Mining for Meaning

The Extraction of Lexico-Semantic Knowledge from Text

Tim Van de Cruys

draft

Contents

Introduction

Human readers have the extraordinary capability to infer the meaning of a word from text, even if they have never heard of the word before, and there are no visual cues present to support its interpretation. Take for example sentences (1) – (3):

(1) **Fomalhaut** staat op 24 lichtjaren van de zon, en is 15 keer zo helder.
Fomalhaut stands on 24 light years of the sun and is 15 times as bright
Fomalhaut is 24 light years away from the sun, and is 15 times as bright.

(2) Zelfs een rijpe **kumquat** smaakt nog tamelijk zuur.
Even a ripe kumquat tastes still rather sour
Even a ripe kumquat still tastes rather sour.

(3) Op een statig deuntje danste men de **pavane**.
On a stately tune danced one the pavane
On a stately tune, they danced the pavane.

Even if we have never heard of words like *Fomalhaut*, *kumquat* and *pavane* before, we can still make reasonable assumptions about their meaning. The surrounding words allow us to assume that *Fomalhaut* is probably a star, that *kumquat* is some kind of edible organic object – most likely a fruit – and that a *pavane* is some kind of dance. We are able to infer the meaning of words because the context gives us cues about their semantic content. Likewise, we might be able to use the context of unknown words to infer their meaning automatically.

Of course, we can only get at the meaning of *Fomalhaut* because we are already familiar with the other words in the sentence, such as *lichtjaar* 'light year' and *zon* 'sun'. Likewise, we know what kind of objects might be ripe and taste sour, and we have a general idea of what it is that can be danced. The situation becomes more complicated if we want to use a computer to automatically infer the meaning of words: unlike humans, a computer does not have any a priori knowledge of words whatsoever. For this reason, most work on the acquisition of semantics from text has focused on

semantic similarity. Determining how similar a word is to other known words is much easier than determining what its actual meaning is. It is hard to extract the meaning of *Fomalhaut* from scratch, but it is much easier to determine that *Fomalhaut* appears in the same contexts as words like *Sirius* and *Betelgeuse*. Likewise, the word *kumquat* will appear in similar contexts as words like *orange*, *lemon* and *apple*, and *pavane* will be found in locations similar to words like *bourrée*, *gigue* or *polka*.

The semantic similarity of a word can thus be determined by accumulating its different contexts in a large corpus, and comparing those contexts to the contexts of other words. If two words have similar contexts, they are likely to be semantically related. Likewise, if two words share only few or no contexts at all, they are probably semantically unrelated. This process of determining a word's semantics by looking at the way it is distributed in texts is called DISTRIBUTIONAL SIMILARITY, and it will be the main foundation of this dissertation.

The extraction of semantics from text is a very broad and extensive subject. It therefore makes sense to clearly demarcate what will be our object of research, and also mention what this dissertation will not be about. The main subject of this dissertation is the *lexico-semantic* extraction of *nouns* from large-scale *written corpora*. The three italicized words are important here. By *lexico-semantic* extraction, we mean that we restrict ourselves to the extraction of semantic information on the lexical level: we are interested in extracting the semantics of single words, as they might be described in a dictionary. Of course, the lexical level is only a part of the vast domain that is semantics. Humans combine words into sentences to form complex meanings. Moreover, the combination of words has an influence on the meaning of the combined parts. These phenomena belong to the domain of *compositional semantics*, and they are beyond the scope of this dissertation.

Secondly, this dissertation mainly investigates the extraction of *nouns*. It goes without saying that there are many more word classes – adjectives, verbs and even function words – that deserve our attention with regard to the determination of their semantics. Verbs, in particular, make up an interesting word class with complex semantic behaviour. This dissertation, however, is limited to the extraction of nouns, and the lexico-semantic extraction of other word classes is again largely beyond its scope.[1]

Thirdly, we will investigate the lexical semantics of nouns by looking at their distribution in large *written text corpora* of newspaper texts. It is very well possible that the semantics present in these corpora is very different from the semantics to be found in spoken corpora, or in sources that go beyond the realm of linguistics. The

[1] Note, however, that we will touch upon the extraction of verb semantics in the last chapter, where we discuss the extraction of selectional preferences.

semantics that we investigate in this dissertation, however, will be the semantics of words as they appear in written newspaper texts.

With these reservations set aside, we can introduce the three main research questions that will be investigated in this dissertation. First of all, we want to determine what kind of semantic similarity is captured by different context models. We will present three different groups of models (each based on a different notion of context), and we will quantitatively investigate what kind of semantic similarity is captured by them. At the same time, we will investigate the usefulness of the models (and the resulting lexico-semantic resources) in a number of applications, notably multi-word expression extraction and word sense discrimination.

Secondly, we want to investigate the applicability of dimensionality reduction methods for semantic similarity extraction. Dimensionality reduction – or factorization – is the collective name for mathematical techniques that try to reduce an abundant number of overlapping features to a limited number of independent, informative dimensions. We will discuss a number of dimensionality reduction methods, and quantitatively evaluate whether they are beneficial for semantic similarity extraction. Again, their usefulness is also investigated in two applications: word sense discrimination and selectional preference induction.

Thirdly, we will investigate the use of three-way methods for lexico-semantic information extraction. Up till now, most research on the extraction of lexical semantics from text has focused on two-way methods, in which two-way co-occurrences (e.g. terms $\times$ documents) are used. Co-occurrences need not be limited to two ways, though; it is easy to think of entities that occur in three (or more) ways (e.g. verbs $\times$ subjects $\times$ direct objects). We will present the mathematical machinery to deal with multi-way co-occurrences, and test the usefulness of three-way methods in an application, namely the induction of selectional preferences.

The outline of this dissertation is as follows. The first part provides a theoretic framework – grounding the distributional similarity theorem and setting up a formal framework for a computational implementation. This includes a thorough overview of two dimensionality reduction algorithms – singular value decomposition and non-negative matrix factorization – and a discussion of three-way methods. The second part investigates the three different groups of models and their various model parameters, and provides a quantitative evaluation of their ability to extract semantic similarity. The final part of this dissertation provides a number of applications, in which the different models and techniques presented in the first part are applied.

Part I

Theory

Chapter 1

The Nature of Meaning

If we want to extract meaning automatically from text, we first need to investigate what 'meaning' actually is. What does it mean for a word to mean something? How is a particular word able to convey a particular content? And how is this content built up? What is, in short, the nature of meaning?

1.1 Theories of meaning

In the course of history, the nature of meaning has been one of the major issues in the philosophical debate. The issue was first raised in the ancient Greek world, and was subsequently tackled by numerous philosophers. In the 19th century, meaning also entered the realm of linguistics – first in the context of diachronic linguistics,[1] later also as a synchronic study. In the following paragraphs, we briefly discuss the different theories of meaning (and their relation to reality) that have been proposed both in philosophy and linguistics, and assess their potential to serve within computationally implemented procedures of meaning extraction.

1.1.1 Referential theory of meaning

In a referential theory of meaning, the meaning of a particular word is regarded as a pointer to the designated object in the real world. The meaning of a word is what it refers to. If we utter a word like *apple*, we refer to an actual apple (or the set of all apples) in reality. Intuitively, a referential theory of meaning seems very appealing.

[1] Christian Karl Reisig proposed the study of 'Semasiologie' in 1825, Michel Bréal coined the term 'sémantique' in 1883.

If parents want to teach their children the meaning of a word like *apple*, chances are pretty high that they will point to an actual apple – or a picture of one. At first sight words indeed seem no more than references to things (entities, actions or relations) existing in the outside world. There are, however, a number of problems with such a referential theory of meaning. The theory is able to account for what is generally called the *denotation* or *extension* of words, but fails to describe other semantic characteristics, generally referred to as *connotation* or *intension*. The German philosopher Gottlob Frege (1848–1925) illustrated this deficiency with a by now well-known example. Compare the following sentences:

(1) The morning star is the morning star.

(2) The morning star is the evening star.

Both *morning star* and *evening star* refer to the same entity, viz. the planet Venus, which might be visible either in the morning or in the evening (depending on the relative position of Venus and the earth). Sentences (1) and (2), however, significantly differ in meaning. Sentence (1) expresses a simple tautology, whereas sentence (2) expresses a new and important astronomical truth. Sentences (1) and (2) do not mean the same thing, but a referential theory of meaning does not account for the difference between them.

Frege's solution to the morning/evening star paradox was to make a distinction between *Sinn* (sense) and *Bedeutung* (reference). *Bedeutung* is the object that the word refers to, whereas *Sinn* is the cognitive representation of the object. By making this distinction, it is possible for words to have a different sense but the same referent (as in the paradox above).

The referential theory of meaning has been popular with logicians (e.g. the young Wittgenstein and Bertrand Russell). It provides a parsimonious and straightforward model of meaning, but the previous examples have shown that it is incapable of capturing all aspects of meaning. Moreover, it is unclear how we ought to proceed in order to extract these 'meaning references' in a computational way. The theoretical problems as well as the practical drawbacks make the referential theory rather unattractive for the computational extraction of meaning.

1.1.2 Mentalist theory of meaning

Another solution – one that has been very popular throughout the history of philosophy, starting with the Greek philosopher Plato – is to represent meaning exclusively as ideas. A mentalist theory of meaning associates the meaning of a particular word with a particular idea in the human mind. This theory effectively solves the morning/evening

star paradox: The morning star might be the same thing as the evening star in reality, but the *idea* of the morning star and the evening star may very well differ. The question that immediately follows is what this notion of *idea* actually entails. Surely, it cannot be the mental representations that are present in each individual person. These mental representations differ a lot among different persons. If one person hears the word *strawberry*, an image of an appetizing dessert plate – possibly covered with lots of whipped cream – might pop up. Another person might prefer them with powder sugar, and another one without any topping at all. Or one might even be disgusted by the idea of strawberries, because of a severe allergic reaction in the past. To be practically usable, the *ideas* need to have some generality, exceeding the individual level. But it is difficult to achieve this generalization without resorting to the notion of idea in the platonic sense, that is somehow mysteriously present in people's minds. This is not the direction we want to venture into, especially if we want to implement semantics in a computational way. If we want a sound theory of semantics that can be implemented computationally, we will need a theory that is not dependent on reference or ideas.

1.1.3 Behavioural theory of meaning

The vagueness and non-generality that inevitably seems to surround the mentalist view has led people to abandon the mentalist theory of meaning in favour of a theory that sticks to 'observable' facts. Inspired by the behaviourist movement that became popular within the field of psychology, the American linguist Leonard Bloomfield defines meaning of a linguistic form as 'the situation in which the speaker utters it and the response which it calls forth in the hearer'. (Bloomfield, 1933, p. 139). The meaning of a word is thus reduced to the speaker's stimulus that elicits its use, and/or the hearer's response to that word.

Although the behavioural theory of meaning claims to overcome the vagueness of ideas in the mentalist view, it seems almost as problematic as the theory it opposes. There is a plethora of different stimuli that elicit the same word, and the number of different responses evoked by that word is equally high. Take, for example, a word like *jazz*. In some situations, a person might utter the word to indicate they would like to hear some jazz tunes. In other situations, they might utter the word to approve – or disapprove – of the music they are listening to at that moment. And one odd person – not particularly familiar with different music styles – might even utter *jazz* when in fact they are listening to hip hop. Similarly, people's reactions to the word *jazz* may differ quite a lot. One person might turn on the radio and look for a suitable radio station, another one might start nodding their head whistling a Duke Ellington tune, while yet another might make an unhappy face and stick out their tongue. Every language utterance has a similar abundance of stimuli eliciting it, and a similar abundance of

responses following it. This makes it practically impossible to describe the meaning of a particular word in terms of the utterance's stimuli and responses.

Moreover, behaviourists have a rather vague and untenable view of what this behaviourist meaning description practically should look like. In his main textbook on linguistics, *Language*, Bloomfield notes:

> The situations which prompt people to utter speech, include every object and happening in their universe. In order to give a scientifically accurate definition of meaning for every form of a language, we should have to have a scientifically accurate knowledge of everything in the speaker's world. (Bloomfield, 1933, p. 139)

Bloomfield himself acknowledges that

> ... the statement of meanings is therefore the weak point in [behavioural] language-study, and will remain so until human knowledge advances very far beyond its present state. (Bloomfield, 1933, p. 140)

Bloomfield also deems it necessary to 'resort to makeshift theories' whenever scientific description is impossible – one of those theories being a referential theory of meaning.

In addition to the theoretical and practical drawbacks associated with a behaviourist's description of meaning, the theory obviously doesn't stand a chance to function within a computational framework. In order to implement meaning in a behavourial, computational framework, a computer should be able to observe, interpret and classify human stimuli and responses. The current state of artificial intelligence does not allow such complex cognitive computations just yet.

1.1.4 Use theory of meaning

A radically different theory of meaning qualifies the meaning of an expression as its use in a language system. A use theory of meaning does not refer to an external entity (a referent, an idea, or stimuli and responses) to qualify a word's meaning, but instead qualifies the meaning of a word as the value it gets through the (linguistic) system in which it is used. It was Wittgenstein who famously noted that 'the meaning of a word is its use in the language' (Wittgenstein, 1953).

