens. Improvements in automatic thesaurus extraction. In *ACL-SIGLEX Workshop on Unsupervised Lexical Acquisition*, Philadelphia, 2002.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, 2005a.

Ido Dagan, Oren Glickman, and Bernardo Magnini, editors. *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, 2005b.

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

Rodolfo Delmonte, Sara Tonelli, Marco Aldo Piccolino Boniforti, Antonella Bristot, and Emanuele Pianta. VENSES - a linguistically-based system for semantic evaluation. In Dagan et al. (2005b).

Christaine Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, Massachusetts, 1989.

John R. Firth. A synopsis of linguistic theory 1930–1955. In F. Palmer, editor, *Selected Papers of J. R. Firth*. Longman, London, 1957a.

John R. Firth. Modes of meaning. In *Papers in Linguistics 1934–1951*. Oxford University Press, London, 1957b.

Abraham Fowler, Bob Hauser, Daniel Hodges, Ian Niles, Adrian Novischi, and Jens Stephan. Applying COGEX to recognize textual entailment. In Dagan et al. (2005b).

Maayan Geffet and Ido Dagan. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, University of Michigan, 2005. URL `http://www.aclweb.org/anthology/P05-1014`.

J. Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

Oren Glickman and Ido Dagan. A probabilistic setting and lexical cooccurrence model for textual entailment. In *ACL-05 Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, 2005.

Petr Hájek. Basic fuzzy logic and BL-algebras. *Soft Computing—A Fusion of Foundations, Methodologies and Applications*, 2(3):124–128, September 1998.

Zellig Harris. *Mathematical Structures of Language*. Wiley, New York, 1968.

Zellig Harris. Distributional structure. In Jerrold J. Katz, editor, *The Philosophy of Linguistics*, pages 26–47. Oxford University Press, 1985.

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. Recognizing textual entailment with lcc's groundhog system. In *Proceedings of the Second PASCAL Challenges Workshop*, 2006.

P. Hinman. *Fundamentals of Mathematical Logic*. A. K. Peters, 2005.

Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the 15th Conference on Uncertainty in AI*, pages 289–296. Morgan Kaufmann, 1999.

Patrick Honeybone. J. R. Firth. In S. Chapman and C. Routledge, editors, *Key Thinkers in Linguistics and the Philosophy of Language*. Edinburgh University Press, 2005.

J. M. Howie. *An Introduction to Semigroup Theory*. Academic Press, London, 1976. ISBN 0-12-356950-8.

Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics*, 1998.

Hans Kamp and Uwe Reyle. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*, volume 42 of *Studies in linguistics and philosophy*. Kluwer, Dordrecht, 1993.

Aaron Nathan Kaplan. *A computational model of belief*. PhD thesis, University of Rochester. Dept. of Computer Science, 2000.

Adam Kilgarriff. Thesauruses for natural language processing. In *Proceedings of the Joint Conference on Natural Language Processing and Knowledge Engineering*, pages 5–13, Beijing, China, 2003.

Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 2003.

Erwin Kreyszig. *Introduction to Functional Analysis with Applications*. Robert E. Krieger Publishing Company, Malabar, Florida, 1989.

S. Kundu and J. Chen. Fuzzy logic or Lukasiewicz logic: a clarification. In Zbigniew W. Raś and Maria Zemankova, editors, *Proceedings of the 8th International Symposium on Methodologies for Intelligent Systems*, volume 869 of *LNAI*, pages 56–64, Berlin, October 1994. Springer. ISBN 3-540-58495-1.

John Lafferty, Daniel Sleator, and Davy Temperley. Grammatical trigrams: A probabilistic model of LINK grammar. In *Proc. of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language. Cambridge, MA, 1992*, pages 89–97, Menlo Park, CA, 1992. AAAI Press.

J. Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65: 154–169, 1958.

J. Lambek. From categorial grammar to bilinear logic. In Kosta Došen and Peter Schroeder-Heister, editors, *Substructural Logics*, pages 207–238. Oxford Univ. Press, 1993.

Joachim Lambek. Type grammars as pregroups. *Grammars*, 4(1):21–39, 2001.

Lillian Lee. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-1999)*, pages 23–32, 1999.

Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL '98)*, pages 768–774, Montreal, 1998.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

Christopher Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.

W. D. Munn. Free inverse semigroup. *Proceedings of the London Mathematical Society*, 29:385–404, 1974.

National Library of Medicine. UMLS Knowledge Sources. National Library of Medicine, U.S. Dept. of Health and Human Services, 8th edition, 1998.

N. Nilsson. Probabilistic logic. *ai*, 28:71–87, 1986.

Christos H. Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In Laura Haas and Ashutosh Tiwary, editors, *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, volume 27, pages 159–168, Seattle, Washington, 1998. ACM Press.

Mati Pentus. Models for the lambek calculus. *Annals of Pure and Applied Logic*, 75: 179–213, 1995.

Rajat Raina, Aria Haghighi, Christopher Cox, Jenny Finkel, Jeff Michels, Kristina Toutanova, Bill MacCartney, Marie-Catherine de Marneffe, Christopher D. Manning, and Andrew Y. Ng. Robust textual inference using diverse knowledge sources. In Dagan et al. (2005b).

Magnus Sahlgren and Jussi Karlgren. Vector-based semantic analysis using random indexing for cross-lingual query expansion. *Lecture Notes in Computer Science*, 2406: 169–176, 2002.

Daniel D. Sleator and Davy Temperley. Parsing english with a link grammar. Technical Report CMU-CS-91-196, Department of Computer Science, Carnegie Mellon University, 1991.

Daniel D. Sleator and Davy Temperley. Parsing english with a link grammar. In *The Third International Workshop on Parsing Technologies*, August 1993.

Marta Tatu and Dan I. Moldovan. A logic-based semantic approach to recognizing textual entailment. In *ACL*. The Association for Computer Linguistics, 2006.

Dan-Virgil Voiculescu. *Free Probability Theory*. American Mathematical Society, 1997.

Julie Weeds. *Measures and Applications of Lexical Distributional Similarity*. PhD thesis, Department of Informatics, University of Sussex, 2003.

Julie Weeds, David Weir, and Diana McCarthy. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th International Conference of Computational Linguistics, COLING-2004, Geneva, Switzerland.*, 2004.

Dominic Widdows. Orthogonal negation in vector spaces for modelling word-meanings and document retrieval. In *Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics, 2003, Sapporo, Japan*, pages 136–143, 2003.

Dominic Widdows. *Geometry and Meaning*. Center for the Study of Language and Information, Stanford, 2004.

Ludwig Wittgenstein. *Philosophical Investigations*. Macmillan, New York, 1953. G. Anscombe, translator.

Mary McGee Wood. *Categorial Grammars*. Routledge, London, 1993.

F. M. Zanzotto, A.AMoschitti, M. Pennacchiotti, and M. T. Pazienza. Learning textual entailment from examples. In *Proceedings of the Second PASCAL Challenges Workshop*, 2006.

Index