The use theory of meaning differs radically from the previous theories of meaning. In the previous theories, there is an existing order of things (a 'meaning') outside of the language system; the words of a language are used to talk about existing entities, but entities and words belong to two different classes, and there is no influence between

the two classes. A use theory of meaning, on the other hand, advances a system in which meaning is defined and constructed within the language itself.

The first person to explore this radically different view in the context of linguistic theory was the Swiss linguist Ferdinand de Saussure (1857–1913), who is the founding father of the linguistic movement nowadays known as structuralism. In his most important work, the *Cours de Linguistique Générale (Course in General Linguistics)*, Saussure lays out the foundations for a differential view on language. Saussure defines a linguistic sign as a combination of the *signifiant* ('signifier') – representing the sound form of the sign – and the *signifié* ('signified') – representing the linguistic meaning of the sign.

According to Saussure,

> . . . la langue est un système dont tous les termes sont solidaires et où la valeur de l'un ne résulte que de la présence simultanée des autres . . .
> [language is a system in which all terms are equal, and in which the value of one is only the result of the simultaneous presence of the others]
> (Saussure, 1916)

This quote represents Saussure's structuralist view on language: the linguistic meaning of a sign is not a given, existing truth in the outside world, but it is defined in terms of its use in particular contexts (and its non-use in other contexts). The meaning of a particular word is not an independent or transcendental fact, but it is defined within a network of different embedded meanings, which in turn get their values from their position in the network of meanings.

The structuralist view on language has been further developed by a number of linguists, one of the most notable being Zellig Harris. Harris advocated a distributional method for linguistic research: linguistic elements (words, but also morphemes or phonemes) can be investigated by looking at the way they are distributed in language. As such, the distributional method is also able to discover the semantic properties of a word. Harris notes:

> 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. (Harris, 1954, p. 156)

The hypothesis that semantically similar words tend to occur in similar contexts has been coined the DISTRIBUTIONAL HYPOTHESIS in subsequent work, and Harris' work is often cited as the main source of inspiration. The distributional hypothesis not only turns out to provide a sound basis for meaning description, it also provides a suitable starting point for an implementation in a computational framework, as will be shown in chapter 2.

1.2 Context

In the first part of this chapter, we have shown that a use theory of meaning is a sound basis for the computational extraction of lexico-semantic meaning: the sum of a word's contexts is a good indicator of the word's use, and hence its 'meaning'. In the second part of this chapter, we focus on the notion of context. The context of a particular word can be interpreted in a number of ways: the context might be the document the word appears in, it might be a window of words around a particular word, or it might be the syntactic context in which the word takes part. In this section, we will have a look at these different kinds of context, investigate which parameters are involved, and examine how different contexts might be useful for lexico-semantic knowledge extraction. The next chapter will then investigate how these various contexts can be formalized and implemented in a computational way.

1.2.1 Document-based context

First of all, a particular word always appears in a particular document. This gives rise to our first instantiation of the distributional hypothesis:

Hypothesis 1. *Words are semantically similar if they appear in similar documents.*

Words that appear in the same documents tend to be thematically related: texts usually focus on one particular topic (or a few topics), so that the majority of content words is related to these topics. Take for example the three newspaper paragraphs in figure 1.1, taken from the MEDIARGUS newspaper corpus, a 1.4 billion word corpus of (Belgian) Dutch newspaper texts.

The three paragraphs all contain words related to the medical domain (printed in boldface). Note that a word like *patiënt* 'patient' appears in all three documents. Likewise, words like *dokter* 'doctor' and *arts* 'doctor' appear in the same documents. In a similar vein, words related to another topic – say economics, soccer, or rock music – appear together in the same documents. If such related words appear in the same documents sufficiently frequently, a computer algorithm might be able to infer that they are indeed semantically related.

Uit het onderzoek blijkt ook dat slechts de helft van de **patiënten** naar de **dokter** gaat. Veertig procent praat over z'n probleem met vrienden. Maar 20 procent van de **patiënten** heeft het er met niemand over. **Patiënten** die hun **kwaal** voor de buitenwereld verbergen zeggen 'de juiste woorden niet te vinden om hun toestand te beschrijven', of 'zich te schamen over hun toestand'. Sommigen hadden zelfs schrik om er met iemand over te praten. Volgens het onderzoek dient de reden waarom mensen er niet over praten ook bij de **arts** te worden gezocht.

The research also shows that only half of the patients goes to the doctor. Forty percent talks about their problem with friends. But twenty percent of the patients does not talk to anybody. Patients hiding their condition for the outside world claim 'not to find the right words to describe their situation', or 'to be ashamed of their situation'. Some were even frightened to talk to someone about it. The research shows that the reason why people are not talking about it also has to be sought with the doctor.

In heel wat gevallen kunnen **dokters** zich beter concentreren op de oorzaken van de **pijn** in plaats van op de **behandeling**, zo wil de nieuwe denktrant in de **medische** wereld. **Operaties** zijn uitzonderlijk, het gros van de **patiënten** is gebaat bij de zogenaamde conservatieve (**niet-operatieve**) **therapieën**. De topper is **oefentherapie** onder begeleiding van een **kinesitherapeut**.

In many cases, doctors would better concentrate on the causes of the pain instead of the treatment, that is the new way of thinking in the medical world. Surgeries are exceptional, the majority of the patients benefits from so-called conservative (non-surgical) therapies. The top therapy is training therapy, coached by a physiotherapist.

Zaterdagvoormiddag werd ingebroken in de wagen van een **arts** die op huisbezoek was bij een **patiënt**. De dader sloeg een ruit van de wagen stuk, vond de **doktersjas** en nam een aantal **spuiten** mee.

Saturday morning, the car of a doctor visiting a patient was burgled. The offender broke a window, found the doctor's coat and took a couple of injections.

Figure 1.1: Three document paragraphs from different newspapers – all containing words from the medical domain – extracted from the MEDIARGUS corpus

The main parameter to be set is the size of the document context. This will depend on the corpus used and the application in mind. In a newspaper corpus, the unit might be an article, or a paragraph. When using a web corpus, one might consider a particular web page as document context.

1.2.2 Window-based context

Secondly, a particular word appears within the context of other words in its vicinity, which brings us to our second instantiation of the distributional hypothesis:

Hypothesis 2. *Words are semantically similar if they appear within similar context windows.*

Below are some examples taken from the TWENTE NIEUWS CORPUS (TWNC), a 500M word corpus of Dutch newspaper texts. Examples (3) to (5) all contain the word *courgette* 'zucchini'. Examples (6) to (8) all contain the word *aubergine* 'eggplant'.

(3) Kies eens voor tomaat, paprika, dun geschaafde **courgette** en plakjes
choose once for tomato pepper thin sliced zucchini and slices
rauwe champignons.
raw mushrooms
Pick a tomato, pepper, thinly sliced zucchini and slices of raw mushrooms for once.

(4) Serveer met pasta en gebakken groente, zoals paprika, **courgette** en
serve with pasta and fried vegetable like pepper zucchini and
tomaat.
tomato
Serve with pasta and fried vegetables, such as pepper, zucchini and tomato.

(5) Deze Indiase currysoep (mulligatawny) krijgt een zomers tintje door
this Indian curry soup mulligatawny gets a summery touch through
de **courgette** en paprika.
the zucchini and pepper
This Indian curry soup (mulligatawny) gets a summery touch because of the zucchini and pepper.

(6) Snijd groenten, zoals paprika, **aubergine** en ui in kleine stukjes.
cut vegetables like pepper eggplant and onion in small pieces
Cut vegetables, such as pepper, eggplant and onion in small pieces.

(7) Natuurlijk, in Purmerend verkopen ze ook tomaten, en **aubergine**, en
of course in Purmerend sell they also tomatoes and eggplant and
paprika.
pepper
Of course, tomatoes, eggplants and peppers are also sold in Purmerend.

(8) Voeg **aubergine**, aardappelen, bataat (zoete aardappel) en paprika
add$_{verb}$ eggplant potatoes bataat sweet potato and pepper
toe.
add$_{particle}$
Add eggplant, potatoes, bataat (sweet potato) and peppers.

Note that *courgette* and *aubergine* in the examples above have a tendency to occur with the same words, such as *paprika* 'pepper', *tomaat* 'tomato', and *groente* 'vegetable'. Again, if such related words (like *courgette* and *aubergine*) occur with the same words (like *paprika* and *tomaat*) sufficiently frequently, a computer algorithm might be able to infer that they are indeed semantically related. Note that *courgette* and *aubergine* even do not have to occur together (although they might). It is their co-occurrence with other words that is indicative of their semantic relatedness.

A simple context window as described above is often called a BAG OF WORDS context. This expression is used to indicate the fact that no order (or syntax) is taken into account; the ordered words are mixed together ('put together in one bag') so that their internal order is lost.

The main parameter to be set is the size of the window in which a word's context words occur. One might take into account a small window, in which only the left and right co-occurring word are used as context. A medium-sized window might use two or five words to the left and right of the word in question. A large window might take into account all context words that occur in the same sentence, or even in the same paragraph.

One can imagine that different context window sizes will lead to different kinds of semantic similarity. When using a small context window, an algorithm might be able to find tight semantic relationships: in a small context window, more closely related context words might appear in the word's vicinity, and the algorithm might even be able to discover some basic syntactic facts (e.g. the fact that a particular word appears with an article). When using a larger context window, more loosely related words might show up, and all order gets lost in the bag of words. Using larger windows, the algorithm might be more likely to discover topically similar words again. In the second part of this thesis, we will investigate whether this hypothesis is true.

1.2.3 Syntax-based context

Thirdly, a particular word always takes part in particular syntactic relations. This gives rise to our third and final instantiation of the distributional hypothesis:

Hypothesis 3. *Words are semantically similar if they appear in similar syntactic contexts.*

In this research, syntactic context will be instantiated in the form of dependency graphs; dependency graphs provide a theory-neutral instantiation of a sentence's syntax, since no particular grammatical framework is assumed. More specifically, this research will use dependency structures that conform to the guidelines for the *Corpus Spoken Dutch* (CGN, Hoekstra et al. (2001)). The dependency structures used in the CGN syntactic annotation have developed into a de facto standard for the computational analysis of Dutch (Bouma, van Noord, and Malouf, 2001) and they are used as output format of the Dutch dependency parser ALPINO (van Noord, 2006). Formally, a CGN dependency structure $D = \langle V, E \rangle$ is a labeled directed acyclic graph, with node labels V representing the categories (phrasal labels and POS labels) and edge labels E representing the dependency relations.

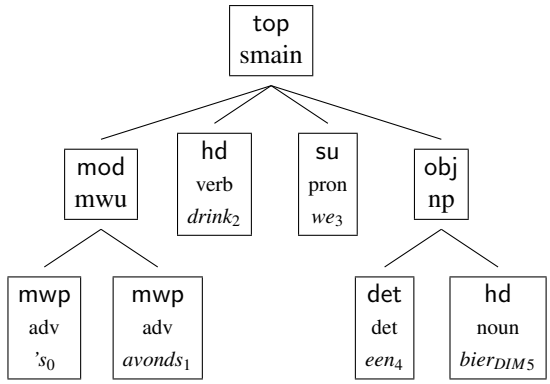

Figure 1.2: Dependency structure for the sentence *'s avonds drinken we een biertje* ('in the evening we'll drink a beer')

Figures 1.2 and 1.3 show dependency structures for two sentences from the MEDIARGUS corpus, parsed with ALPINO. Table 1.1 shows the set of dependencies that can be deduced from the structures. In our syntax-based models of semantic similarity (discussed in the next chapter), we will use these dependency triples as the input data.

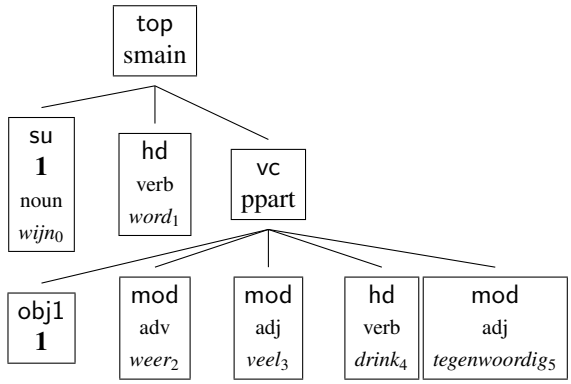

Figure 1.3: Dependency structure for the sentence *wijn wordt weer veel gedronken tegenwoordig* ('wine is drunk a lot again today')

⟨drink	mod	's avonds	⟩
⟨drink	su	we	⟩
⟨drink	obj1	bier_{DIM}	⟩
⟨word	su	wijn	⟩
⟨drink	obj1	wijn	⟩
⟨drink	mod	weer	⟩
⟨drink	mod	veel	⟩
⟨drink	mod	tegenwoordig	⟩

Table 1.1: The set of dependency triples extracted from the two parses in figures 1.2 and 1.3

Note that both *biertje*[2] and *wijn* appear as direct object of the verb *drink*. Again, if we look at a large number of sentences, we might notice that *biertje* and *wijn* appear in similar syntactic contexts.

One important parameter in syntax-based models is the set of dependency relations that will be incorporated into the model. A number of dependency relations that might be useful in distributional similarity models are given in table 1.2. These are the dependency relations that will be used in the syntax-based models presented in this thesis.

[2] DIM indicates the word is a diminutive form.

abbr.	relation	example
SU	subject	⟨*author*, SU, *write*⟩
OBJ1	direct object	⟨*wine*, OBJ1, *drink*⟩
OBJ2	indirect object	⟨*him*, OBJ2, *give*⟩
PC	prepositional complement	⟨*dog*, PC, *look_after*⟩
MOD	modifier	⟨*red*, MOD, *apple*⟩
PREDC	predicative complement	⟨*apple*, PREDC, *tasty*⟩
COO	coordination	⟨*apple*, COO, *pear*⟩
APP	apposition	⟨*London*, APP, *city*⟩

Table 1.2: Dependency relations used as contexts

1.3 Tight vs. topical similarity

We already briefly mentioned the difference between tight, synonym-like semantic similarity and more loosely related, topical similarity. With tight similarity, we indicate the fact that two words are very similar, i.e. there is a (near-)synonymous or (co-)hyponymous relationship between the two words. With topically similar words, we mean words that belong to the same semantic domain.

The example below makes clear the difference between both kind of similarities. Two sets of words are given that are semantically similar to the word *arts* 'doctor'. The first set contains words that are tightly similar to *arts*, containing synonyms (e.g. *dokter* 'doctor') and hyponyms (e.g. *chirurg* 'surgeon'). The second set of words is topically related to *arts*, containing words that all belong to the medical domain. The topically related words are related to the target word by more loose relationships, such as association and meronymy (part-whole relationships).

1. *dokter* 'doctor', *medicus* 'doctor', *huisarts* 'family doctor', *chirurg* 'surgeon', *specialist* 'specialist', *gynaecoloog* 'gynaecologist'

2. *patiënt* 'patient', *ziekte* 'disease', *diagnose* 'diagnosis, *behandeling* 'treatment, *ziekenhuis* 'hospital', *stethoscoop* 'stethoscope'

In the evaluation part (the second part of this thesis), we will not only try to evaluate the performance of the various models for the extraction of semantic similarity; we will also try to determine the nature of the similarity, i.e. whether the models are extracting tight, synonym-like similarity or more loosely related, topical similarity.

Chapter 2

The Computation of Meaning

In the last chapter, we made clear that the context of a particular word is able to suitably inform us about its semantics, and we looked at the various contexts that might be useful for the induction of semantic similarity. In this chapter, we will investigate how this notion of context can be formally implemented in a computational framework.

2.1 Formal model

The last chapter provides an intuitive idea of how the context of a word might be used to calculate its semantic similarity to other words. Now, in order to implement this idea in a computational framework, it needs to be expressed in more formal terms. In this section, we will have a look at some existing literature that stipulates semantic space models and the notion of context in more formal terms.

Lowe (2001) provides a formal definition of a semantic space model; he defines the model as a quadruple $\langle A, B, S, M \rangle$. B is a set of basic elements ($b_1 \ldots b_D$) determining the dimensionality D of the vector space and the interpretation of each dimension. B might be a set of documents, words, or dependency relations, depending on the context that is used. A specifies the function that maps the standard co-occurrence frequencies of basis elements and words to their final value, so that each word is represented by a vector $v = [A(b_1,t), A(b_2,t), \ldots, A(b_D,t)]$. A may be the identity function (so that the final vector contains simple co-occurrence counts), but often a more advanced mapping is used. A is called the lexical association function or weighting function. The weighting function is discussed in section 2.3. S is a similarity measure that maps pairs of vectors onto a real number that represents semantic similarity. Different similarity measures are discussed in 2.2. Finally, M is a mathematical transformation that takes

one semantic space and maps it onto another, e.g. by reducing its dimensionality. M may be an identity mapping, so that the original space remains unchanged, but often a mathematical transformation proves beneficial for countering data sparseness and reducing noise. Various dimensionality reductions are discussed in chapter 3.

Pado & Lapata (2007) extend Lowe's framework for constructing semantic space models based on syntax (dependency parses). Dependency parses are interpreted as directed graph structures of nodes and labeled edges. A particular dependency path π for a particular target word t can then be represented as an ordered set of tuples according to the dependency graph (ensuring connectedness and cycle-freeness).

In their framework, a semantic space is a tuple $\langle B, T, M, S, A, cont, \mu, v \rangle$. B is the set of basis elements, T is the set of target words, M is the matrix $M = B \times T$, A is the lexical association function, and S is the similarity measure. These parameters do not differ significantly from Lowe's model. The additional parameters used are the content selection function $cont : T \rightarrow 2^{\pi}$, the basis mapping function $\mu : \pi \rightarrow B$, and the path value function $v : \pi \rightarrow \mathbb{R}$.

The context selection function *cont* allows us to select certain paths in the graph that contribute to the context of a particular target word. This function allows us to select only paths with a particular length, or paths that are labeled with a particular dependency relation. The basis mapping function μ maps paths onto basis elements. This way, the dependency paths are decoupled from their representation in the final semantic space. Such decoupling allows, for example, to use words instead of syntactic features in the final representation. The path value function v is used to assign weights to particular paths. E.g., longer dependency paths might be given less weight; another possibility is to give more weight to particular dependency relations.

2.2 Similarity calculations: geometry vs. probability

Semantic similarity can be implemented in two different – albeit related – ways: in a (geometrically oriented) vector space model or in a (statistically oriented) probability distribution model. Both models are instantiations of the formal model described above. We will discuss the former in section 2.2.1 and the latter in section 2.2.2. Different similarity measures S are presented for both models.

2.2.1 Vector space model

In a semantic vector space model, each word in a language is mapped to a point in a real finite dimensional vector space. The vector space model is one of the most widely used models for the acquisition of semantic similarity. The model makes it possible to

express 'semantic proximity' between entities in terms of spatial distance. In a vector space model, particular entities (words, for example) are represented as vectors of features (the word's different contexts) in a multi-dimensional Euclidean space. By applying a suitable similarity measure (cfr. infra), one can straightforwardly calculate the similarity between the different entities.

The vector space model was first developed in the context of information retrieval (Salton, Wong, and Yang, 1975), representing documents and queries as vectors of the words they contain. The documents that are the closest to a particular query in this vector space (i.e. the documents that are using the same words as in the query) will most likely represent the documents that the user was looking for.

This model can straightforwardly be applied to similarity calculations between words. The two words for which the semantic similarity is to be calculated, are represented as vectors of the words' various contexts. Figure 2.1 shows an example matrix *M* containing vectors for four different target words (using dependency relations as features). In this example the set of target words is

$$T = \{apple, banana, car, truck\}$$

and the set of basic elements is

$$B = \{red_{adj}, yellow_{adj}, tasty_{adj}, fast_{adj}, eat_{obj}, drive_{obj}\}$$

	red_{adj}	yellow_{adj}	tasty_{adj}	fast_{adj}	eat_{obj}	drive_{obj}
apple	200	24	129	0	289	0
banana	1	152	87	1	214	1
car	120	74	0	98	1	386
truck	67	44	0	37	0	175

Figure 2.1: A noun-by-features matrix

The value in matrix cell (i, j) is the co-occurrence frequency of word i with value j. In the example above, the adjective *red* appears 200 times with the word *apple*, and the word *car* appears 386 times as the object of *drive*.

To facilitate computations, vectors are often normalized to vector length of 1. The vector length or *norm* of a vector $\overrightarrow{v}$ with length k is calculated with equation 2.1.

$$|\overrightarrow{v}| = \sqrt{\sum_{i=1}^{k} v_i^2} \tag{2.1}$$

Dividing a vector by its vector length normalizes it to a vector length of 1. When normalizing the vectors in figure 2.1, the resulting matrix looks like the one in figure 2.2.

	red_{adj}	yellow_{adj}	tasty_{adj}	fast_{adj}	eat_{obj}	drive_{obj}
apple	.533	.064	.344	.000	.770	.000
banana	.004	.550	.315	.004	.774	.004
car	.284	.175	.000	.232	.002	.914
truck	.342	.224	.000	.189	.000	.893

Figure 2.2: A noun-by-features matrix normalized by vector length

In order to calculate the contextual overlap between two vectors $\overrightarrow{v}$ and $\overrightarrow{w}$ (which – as has been described in the previous chapter – we think of as a good predictor for semantic similarity), we need a proper vector similarity measure $S = sim(\overrightarrow{v}, \overrightarrow{w})$.

The two simplest measures for vector similarity are the Manhattan distance and the Euclidean distance. The Manhattan distance or L_1 norm is defined as

$$dist_{\text{MANHATTAN}}(\overrightarrow{v}, \overrightarrow{w}) = \sum_{i=1}^{k} |v_i - w_i| \tag{2.2}$$

and the Euclidean distance, or L_2 norm, is defined as

$$dist_{\text{EUCLIDEAN}}(\overrightarrow{v}, \overrightarrow{w}) = \sqrt{\sum_{i=1}^{k} (v_i - w_i)^2} \tag{2.3}$$

Both distance measures are intuitively easy to understand, and provide a straightforward extension of semantic similarity calculations in terms of spatial distance. In practice, though, neither the Manhattan distance nor the Euclidean distance are frequently used as word similarity measures. Both measures are very sensitive to extreme values – which often occur with frequency counts – even after normalization.

Two other similarity measures that do show up in word similarity calculations – Jaccard and Dice – are derived from set theory. Originally, they were designed for binary vectors, but they can easily be extended in order to deal with frequency data.

The Jaccard similarity measure is defined as

$$sim_{\text{JACCARD}}(\overrightarrow{v}, \overrightarrow{w}) = \frac{\sum_{i=1}^{k} \min(v_i, w_i)}{\sum_{i=1}^{k} \max(v_i, w_i)} \tag{2.4}$$

and the Dice similarity measure is defined as

$$sim_{\text{DICE}}(\overrightarrow{v}, \overrightarrow{w}) = \frac{2 \times \sum_{i=1}^{k} \min(v_i, w_i)}{\sum_{i=1}^{k} (v_i + w_i)} \tag{2.5}$$

Intuitively, both measures calculate the weight of overlapping features (the numerator with the min function) compared to the total feature weight (the denominator, either using the max function for the Jaccard measure or the sum of both vectors' feature values for the Dice measure).

The best known and most widely used similarity measure, however, is the cosine similarity measure. The cosine similarity is easy to compute, and it often achieves the best results. It has therefore become the best known and most widely used vector space similarity measure. The cosine similarity measure is calculated as

$$cos(\overrightarrow{v}, \overrightarrow{w}) = \frac{\overrightarrow{v} \cdot \overrightarrow{w}}{|\overrightarrow{v}| \, |\overrightarrow{w}|} \tag{2.6}$$

where $\overrightarrow{v} \cdot \overrightarrow{w}$ is the dot product between vector $\overrightarrow{v}$ and $\overrightarrow{w}$, both of length k

$$\overrightarrow{v} \cdot \overrightarrow{w} = \sum_{i=1}^{k} v_i w_i \tag{2.7}$$

Note that, when both vectors $\overrightarrow{v}$ and $\overrightarrow{w}$ are normalized to unit length, the denominator is redundant, so that the cosine similarity amounts to a simple dot product between two vectors.

Once we have defined the similarity measure S, we can calculate the similarity between the different word vectors. The resulting calculation yields the similarity matrix S_{sim} of size $n \times n$, where n is the number of target words T. The similarity matrix is represented in figure 2.3.

	apple	banana	car	truck
apple	1.000	.741	.164	.197
banana	.741	1.000	.103	.129
car	.164	.103	1.000	.996
truck	.197	.129	.996	1.000

Figure 2.3: A word by word similarity matrix

2.2.2 Probabilistic model

The vector space model is the oldest, best known and most widely used model for semantic similarity, but it is not the only one. A word's contextual information can also be captured in a statistically oriented probability distribution model. Probability distribution models allow for the use of well-known information-theoretic measures of similarity, and they offer the possibility of implementing semantic similarity in a Bayesian framework.

The probabilistic model of semantic similarity looks similar to the vector space model of semantic similarity, but its underpinnings are different. In a probabilistic semantic similarity model, a word's context is represented as a proper probability distribution, obeying the laws of probability. Each feature on a word's vector represents the probability $p(f|w)$, the probability of the feature given the word. This means that the original frequency matrix is normalized, so that each vector sums to 1, i.e. each feature value is divided by the sum of the vector's feature values. If applied to the matrix in figure 2.1, this gives the matrix in figure 2.4.

	red_{adj}	yellow_{adj}	tasty_{adj}	fast_{adj}	eat_{obj}	drive_{obj}
apple	.312	.037	.201	.000	.450	.000
banana	.002	.333	.191	.002	.469	.002
car	.177	.109	.000	.144	.001	.568
truck	.207	.136	.000	.115	.000	.542

Figure 2.4: A noun-by-features matrix normalized to probability $p(f|w)$

A number of similarity measures $S = sim(\overrightarrow{v}, \overrightarrow{w})$ are available to calculate the similarity between two probability vectors; the best known measure to calculate the similarity between probability distributions is the Kullback-Leibler (KL) divergence, which is defined as:

$$D_{\text{KL}}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{2.8}$$

The KL divergence measures how well probability distribution Q approximates probability distribution P; it tells us how much information we lose if we encode data with Q when P is the actual probability distribution.

There are, however, a number of problems with the KL divergence. First of all, it is undefined if there is a dimension i with $Q(i) = 0$ and $P(i) \neq 0$. Secondly, the KL divergence is an asymmetric distribution, which means that $D_{\text{KL}}(P \parallel Q) \neq D_{\text{KL}}(Q \parallel P)$.

There are other similarity measures that overcome these problems. The first one is the Jensen-Shannon (JS) divergence. The JS divergence is defined as:

$$D_{\mathrm{JS}}(P \parallel Q) = \frac{1}{2} D_{\mathrm{KL}}(P \parallel \frac{P+Q}{2}) + \frac{1}{2} D_{\mathrm{KL}}(Q \parallel \frac{P+Q}{2}) \tag{2.9}$$

Intuitively, the JS divergence tells us how much information is lost if the two probability distributions P and Q are replaced by the average of both distributions. The JS divergence does not have any problems with infinite values, and it is symmetric.

Another possibility is to approximate the KL divergence as close as possible by mixing it to a small degree with the other distribution. This is what the skew divergence does. The skew divergence is defined as:

$$D_{skew(\alpha)}(P,Q) = D_{\mathrm{KL}}(P \parallel \alpha Q + (1-\alpha)P) \tag{2.10}$$

The skew divergence constant α is a number between 0 and 1, usually set close to 1 to approximate the KL divergence as close as possible; a normal value of $\alpha = 0.99$. The measure remains asymmetric, but mixing in the other probability distribution to a small degree, effectively solves the infinity problem with zero values.

2.3 Weighting schemes

The methods described above can be applied to raw frequency counts. Often, though, an extra weighting step is applied in order to adapt the feature value according to its actual importance. In our formal model, this is the lexical association function or weighting function A. Many different weighting functions have been applied to the problem of semantic similarity. In the following paragraphs, we will have a look at the intuition behind them, and investigate the different possibilities.

2.3.1 Introduction: Zipf's law

Zipf's law states that the frequency of a word in any particular corpus is inversely proportional to its rank in a frequency list. As a result, word distributions are extremely skewed: the majority of words occur very infrequently, whereas the top few most frequent words take up the largest part of the corpus. This fact brings about a frequency bias: words with similar frequencies will be considered more similar than they actually are.

Intuitively, it is more significant for a word's semantics to appear with an infrequent but highly specific, 'meaningful' feature than to appear with a very frequent, broad, 'meaningless' feature. As an example, compare *denim skirt* with *nice skirt*. It seems

reasonable to attach more weight to the first feature *denim* than to the second feature *nice*. The former feature is highly specific and appears with a small subset of words (like *denim pants*, *denim jeans*, *a denim jacket*), whereas the latter is more broad and unspecific, and appears with a much larger set of words (*a nice girl*, *a nice feeling*, *a nice zebra*, ...). Moreover, a co-occurrence like *denim skirt* is much more informative than e.g. *a skirt*, although the latter one will have a much higher frequency. By applying a suitable weighting, we can neutralize the skewed frequencies arising from the Zipfian distribution.

Weighting functions can be divided into local and global weighting functions, according to the information they use in order to calculate the weighted value; local weighting functions only use a particular co-occurrence frequency count on its own to calculate the weighted value, whereas global weighting functions make use of global word and feature distribution statistics calculated over the corpus as a whole. In the following paragraphs, we will have a look at both types, and discuss their most important instantiations in the scope of semantic similarity.

2.3.2 Local weighting

A local weighting function is a function that is applied to a particular co-occurrence frequency without any knowledge about the corpus frequencies as a whole. In the simplest case, this amounts to applying the identity function to a particular co-occurrence frequency. Another simple local weighting is the application of a binary function, which assigns a value of one if the co-occurrence frequency is larger than zero (i.e. the combination occurs at least once in the corpus) and zero otherwise.

A local weighting function that is often used in the scope of semantic similarity is a logarithmic weighting function. A logarithmic weighting dampens large frequency values, so that frequent words are assigned less extreme values. The base of the logarithm is often taken to be natural, though in practice the algorithm's base does not influence the results.

$$A_{log}(f_{ij}) = 1 + \log(f_{ij}) \tag{2.11}$$

for $f_{i,j} > 0$. The function is represented graphically in figure 2.5.

2.3.3 Global weighting

Entropy

In information theory, entropy is a measure to express the uncertainty (or surprise) that is associated with a random variable. Intuitively, if a particular word appears with only

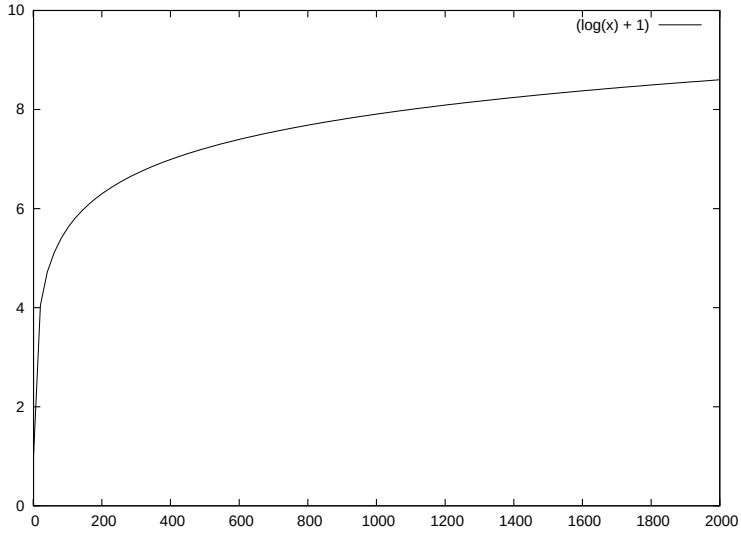

Figure 2.5: A logarithmic function is used to smooth extreme frequency values

a few features (documents), it will be much more informative than a word that appears with lots of features (documents). Words that appear in many documents (i.e. words that are more uniformly distributed) will have a high entropy value. Words that appear in only a limited number of documents, on the other hand, will have a low entropy value.

Formally, entropy weighting is usually calculated according to the formula in 2.12.

$$A_{ent}(i,j) = f_{ij}(1 + \sum_{k=1}^{n} \frac{p_{ik}\log(p_{ik})}{log(n)}), p_{ik} = \frac{f_{ik}}{\sum_{l=1}^{n} f_{il}} \tag{2.12}$$

with f_{ij} being the original co-occurrence frequency and n the total number of features (documents). This formula actually calculates an entropy ratio $G(i) = 1 - \frac{H(d|i)}{H(d)}$, where $H(d)$ is the entropy of the uniform distribution of the documents, and $H(d \mid i)$ is the entropy of the conditional distribution given that the noun i appeared. The original co-occurrence frequency is then multiplied by this ratio.

We will evaluate entropy as a weighting function in the document-based models.

Pointwise mutual information

Another popular weighting function is called pointwise mutual information (PMI). PMI was first proposed by Church and Hanks (1990), and is based on the information-theoretic notion of mutual information. Mutual information measures the mutual dependence between to random variables X and Y. It is defined as

$$I(X,Y) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{2.13}$$

Pointwise mutual information – i.e. the mutual information for particular events – is defined as

$$I(i,j) = \log \frac{p(i,j)}{p(i)p(j)} \tag{2.14}$$

Intuitively, PMI tells us how much information a particular feature contains about a target word (and vice versa). PMI measures how often two events i and j occur, compared to the expected value if they were independent. The numerator gives the actual probability of the target word and feature occurring together, whereas the denominator contains the probability of the target word and feature occurring independently (multiplying the marginal probabilities). Thus, the ratio indicates how much more the target word and feature co-occur than we would expect by chance.

Although PMI is used a lot as weighting function, it is problematic for low frequency counts: the score will depend on the frequency of individual words. Thus, low-frequency co-occurrences will receive a higher score than high-frequency co-occurrences (all other things being equal). One solution to this problem is to use a particular cut-off (e.g. a co-occurrence frequency of at least 3). Another solution is the use of an extra weighting factor dependent on the frequency (Pantel and Lin, 2002).

We will evaluate PMI as a weighting function in the window-based and syntax-based models.

Chapter 3

Dimensionality Reduction

3.1 Introduction

In the previous chapters, semantic similarity calculations have been carried out using the words' original feature space, which usually contains a large number of highly correlated features. The goal of a dimensionality reduction – also called factorization – is to find a smaller number of uncorrelated or lowly correlated dimensions (factors). There are two reasons for applying such a transformation to the data:

- When the feature space is large, similarity calculations often become computationally expensive or even impossible. A dimensionality reduction reduces the feature space to a much smaller number of dimensions, so that computations become tractable again.
- A dimensionality reduction is able to discover latent structure present in the data. This way, a dimensionality reduction is able to generalize over individual data samples. By classifying the data according to the latent structure and not according to the individual features, a dimensionality reduction is able to overcome data sparseness and noise.

One of the most famous dimensionality reduction methods for text processing is latent semantic analysis (LSA). LSA allegedly finds 'latent semantic dimensions', according to which nouns and documents can be represented more efficiently. In the subsequent section, we will first have a look at LSA and its underlying singular value decomposition. Next, we will examine non-negative matrix factorization, an algorithm that overcomes some of the problems linked to LSA.

3.2 Latent semantic analysis

3.2.1 Introduction

Latent semantic analysis (Landauer and Dumais, 1997; Landauer, Foltz, and Laham, 1998) models the meaning of words and documents by projecting them into a vector space of reduced dimensionality; the reduced vector space is built up by applying singular value decomposition (SVD) – a well known linear algebraic method – to a simple term-by-document frequency matrix $\mathbf{A}$. The resulting lower dimensional matrix $\hat{\mathbf{A}}$ is the best possible fit in a least squares sense (minimization of the Frobenius norm; equation 3.1).

$$\arg\min_{\hat{\mathbf{A}}} \| \mathbf{A} - \hat{\mathbf{A}} \|_F \tag{3.1}$$

By enforcing a lower number of dimensions, the algorithm is forced to make generalizations over the simple frequency data. Co-occurring terms are mapped to the same dimensions; terms that do not co-occur are mapped to different dimensions.

In the next section, we have a closer look at the principles and mathematics behind SVD. Next, some example SVD's are provided in order to exemplify their generalization capacity. We conclude with a discussion of the drawbacks linked to LSA.

3.2.2 Singular value decomposition

While rooted in linear algebra, singular value decomposition has proven to be a useful tool in statistical applications. It is closely akin to statistical methods such as principal components analysis, and has been used as a versatile dimensionality reduction technique in different scientific fields, such as image recognition, signal processing (Deprettere, 1988), and information retrieval. SVD stems from a well known theorem in linear algebra: a rectangular matrix can be decomposed into three other matrices of specific forms, so that the product of these three matrices is equal to the original matrix:[1]

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times z}\, \Sigma_{z\times z}\, (\mathbf{V}_{n\times z})^T \tag{3.2}$$

where $z = min(m,n)$. A graphical representation of SVD (with $z = n$) is given in figure 3.1.

[1] The singular value decomposition that is presented here is called the 'thin' or 'reduced' SVD. In applications such as LSA, it is unusual to compute the full SVD; the reduced version is faster to compute and more economical in storage, and it provides sufficient information for statistical applications.

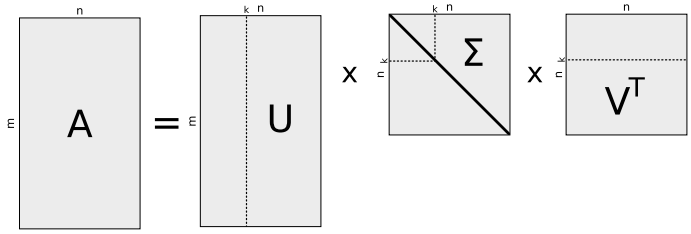

Figure 3.1: Graphical representation of SVD

Matrix **A** is the original matrix of size $m \times n$. Matrix **U** is an $m \times z$ matrix that contains newly derived vectors called left-singular vectors. Matrix $\mathbf{V}^T$ denotes the transpose of matrix **V**, an $n \times z$ matrix of derived vectors called right-singular vectors. The third matrix Σ is a $z \times z$ square diagonal matrix (i.e. a square matrix with non-zero entries only along the diagonal); Σ contains derived constants called singular values. A key property of the derived vectors is that all dimensions are orthogonal (i.e. linearly independent) to each other, so that each dimension is uncorrelated to the others.

The singular value decomposition can be interpreted as a method that rotates the axes of the n-dimensional space in such a way that the largest variation is captured by the leading dimensions. The diagonal matrix Σ contains the singular values sorted in descending order. Each singular value represents the amount of variance that is captured by a particular dimension. The left-singular and right-singular vector linked to the highest singular value represent the most important dimension in the data (i.e. the dimension that explains the most variance of the matrix); the singular vectors linked to the second highest value represent the second most important dimension (orthogonal to the first one), and so on. Typically, one uses only the first $k \ll z$ dimensions, stripping off the remaining singular values and singular vectors. If one or more of the least significant singular values are omitted, then the reconstructed matrix will be the best possible least-squares approximation of the original matrix in the lower dimensional space. Intuitively, SVD is able to transform the original matrix – with an abundance of overlapping dimensions – into a new, many times smaller matrix that is able to describe the data in terms of its principal components. Due to this dimension reduction, a more succinct and more general representation of the data is obtained. Redundancy is filtered out, and data sparseness is reduced.

The calculation of SVD involves iteratively solving a number of eigenvalue problems. A thorough understanding of the algorithm's computational details requires a firm background in linear algebra, and explaining all the mathematical nuts and bolts is well beyond the scope of this thesis. Suffice it to say that there are a number of

programs available that can handle the kind of large-scale singular value decompositions necessary for linguistic data sets. In this research, SVDPACK (Berry, 1992) has been used. SVDPACK is a program that is able to handle sparse matrices quickly and efficiently (depending on the number of singular values one wants to retain).

3.2.3 Examples

Consider two documents, one about *Belgium* (B) and one about the *Netherlands* (NL).

- Belgium is a kingdom in the middle of Europe, and **Brussels** is its capital. **Brussels** has a Dutch-speaking and a French-speaking university, but the largest student city is **Leuven**. **Leuven** has 31,000 students.
- The Netherlands is a country in Western Europe, located next to the North Sea. The Netherlands's capital is **Amsterdam**. **Amsterdam** has two universities. **Groningen** is another important student city. In **Groningen**, there are 37,000 students.

As we have seen in the previous chapter, these documents can easily be transformed into a term-document matrix, in which each document is represented by a column vector. Each element in the column vector corresponds to the frequency of a particular term (in this case cities) in the document. Similarly, each element on the row vector indicates how often a term appears in a particular document. The resulting matrix, together with its singular value decomposition, is given in figure 3.2.

$$\mathbf{A}\begin{bmatrix} & B & NL \\ Groningen & 0 & 2 \\ Leuven & 2 & 0 \\ Amsterdam & 0 & 2 \\ Brussel & 2 & 0 \end{bmatrix} = \mathbf{U}\begin{bmatrix} 0.00 & 0.71 \\ -0.71 & 0.00 \\ 0.00 & 0.71 \\ -0.71 & 0.00 \end{bmatrix} \Sigma \begin{bmatrix} 2.83 & 0 \\ 0 & 2.83 \end{bmatrix} \mathbf{V}^T \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Figure 3.2: Singular value decomposition of a term-document matrix

The original matrix $\mathbf{A}$ is decomposed into three other matrices $\mathbf{U}$, Σ and $\mathbf{V}^T$. The singular values in Σ show that two equally important dimensions are found; furthermore,

the left- and right-singular vectors show that the frequencies are evenly divided among terms as well as among documents.

Figure 3.3 shows what happens when we add another document about Belgium, with a slightly different frequency distribution of terms: the Belgian dimension becomes the most important (i.e. captures the most variation, 2.92), while the Dutch dimension remains the same (2.83). The third dimension (0.68) captures the remaining variation (the fact that the third document only talks about Brussels).

$$
A \begin{bmatrix}
 & B & NL & B \\
\textit{Groningen} & 0 & 2 & 0 \\
\textit{Leuven} & 2 & 0 & 0 \\
\textit{Amsterdam} & 0 & 2 & 0 \\
\textit{Brussel} & 2 & 0 & 1
\end{bmatrix} =
$$

$$
U \begin{bmatrix}
0.00 & -0.71 & \sout{0.00} \\
-0.66 & 0.00 & \sout{0.75} \\
0.00 & -0.71 & \sout{0.00} \\
-0.75 & 0.00 & \sout{-0.66}
\end{bmatrix}
\Sigma \begin{bmatrix}
2.92 & 0.00 & \sout{0.00} \\
0.00 & 2.83 & \sout{0.00} \\
\sout{0.00} & \sout{0.00} & \sout{0.68}
\end{bmatrix}
$$

$$
V^T \begin{bmatrix}
-0.97 & 0.00 & 0.26 \\
0.00 & -1.00 & 0.00 \\
\sout{-0.26} & \sout{0.00} & \sout{-0.97}
\end{bmatrix}
\cong \hat{A} \begin{bmatrix}
0.00 & 2.00 & 0.00 \\
1.87 & 0.00 & 0.50 \\
0.00 & 2.00 & 0.00 \\
2.12 & 0.00 & 0.56
\end{bmatrix}
$$

Figure 3.3: Truncated singular value decomposition

If we now truncate the SVD by keeping only the two most important dimensions, and then reconstruct our original matrix, we get matrix $\hat{\mathbf{A}}$, which is the best possible reconstruction from only two dimensions. Note that matrix $\hat{\mathbf{A}}$ resembles matrix $\mathbf{A}$, except for the numbers of the third document: instead of assigning all frequency mass to the term *Brussel*, the mass is almost evenly divided among the Belgian terms *Brussel* and *Leuven*. When keeping only two dimensions, the SVD 'guesses' the best possible distribution. This is an example of how the technique is used to obtain a more succinct model that is able to generalize among the data.

Below, we describe a more elaborate example, illustrating once again the generalization capacity of a singular value decomposition. Figure 3.4 represents another term-by-document matrix, containing Dutch nouns that are related to two distinct semantic topics. The nouns *tulp* 'tulip', *tuin* 'garden', and *park* 'park' are related to the topic of gardening. The nouns *ei* 'egg', *kaas* 'cheese', and *boter* 'butter' all relate

to the topic of food. The noun *bloem* is an ambiguous word in Dutch, meaning 'flower' (related to the gardening topic) as well as 'flour' (related to the food topic). Figure 3.4 represents the distribution of the seven nouns across five different documents. The complete SVD (matrices $\mathbf{U}$, Σ and $\mathbf{V}^T$) is given in figures 3.5 to 3.7.

$$\mathbf{A} = \begin{bmatrix} & d_1 & d_2 & d_3 & d_4 & d_5 \\ \text{tulp} & 1 & 0 & 1 & 0 & 0 \\ \text{tuin} & 1 & 1 & 0 & 0 & 0 \\ \text{park} & 0 & 1 & 0 & 0 & 0 \\ \text{ei} & 0 & 0 & 0 & 1 & 1 \\ \text{kaas} & 0 & 0 & 0 & 1 & 0 \\ \text{boter} & 0 & 0 & 0 & 1 & 1 \\ \text{bloem} & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 3.4: Term-by-document matrix $\mathbf{A}$

$$\mathbf{U} = \begin{bmatrix} & \text{dim1} & \text{dim2} & \text{dim3} & \text{dim4} & \text{dim5} \\ \text{tulp} & -0.21 & 0.52 & -0.48 & -0.58 & 0.35 \\ \text{tuin} & -0.22 & 0.60 & 0.46 & 0 & -0.40 \\ \text{park} & -0.05 & 0.22 & 0.65 & 0 & 0.55 \\ \text{ei} & -0.56 & -0.30 & 0.06 & 0 & 0.20 \\ \text{kaas} & -0.26 & -0.22 & 0.17 & -0.58 & -0.55 \\ \text{boter} & -0.56 & -0.30 & 0.06 & 0 & 0.20 \\ \text{bloem} & -0.47 & 0.30 & -0.31 & 0.58 & -0.20 \end{bmatrix}$$

Figure 3.5: The left-singular matrix $\mathbf{U}$

$$\Sigma = \begin{bmatrix} 2.30 & 0 & 0 & 0 & 0 \\ 0 & 1.93 & 0 & 0 & 0 \\ 0 & 0 & 1.30 & 0 & 0 \\ 0 & 0 & 0 & 1.00 & 0 \\ 0 & 0 & 0 & 0 & 0.52 \end{bmatrix}$$

Figure 3.6: The square diagonal matrix Σ

$$\mathbf{V}^T = \begin{bmatrix} & d_1 & d_2 & d_3 & d_4 & d_5 \\ \text{dim1} & -0.39 & -0.12 & -0.09 & -0.60 & -0.69 \\ \text{dim2} & 0.74 & 0.43 & 0.27 & -0.43 & -0.16 \\ \text{dim3} & -0.26 & 0.85 & -0.37 & 0.22 & -0.15 \\ \text{dim4} & 0 & 0 & -0.58 & -0.58 & 0.58 \\ \text{dim5} & -0.49 & 0.28 & 0.67 & -0.28 & 0.39 \end{bmatrix}$$

Figure 3.7: The right-singular matrix $\mathbf{V^T}$

We can now easily project the terms and documents of the original matrix into a space of reduced dimensionality; in the following example, we will retain two dimensions. Matrix **B** gives the terms after a reduction to two dimensions, scaled with the singular values. The matrix is obtained by multiplying a slice of matrix **U** ($\mathbf{U}_{7\times2}$) with a slice of matrix Σ ($\Sigma_{2\times2}$). Matrix **B** is given in figure 3.8. The term vectors – normalized to vector length – are represented graphically in figure 3.9.

$$\mathbf{B} = \begin{bmatrix} & \text{dim1} & \text{dim2} \\ \text{tulp} & -0.48 & 1.01 \\ \text{tuin} & -0.51 & 1.16 \\ \text{park} & -0.12 & 0.43 \\ \text{ei} & -1.28 & -0.58 \\ \text{kaas} & -0.60 & -0.43 \\ \text{boter} & -1.28 & -0.58 \\ \text{bloem} & -1.08 & 0.58 \end{bmatrix}$$

Figure 3.8: Matrix **B**, the multiplication of $\mathbf{U}_{7\times2}$ and $\Sigma_{2\times2}$

In figure 3.9, we can clearly distinguish the two different topics: the 'garden' topic (with *tulp*, *tuin* and *park*) in the second quadrant, and the 'food' topic (with *ei*, *kaas* and *boter*) in the third quadrant. Note that the terms *tulp* and *park* do not appear together in the same document in the original matrix; in the reduced two-dimensional SVD space, however, they are clearly closely related. This is again an example of the generalization capability of the SVD. Also note that the ambiguous word *bloem*, related to both the 'garden' topic and the 'food' topic, ends up in between them.

Similarly, we obtain matrix **C** – the projection of the documents into the two-dimensional reduced vector space – by multiplying $\Sigma_{2\times2}$ with $\mathbf{V}^T_{2\times5}$. Matrix **C** is given in figure 3.10. The document vectors – again normalized to vector length – are

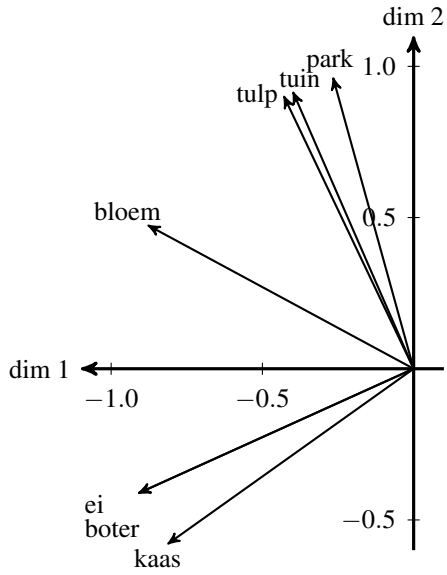

Figure 3.9: A graphical representation of the term vectors in the reduced dimensional space

represented graphically in figure 3.11.

$$\mathbf{C} = \begin{bmatrix} & d_1 & d_2 & d_3 & d_4 & d_5 \\ \text{dim1} & -0.89 & -0.27 & -0.21 & -1.37 & -1.58 \\ \text{dim2} & 1.42 & 0.82 & 0.52 & -0.82 & -0.30 \end{bmatrix}$$

Figure 3.10: Matrix $\mathbf{C}$, the multiplication of $\Sigma_{2\times 2}$ and $\mathbf{V}^T_{2\times 5}$

Again, we see the same topic division among the document vectors. Documents d_1, d_2 and d_3 are grouped together in the second quadrant, and documents d_4 and d_5 appear together in the third quadrant. Note again that documents d_2 and d_3 appear closely together, although they do not share any terms in the original term-document matrix.

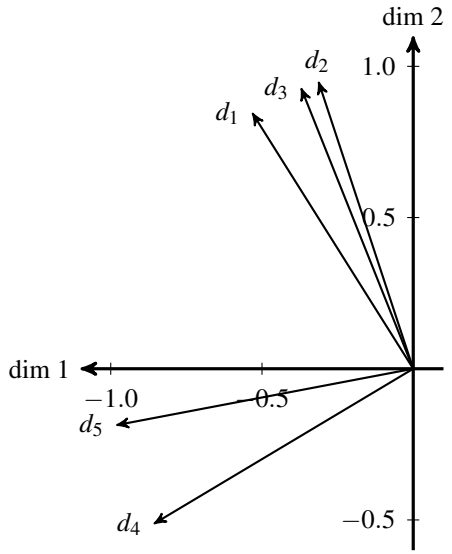

Figure 3.11: A graphical representation of the document vectors in the reduced dimensional space

3.2.4 Drawbacks

LSA suffers from a number of drawbacks, that have been regularly noted in the literature. (Manning and Schütze, 2000, p. 565)

The first major drawback is that a singular value decomposition assumes normally distributed data. A normal distribution is inappropriate for frequency count data, such as textual co-occurrence data. There are other distributions – such as a Poisson distribution – that are better suited for modeling count data. As a consequence of the normality assumption, the reconstruction $\mathbf{A}'$ of the original matrix $\mathbf{A}$ may contain negative numbers, which clearly is a bad approximation for frequency counts.

A second drawback – related to the first one – is the presence of negative values in the derived dimensions themselves. The derived dimensions are said to represent actual 'latent semantic' dimensions. A particular term or document can have a positive or negative value on those dimensions. It is not clear what negative values on a semantic scale should designate. A particular term or document either is related (positive value) or is not related (zero value) to a particular topic; it seems counterintuitive to say that a particular word is negatively related to a particular topic. This intuition is confirmed by

experiments. In the following section, we will present an algorithm that only allows non-negative data in its dimensionality reduction. By enforcing this constraint, the algorithm is able to find much more distinct and clear-cut semantic dimensions.

3.3 Non-negative matrix factorization

3.3.1 Introduction

In this section, we describe a dimensionality reduction technique called non-negative matrix factorization (NMF) that does not suffer from the drawbacks of LSA and its underlying singular value decomposition. Non-negative matrix factorization is a dimensionality reduction technique that has become popular in fields such as image recognition, speech recognition and machine learning. Its key idea is to impose a non-negativity constraint on the factorization. This constraint brings about a parts-based representation, because only additive and no subtractive combinations are allowed. In many cases, this constraint proves beneficial for the inductive capabilities of the dimensionality reduction: the algorithm is able to extract more clear and distinct characteristics from the data.

The difference between the parts-based induction of NMF and the holistic induction of non-constrained methods such as PCA (and the related singular value decomposition) can be illustrated with an example from facial image recognition (Lee and Seung, 1999). A famous method in facial image recognition uses so-called 'eigenfaces' (Turk and Pentland, 1991). These are a small number of prototypical faces represented by the eigenvectors that are found by applying PCA to a database of facial images. Eigenfaces may contain positive as well as negative values. A key characteristic is that they are 'holistic': an eigenface contains all kinds of facial traits, and thus represents a prototypical face. By taking a linear combination of various 'eigenfaces', a particular instance of a face may be reconstructed.

The representation that is found by NMF looks quite different: instead of finding holistic, prototypical faces, the algorithm induces particular facial traits (different kinds of eyes, noses, mouths, ...). By enforcing a non-negative constraint, the algorithm is able to build up a parts-based representation of facial images. A particular instance of a face may then be reconstructed by taking a linear combination of the different parts. The very same characteristic will also prove to be beneficial for building up semantic representations from text.

3.3.2 Theory

Non-negative matrix factorization (NMF) (Lee and Seung, 2000) is the name for a group of algorithms in which a matrix **V** is factorized into two other matrices, **W** and **H**.

$$\mathbf{V}_{n\times m} \approx \mathbf{W}_{n\times r}\mathbf{H}_{r\times m} \tag{3.3}$$

Figure 3.12 gives a graphical representation of non-negative matrix factorization.

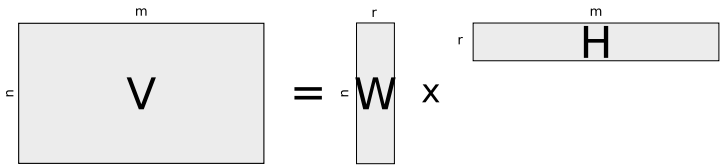

Figure 3.12: A graphical representation of non-negative matrix factorization

Typically r is much smaller than n, m so that both instances and features are expressed in terms of a few components. As mentioned above, non-negative matrix factorization enforces the constraint that all three matrices must be non-negative, so all elements must be greater than or equal to zero.

There are two objective functions that may be used in order to quantify the quality of the approximation of the original matrix. One objective function minimizes the sum of squares (equation 3.4).

$$\min \ \| \mathbf{V} - \mathbf{WH} \|_F = \ \min \sum_i \sum_j (\mathbf{V}_{ij} - (\mathbf{WH})_{ij})^2 \tag{3.4}$$

The other one minimizes the Kullback-Leibler divergence (equation 3.5).

$$\min \ D_{\mathrm{KL}}(\mathbf{V} \parallel \mathbf{WH}) = \min \sum_i \sum_j \left(\mathbf{V}_{ij} \log \frac{\mathbf{V}_{ij}}{(\mathbf{WH})_{ij}} - \mathbf{V}_{ij} + (\mathbf{WH})_{ij} \right) \tag{3.5}$$

Practically, the factorization can be efficiently carried out through the iterative application of multiplicative update rules. The set of update rules that minimize the Euclidean distance are given in 3.6 and 3.7.

$$\mathbf{H}_{a\mu} \leftarrow \mathbf{H}_{a\mu} \frac{(\mathbf{W}^T\mathbf{V})_{a\mu}}{(\mathbf{W}^T\mathbf{WH})_{a\mu}} \tag{3.6}$$

$$\mathbf{W}_{ia} \leftarrow \mathbf{W}_{ia} \frac{(\mathbf{V}\mathbf{H}^T)_{ia}}{(\mathbf{W}\mathbf{H}\mathbf{H}^T)_{ia}} \tag{3.7}$$

The set of update rules that minimize the Kullback-Leibler divergence are given in 3.8 and 3.9.

$$\mathbf{H}_{a\mu} \leftarrow \mathbf{H}_{a\mu} \frac{\sum_i \mathbf{W}_{ia} \frac{\mathbf{V}_{i\mu}}{(\mathbf{W}\mathbf{H})_{i\mu}}}{\sum_k \mathbf{W}_{ka}} \tag{3.8}$$

$$\mathbf{W}_{ia} \leftarrow \mathbf{W}_{ia} \frac{\sum_\mu \mathbf{H}_{a\mu} \frac{\mathbf{V}_{i\mu}}{(\mathbf{W}\mathbf{H})_{i\mu}}}{\sum_v \mathbf{H}_{av}} \tag{3.9}$$

Matrices $\mathbf{W}$ and $\mathbf{H}$ are randomly initialized, and the update rules are iteratively applied – alternating between them. In each iteration, the matrices $\mathbf{W}$ and $\mathbf{H}$ are suitably normalized, so that the rows of the matrices sum to 1. The algorithm stops after a fixed number of iterations, or according to some stopping criterion (the change of the objective function drops below a certain threshold). The update rules are guaranteed to converge to a local optimum. In practice, it is usually sufficient to run the NMF algorithm repeatedly in order to find the global optimum.

3.3.3 Example

In the following example, we take matrix $\mathbf{V}$ in figure 3.13 (reproduced from matrix $\mathbf{A}$ used in the SVD example on page 38), and factorize it to two dimensions using non-negative matrix factorization. As objective function, we take the Kullback-Leibler divergence (which implies the use of the update rules in 3.8 and 3.9). The globally optimal matrices $\mathbf{W}$ and $\mathbf{H}$ are represented in figures 3.14 and 3.15.[2]

Matrices $\mathbf{W}$ and $\mathbf{H}$ can be interpreted as conditional probabilities. Matrix $\mathbf{W}$ represents the probability of a word given a particular topical, 'semantic' dimension.

Matrix $\mathbf{H}$ gives the probability of a dimension given a document (in the example, each document contains one particular topic).

By multiplying matrices $\mathbf{W}$ and $\mathbf{H}$, we get matrix $\mathbf{V}'$, containing the probabilities of a word given a document.

Note that the values of (tulp, d_2), (tuin, d_3), and (park, $d_{1,3}$) – that have zeros in the original co-occurrence matrix – have received appropriate probability values in the reconstructed matrix $\mathbf{V}'$. The factorization model has made correct inferences about

[2]Note once again that the update rules are guaranteed to converge to a local optimum; it might take a number of tries to find the globally optimal solution.

$$\mathbf{V} = \begin{bmatrix} & d_1 & d_2 & d_3 & d_4 & d_5 \\ \text{tulp} & 1 & 0 & 1 & 0 & 0 \\ \text{tuin} & 1 & 1 & 0 & 0 & 0 \\ \text{park} & 0 & 1 & 0 & 0 & 0 \\ \text{ei} & 0 & 0 & 0 & 1 & 1 \\ \text{kaas} & 0 & 0 & 0 & 1 & 0 \\ \text{boter} & 0 & 0 & 0 & 1 & 1 \\ \text{bloem} & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 3.13: Term-by-document matrix $\mathbf{V}$

$$\mathbf{W} = \begin{bmatrix} & \text{dim1} & \text{dim2} \\ \text{tulp} & 0 & \frac{1}{3} \\ \text{tuin} & 0 & \frac{1}{3} \\ \text{park} & 0 & \frac{1}{6} \\ \text{ei} & \frac{1}{3} & 0 \\ \text{kaas} & \frac{1}{6} & 0 \\ \text{boter} & \frac{1}{3} & 0 \\ \text{bloem} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}$$

Figure 3.14: Matrix $\mathbf{W}$, containing the original nouns and a reduced number of dimensions

$$\mathbf{H} = \begin{bmatrix} & d_1 & d_2 & d_3 & d_4 & d_5 \\ \text{dim1} & 0 & 0 & 0 & 1 & 1 \\ \text{dim2} & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Figure 3.15: Matrix $\mathbf{H}$, containing the original documents and a reduced number of dimensions

words related to the gardening topic appearing in sentences related to the gardening topic. Likewise, (kaas, d_5) has received an appropriate probability value. The ambiguous word *bloem* – related to the two topics – has received appropriate probability values across the whole document range.

$$\mathbf{V}' = \begin{bmatrix} & d_1 & d_2 & d_3 & d_4 & d_5 \\ \text{tulp} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \text{tuin} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \text{park} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 & 0 \\ \text{ei} & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} \\ \text{kaas} & 0 & 0 & 0 & \frac{1}{6} & \frac{1}{6} \\ \text{boter} & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} \\ \text{bloem} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}$$

Figure 3.16: The reconstructed matrix $\mathbf{V}'$, the multiplication of $\mathbf{W}$ and $\mathbf{H}$

Chapter 4

Three-way Methods

4.1 Introduction

In the former chapters, language data has been treated as two-way co-occurrences; we have shown that the semantic similarity of words can be investigated by looking at the contextual features with which those words appear. The two-way co-occurrence data has been represented in the form of a *matrix*. It is the form that is best suited for representing two-way co-occurrence frequencies. The matrix representation allows for the application of algebraic and statistical methods, which in turn allows for the induction of semantic generalizations.

Reducing language to two-way co-occurrence frequencies is of course a vast oversimplification. Language is a complex system of interrelated words, driven by grammar rules and subcategorization frames. By keeping track of multi-way co-occurrences, we can attempt to do some more justice to this complexity (although the framework represented here remains a major simplification). Compare the following sentences:

(1) De voetbalclub$_{su}$ speelt een goede wedstrijd$_{obj}$.
The soccer team plays a good game
The soccer team is playing a good game.

(2) De acteur$_{su}$ speelt Hamlet$_{obj}$.
The actor plays Hamlet
The actor is playing Hamlet.

⟨speel	su	voetbalclub⟩
⟨speel	obj	wedstrijd ⟩
⟨speel	su	acteur ⟩
⟨speel	obj	Hamlet ⟩

Table 4.1: Extracted dependency relations

The dependency relations (subject-verb and verb-object) for examples (1) and (2) are given in table 4.1.

Say we now want to investigate the semantics of Dutch verbs (like *spelen* 'to play'). In our standard two-way framework, we would then take the verbs to be one mode, and combine the syntactic dependencies that the verbs appear with (subjects and direct objects in this case) together in the other mode, yielding a matrix like the one in figure 4.1.

$$\mathbf{V} = \begin{bmatrix} & voetbalclub_{su} & acteur_{su} & wedstrijd_{obj} & Hamlet_{obj} & \dots \\ \text{speel} & 1 & 1 & 1 & 1 & \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

Figure 4.1: Two-way co-occurrence matrix

This is a genuine, appropriate way of investigating a verb's semantics, but we do lose some more complex and interesting relations between a verb and its objects. By capturing a verb's co-occurrences in a matrix form, we are able to investigate its co-occurrence with particular subjects and direct objects separately, but we are not able to investigate a verb's co-occurrence with subjects and direct objects *at the same time*: we lose the three-way relationship that exists between verb, subject and direct object.

It is possible, however, to capture in a matrix the relationship between subjects and direct objects for one particular verb. For the examples with *spelen* above, this yields the matrix given in figure 4.2. The rows of the matrix represent the subjects, the columns represent the direct objects. A graphical representation of this matrix is given in figure 4.3.

We can now construct similar matrices for each verb we want to investigate. Say we want two include two more verbs, *winnen* 'to win', and *bewerken* 'to adapt'. The matrices for these verbs are given in figures 4.4 and 4.5. Note that the two matrices

$$\mathbf{V}_{speel} = \begin{bmatrix} & \text{wedstrijd} & \text{Hamlet} & \dots \\ \text{voetbalclub} & 1 & 0 & \dots \\ \text{acteur} & 0 & 1 & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix}$$

Figure 4.2: Two-way co-occurrence matrix of subjects and direct objects for the verb *spelen* 'to play'

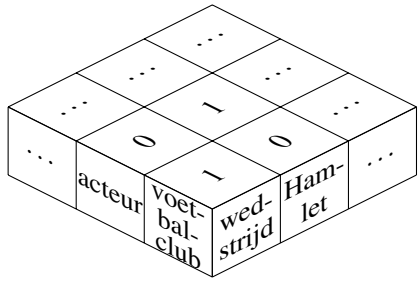

Figure 4.3: Graphical representation of the co-occurrence matrix for the verb *spelen* 'to play'

$\mathbf{V}_{win}$ and $\mathbf{V}_{bewerk}$ contain the same instances and features (subjects and direct objects) in their rows and columns as the first matrix $\mathbf{V}_{speel}$.

$$\mathbf{V}_{win} = \begin{bmatrix} & \text{wedstrijd} & \text{Hamlet} & \dots \\ \text{voetbalclub} & 1 & 0 & \dots \\ \text{acteur} & 1 & 0 & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix}$$

Figure 4.4: Two-way co-occurrence matrix of subjects and direct objects for the verb *winnen* 'to win'

As a last step, we can now stack the three matrices together, which yields a three-dimensional cube like the one in figure 4.6.

It will have become clear by now that we are leaving the familiar domain of two-dimensional matrices. As we have noted before, a matrix is not a suitable form for the representation of multi-way data. For co-occurrence data beyond two modes, we

$$\mathbf{V}_{bewerk} = \begin{bmatrix} & \text{wedstrijd} & \text{Hamlet} & \dots \\ \text{voetbalclub} & 0 & 0 & \dots \\ \text{acteur} & 0 & 1 & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix}$$

Figure 4.5: Two-way co-occurrence matrix of subjects and direct objects for the verb *bewerken* 'to adapt'

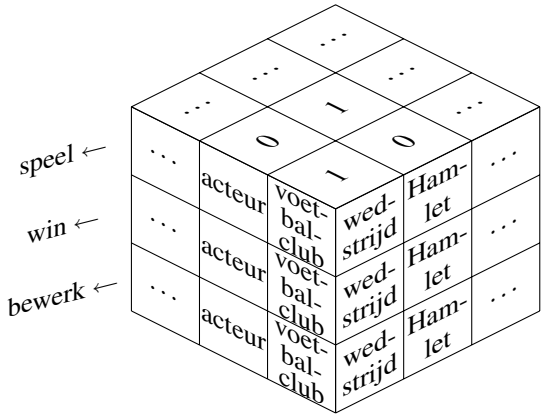

Figure 4.6: Graphical representation tensor

need a more general representation. The generalization of a matrix is called a *tensor*. A tensor is able to encode co-occurrence data of any n modes. Figure 4.7 shows a graphical comparison of a matrix and a tensor with three modes – although a tensor can easily be generalized to more than three modes.[1]

We are now leaving the familiar domain of two-dimensional matrix algebra, and enter the realm of higher-dimensional tensor algebra. Tensor algebra involves some novel mathematical machinery. The goal of the next section is to provide a succinct introduction.

[1] We will stick to examples that use no more than three modes; it is easy to construct higher order tensors mathematically, but it is impossible – or at least very difficult – to represent them visually.

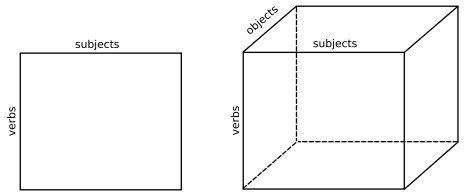

Figure 4.7: Matrix representation vs. tensor representation

4.2 Tensor algebra

In this overview of tensor algebra, we will review some conceptual and notational preliminaries (based on Kiers (2000) and Kolda and Bader (2009)), and focus on some vital tensor operations required for the tensor-based dimensionality reductions that are explained in the subsequent sections.

The *order* of a tensor is the number of 'dimensions'.[2] Vectors (tensors of order one) are denoted by boldface lowercase letters ($\mathbf{x}$). Matrices (tensors of order two) are denoted by boldface capital letters ($\mathbf{X}$). Higher-order tensors (order three or higher) are denoted by boldface calligraphic letters ($\mathcal{X}$). Scalars are denoted by lowercase letters (x).

The ith entry of a vector $\mathbf{x}$ is written as x_i, element (i, j) of a matrix $\mathbf{X}$ is written as x_{ij}, and element (i, j, k) of a third-order tensor $\mathcal{X}$ is written as x_{ijk}. Indices range from 1 to D, so $i = 1, \ldots, D$. The nth element in a sequence is written as a superscript in parentheses, so $\mathbf{A}^{(n)}$ denotes the nth matrix in a sequence.

The *norm* of a tensor $\mathcal{X} \in \mathbb{R}^{D_1 \times D_2 \times D_3}$ – analogous to the Frobenius norm for matrices – is the square root of the sum of squares of all its elements (equation 4.1).

$$\| \mathcal{X} \| = \sqrt{\sum_{i_1=1}^{D_1} \sum_{i_2=1}^{D_2} \cdots \sum_{i_N=1}^{D_N} x_{i_1 i_2 \ldots i_N}^2} \tag{4.1}$$

An N-order tensor $\mathcal{X} \in \mathbb{R}^{D_1 \times D_2 \times \ldots \times D_N}$ is of *rank one* if it can be written as the outer product of N vectors.

$$\mathcal{X} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \ldots \circ \mathbf{a}^{(N)} \tag{4.2}$$

[2] The term *dimension* is ambiguous, because it is also used to denote the cardinality of vectors and vector spaces. Therefore, the dimensions of a tensor are usually referred to as *modes* or *ways*.

The small circle ($\circ$) denotes the outer product, and it is calculated by multiplying for each element of the tensor the corresponding vector elements.

$$x_{i_1 i_2 \ldots i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} \ldots a^{(N)}_{i_N} \quad \text{for all } 1 \leq i_n \leq I_n \tag{4.3}$$

A third order rank one tensor is given in figure 4.8.

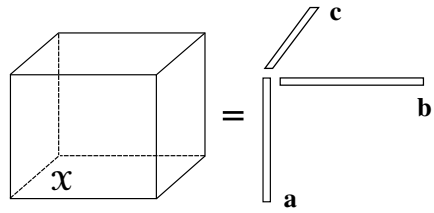

Figure 4.8: A third order rank one tensor, which can be written as the outer product of three vectors

Finally, the *rank* of a tensor $\mathcal{X}$ is defined as the smallest number of rank one tensors whose sum is equal to $\mathcal{X}$. Figure 4.9 shows a tensor that can be generated with the sum of three rank one tensors (the sum of three outer products). Therefore, rank$(\mathcal{X}) = 3$.

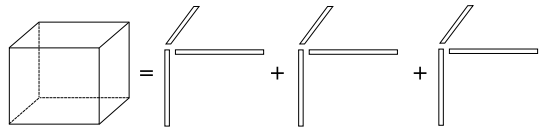

Figure 4.9: A third order tensor of rank three, which can be written as the sum of three outer products

4.3 Three-way factorization algorithms

To be able to cope with multi-way data, several algorithms have been developed as multilinear generalizations of the SVD. In statistics, three-way component analysis has been extensively investigated (for an overview, see Kiers and van Mechelen (2001)). The two most popular methods are parallel factor analysis (PARAFAC, Harshman (1970), Carroll and Chang (1970)) and three-mode principal component analysis (3MPCA, Tucker (1966)), also called higher order singular value decomposition (HOSVD, De Lathauwer

et al. (2000)). Three-way factorizations have been applied in various domains, such as psychometry and image recognition (Vasilescu and Terzopoulos, 2002). In information retrieval, three-way factorizations have been applied to the problem of link analysis (Kolda and Bader, 2006).

One last important method dealing with multi-way data is non-negative tensor factorization (NTF, Shashua and Hazan (2005)). NTF is a generalization of non-negative matrix factorization, and can be considered an extension of the PARAFAC model with the constraint of non-negativity.

In the next sections, we will look in some more detail at two different factorization algorithms: parallel factor analysis and non-negative tensor factorization.

4.3.1 Parallel factor analysis

Parallel factor analysis (PARAFAC) is a multilinear analogue of the singular value decomposition (SVD) used in latent semantic analysis. The key idea is to minimize the sum of squares between the original tensor and the factorized model of the tensor. For the three mode case of a tensor $\mathcal{T} \in \mathbb{R}^{D_1 \times D_2 \times D_3}$ this gives equation 4.4, where k is the number of dimensions in the factorized model (recall that $\circ$ denotes the outer product).

$$\min_{x_i \in \mathbb{R}^{D1}, y_i \in \mathbb{R}^{D2}, z_i \in \mathbb{R}^{D3}} \| \mathcal{T} - \sum_{i=1}^{k} x_i \circ y_i \circ z_i \| \tag{4.4}$$

The factorization algorithm finds a tensor (as a sum of rank one tensors) that is as similar as possible to the original tensor in the least squares sense, but has a fixed lower rank k: it is the best possible low rank approximation of the original tensor for rank k.

The algorithm results in three matrices, indicating the loadings of each mode on the factorized dimensions. The model is represented graphically in figure 4.10, visualizing the fact that the PARAFAC decomposition consists of the summation over the outer products of n (in this case three) vectors.

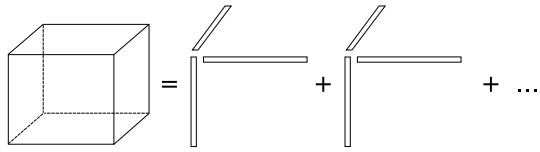

Figure 4.10: Graphical representation of PARAFAC as the sum of outer products

Figure 4.11 represents the factorization as three loadings matrices, containing the loadings on each factor for the three different modes. The representation is equivalent to the representation with the sum of outer products – it differs only conceptually.

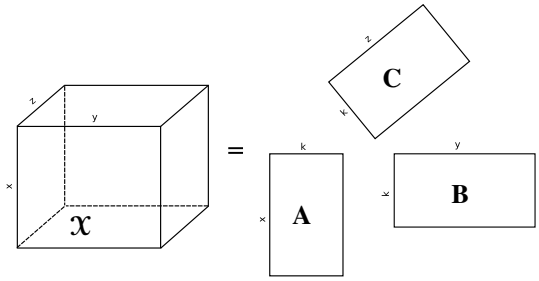

Figure 4.11: Graphical representation of PARAFAC as three loadings matrices

There are a number of algorithms available to calculate the PARAFAC decomposition. The most popular one is the alternating least squares method (ALS), that was proposed in the original papers by Harshman (1970), and Carroll and Chang (1970). In each iteration, two of the modes are fixed and the third one is fitted in a least squares sense. This calculation is done for each mode in turn, and the process is repeated until convergence.

4.3.2 Non-negative Tensor Factorization

Our second multi-way factorization is called non-negative tensor factorization (NTF; it is the generalization of non-negative matrix factorization for multi-way data. The NTF model is similar to the PARAFAC analysis, with the constraint that all data needs to be non-negative (i.e. ≥ 0).

With non-negative tensor factorization, the non-negativity constraint is enforced, yielding a model like the one in equation 4.5:

$$\min_{x_i \in \mathbb{R}^{D1}_{\geq 0}, y_i \in \mathbb{R}^{D2}_{\geq 0}, z_i \in \mathbb{R}^{D3}_{\geq 0}} \| \mathcal{T} - \sum_{i=1}^{k} x_i \circ y_i \circ z_i \|_F^2 \tag{4.5}$$

As with the PARAFAC model, the algorithm results in three matrices, indicating the loadings of each mode on the factorized dimensions.

There are again a number of ways to compute the factorization. Bro and De Jong (1997) use again an alternating least squares algorithm (similar to the ALS

algorithm for PARAFAC, but with some adaptations to enforce non-negativity; the non-negativity can be enforced by using a non-negative least squares computation, based on Lawson and Hanson (1974). Another possibility is to use multiplicative update rules – similar to the update rules for non-negative matrix factorization, explained in section 3.3.2 (Welling and Weber, 2001).

Part II

Evaluation

Chapter 5

Evaluation of Wordnet-based Similarity

5.1 Introduction

In this chapter – and the next two – we provide a quantitative evaluation of different word space models and their parameters. More specifically, we investigate three different groups of models that have been built up according to the three hypotheses formulated in chapter 1 (viz. document-based, window-based and syntax-based models), using the formal framework presented in chapter 2. With this quantitative evaluation, we want to determine which models are best suited for the extraction of lexico-semantic information. Also, we want to find out whether there are differences in the kind of lexico-semantic information that is captured by the different models. Special attention is devoted to models applying some form of dimensionality reduction; we want to investigate whether the use of dimensionality reduction algorithms proves beneficial for particular models.

In this chapter, the models are evaluated by comparing them to CORNETTO, a handcrafted lexico-semantic hierarchy for Dutch, similar to WordNet. If two words are close to each other in the hierarchy, they are semantically similar. By comparing a model's output to the CORNETTO database, we can determine how good a particular model is at extracting semantic similarity. More specifically, we are able to determine the different models' ability to induce tight, synonym-like similarity (cfr. section 1.3). By nature, words that are close to each other in an IS-A hierarchy tend to be tightly related. The results presented in this chapter thus evaluate the models' ability to extract tight, synonym-like similarity. In chapter 6, we will evaluate the same synonym-

like similarity extraction by comparing the output of a clustering algorithm to a gold standard classification. In chapter 7, we will then evaluate the models' ability to induce semantically related, i.e. topically similar words.

We start out with a description of the different resources and tools that have been used for the construction of the models and their evaluation. Next, we describe the evaluation framework employed for the calculation of wordnet-based similarity. The next part – the main section of this chapter – presents the evaluation results for the three different models (document-based, window-based and syntax-based) and their various parameters. In the last section, we compare the different models, and draw some conclusions about their ability to extract semantic similarity – and their aptness for particular applications.

5.2 Methodological remarks

5.2.1 Corpus

All contextual co-occurrence information that has been used for building the models has been extracted from the Twente Nieuws Corpus (TWNC), a 500M word corpus of Dutch newspaper texts from the period 1999-2005.[1] The corpus has been consistently divided into articles and paragraphs, which can be used as document sizes in the document-based models. Additionally, the corpus has been parsed with ALPINO (van Noord, 2006) – a dependency-parser for Dutch – so that dependency-relations are available for use in the syntax-based models.

5.2.2 Lexico-semantic database

The models are evaluated by comparing the results to the Dutch lexico-semantic database CORNETTO (Horak, Vossen, and Rambousek, 2008). CORNETTO is a recently developed database that combines information from the Dutch part of EuroWordNet (Vossen, 1998) and Referentiebestand Nederlands (Maks, Martin, and de Meerseman, 1999). The database contains hierarchically structured IS-A relations, similar to the original WordNet (Fellbaum, 1998). CORNETTO contains about 70K synsets,[2] which altogether contain about 100K words. About 50K of all synsets are noun synsets, accounting for about 75K nouns in total. A sample extract of CORNETTO's noun hierarchy is given in figure 5.1 on page 63.

[1] The corpus contains articles from four different Dutch newspapers, viz. *Algemeen Dagblad*, *NRC Handelsblad*, *Parool*, *Trouw* and *Volkskrant*.

[2] A synset is a set of synonymous words within the semantic IS-A hierarchy.

5.2.3 Tools & implementation

All models and corresponding similarity calculations have been implemented in PYTHON, partially using the NUMPY module for scientific computations (which is useful for a number of linear algebraic operations). SVD calculations have been carried out using the SVDPACKC library (Berry, 1992), which allows fast sparse truncated singular value decompositions, depending on the number of singular values one wants to retain. NMF calculations have been carried out in MATLAB, using a sparse implementation of the NMF algorithm. The evaluation framework itself has been implemented in PYTHON as well, making extensive use of the PYCORNETTO interface developed by Erwin Marsi.

5.3 Evaluation framework

5.3.1 Introduction

The models are automatically evaluated by comparing them to the lexico-semantic database CORNETTO. By looking up the distance between two words in the database, we can determine how similar they are. A number of similarity measures have been developed in order to formalize this distance measurement in WordNet-like hierarchies. First, we provide a description of the two wordnet similarity measures used in the subsequent evaluations. Next, we describe the actual evaluation framework and the design of the test set.

5.3.2 Similarity measures

In this section, we describe two wordnet similarity measures that will be used for measuring the similarity between words in a wordnet taxonomy. We will use the following definitions and notation, largely following Budanitsky and Hirst (2006):[3]

- The **length** of the shortest path from synset c_i to synset c_j (measured by counting the edges between nodes) is denoted by $\text{len}(c_i, c_j)$. A top *root* node ensures the existence of a path between any two nodes.
- The **depth** of a node is the length of the path from the global root to the node, i.e., $\text{depth}(c_i) = \text{len}(root, c_i)$.
- the **least common subsumer** – $\text{lcs}(c_1, c_2)$ – is the common subsumer for which the path length $\text{len}(c_1, c_2)$ is minimal, i.e. the most specific common superclass.

[3] An extensive overview of different similarity measures is provided in this paper as well.

- Given any formula $sim_x(c_1, c_2)$ for semantic similarity between two concepts c_1 and c_2, the similarity $sim_x(w_1, w_2)$ between two words w_1 and w_2 can be calculated as

$$sim_x(w_1, w_2) = \max_{c_1 \in s(w_1), c_2 \in s(w_2)} sim_x(c_1, c_2) \tag{5.1}$$

  where $s(w_i)$ is 'the set of concepts in the taxonomy that are senses of word w_i' (Resnik, 1995), and $sim_x(c_1, c_2)$ is either of the two similarity measures described below. So we make the assumption that the similarity of two words is equal to the similarity of the words' most related senses.

Similarity by path length

Wu and Palmer (1994) propose a measure that computes the similarity between two concepts by looking at the path length between them in the wordnet taxonomy. Intuitively, it seems reasonable to take the path length into account when determining semantic similarity: the closer two nodes are together in a taxonomy, the more semantically similar they are. Some precaution needs to be taken, however. If we only consider path length, then all links need to represent a uniform 'semantic distance'. Usually, this is not the case. Particular subtaxonomies might be much denser than others,[4] and the distance that one particular link represents may therefore differ a lot. Wu and Palmer propose the following measure to measure the path length – while at the same time scaling the outcome to the relative position of the words in the taxonomy:

$$sim_{wp}(c_1, c_2) = \frac{2C}{A + B + 2C} \tag{5.2}$$

where $A = \text{len}(c_1, \text{lcs}(c_1, c_2))$, $B = \text{len}(c_2, \text{lcs}(c_1, c_2))$, and $C = \text{depth}(\text{lcs}(c_1, c_2))$. Note that $\text{len}(c_1, \text{lcs}(c_1, c_2)) + \text{len}(c_2, \text{lcs}(c_1, c_2))$ denotes the path length from c_1 to c_2, and $\text{depth}(\text{lcs}(c_1, c_2))$ denotes the global depth of the path in the taxonomy. If a particular path has a large depth, it will be given more weight compared to a path that is high up in the taxonomy.

Let us illustrate the Wu and Palmer similarity measure with an example. In figure 5.1, an extract of the CORNETTO database is given.

[4] Some taxonomies might concentrate on technical topics, or have a detailed subtaxonomy for particular scientific fields such as biology.

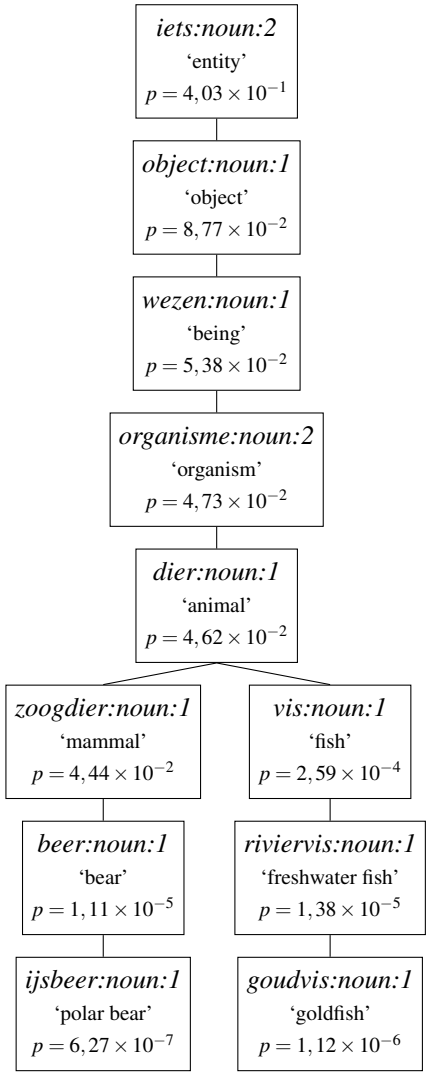

Figure 5.1: CORNETTO extract

Say we want to calculate the Wu and Palmer similarity between *ijsbeer* 'polar bear' and *haai* 'shark', i.e. sim_{wp}(*ijsbeer*,*haai*).[5] To do so, we first determine the least common subsumer lcs(*ijsbeer:noun:1*, *goudvis:noun:1*) = *dier:noun:1*. Next, we determine that len(*ijsbeer:noun:1*, *dier:noun:1*) = 3, that len(*goudvis:noun:1*, *dier:noun:1*) = 3, and that depth(*dier:noun:1*) = 4.[6] It is now straightforward to determine that $sim_{wp} = \frac{2\times4}{3+3+(2\times4)} = 0.571$.

Wu and Palmer's similarity measure is not the only similarity measure based on path length in a taxonomy. A similar measure has been proposed by Leacock and Chodorow (1998).

The Wu and Palmer similarity measure is able to mend the problem of varying link distances, and in practice proves to correlate well with human judgments of semantic similarity. However, the measure still depends on the actual taxonomic structure. In the next section, an alternative method for measuring semantic similarity in a taxonomy is presented, based on information theory. It takes into account the taxonomic structure, but does not depend directly on the link distances.

Information-theoretic similarity

Intuitively, the more information two particular concepts share, the more similar they are. This information is captured indirectly by edge-counting methods. If two concepts are subsumed by a common subsumer low in the taxonomy, they are likely to share many characteristics; if two concepts only have a subsumer higher up in the taxonomy, they probably do not share many characteristics.

If we associate probabilities with concepts in the taxonomy, we are able to capture the same information without being dependent on the (varying and unreliable) link distances. For every concept c in the taxonomy, we calculate the probability $p(c)$, the probability of encountering an instance of concept c.[7] We can then straightforwardly calculate the information content of a concept c as the *self-information* of $p(c)$:

$$I(p(c)) = -\log p(c) \tag{5.3}$$

[5]Note that this means looking up the different concepts that are designated by *ijsbeer* and *goudvis*. In this case, there is only a single concept for each word, *ijsbeer:noun:1* and *goudvis:noun:1*. If there were more concepts for the words we want to look up, we would calculate the similarity between each pair of concepts, and take the maximum of these similarity scores.

[6]Note that we determine the path length by counting the edges between nodes. Another option is to count the number of nodes themselves, which is the option that Wu and Palmer take in their original implementation. Other researchers (Resnik, 1999) have opted for the edge-counting method in order to determine the path length, which is the approach we adopt here.

[7]Note that this means that the probability function p is monotonically nondecreasing when moving up in the taxonomy: if c_1 subsumes c_2, then $p(c_1) \geq p(c_2)$.

Note that, as the probability increases, informativeness decreases. So the more abstract a concept, the lower its information content. This quantitative characterization of the information content of a concept provides an alternative way to compute the semantic similarity of concepts. "The more information two different concepts share, the more similar they are, and the information shared by two concepts is indicated by the information content of the concepts that subsume them in the taxonomy." (Resnik, 1999)

Formally, this can be defined as:

$$sim_{IC} = \max_{c \in S(c_1,c_2)} -logp(c) \tag{5.4}$$

where $S(c_1,c_2)$ is the set of concepts which subsume both c_1 and c_2. The synset that achieves the maximum value in equation 5.4. In practice, this will always be the least common subsumer of c_1 and c_2, because no subordinate synset is less informative than its superordinates:[8]

$$sim_{IC} = -logp(lcs(c_1,c_2)) \tag{5.5}$$

Information content in itself is a suitable measure of semantic similarity (see Resnik (1999)), but in our evaluation framework we will make use of Lin's (1998b) measure as an information-theoretic measure of semantic similarity. Lin's measure for semantic similarity can be seen as a normalized version of information content. It is defined as:

$$sim_{lin}(c_1,c_2) = \frac{2 \times \log p(lcs(c_1,c_2))}{\log p(c_1) + \log p(c_2)} \tag{5.6}$$

Note that – except for the multiplicative constant 2 – the measure is the same as the information content similarity measure in equation 5.5, but normalized by the combined information content of the two concepts assuming they are independent. Lin's measure thus not only takes the commonality of the two concepts into account, but also the differences (both expressed in terms of their information content).

To be able to implement the measure, we need to estimate the probabilities of all concepts in the taxonomy. To do so, we look at the frequencies of the words in a large corpus. Each word in the corpus is counted as an occurrence of the concept that subsumes it. In the example in figure 5.1, an occurrence of the word *ijsbeer* counts as an occurrence of all the superordinate classes (*beer*, *zoogdier*, *dier*, *organisme*, and so on). Formally,

[8] The formula was designed this way to be able to cope with multiple inheritance. In this case, two nodes might both be least common subsumers, but they will have different information content values.

$$freq(c) = \sum_{n \in words(c)} freq(n) \tag{5.7}$$

where $words(c)$ is the set of words subsumed by concept c. The probabilities then just amount to the relative frequencies:

$$p(c) = \frac{freq(c)}{N} \tag{5.8}$$

where N is the total number of words in the corpus (that are present in the taxonomy).

To be able to precisely estimate the probabilities of all concepts, we would need a sense-tagged corpus. But large sense-tagged corpora are not readily available, especially for Dutch. Therefore, upon encountering a particular word in the corpus, the frequency of all senses of the word will be increased. This method makes the frequency estimation weaker but more generally applicable. In practice, it is the only method that allows the use of a large corpus.

Let us illustrate the Lin similarity measure again with the example in figure 5.1. To calculate the Lin's similarity between *ijsbeer* 'polar bear' and *haai* 'shark', we first extract the probabilities of the two words and their least common subsumer:

$$p(\textit{ijsbeer:noun:1}) = 6,27 \times 10^{-7} \quad p(\textit{goudvis:noun:1}) = 1,12 \times 10^{-6}$$
$$p(\textit{dier:noun:1})^9 = 4,62 \times 10^{-2}$$

It is now straightforward to calculate Lin's similarity according to equation 5.6:

$$sim_{lin}(\textit{ijsbeer,haai}) = \frac{2 \times \log(4,62 \times 10^{-2})}{\log(6,27 \times 10^{-7}) + \log(1,12 \times 10^{-6})} = 0.220$$

5.3.3 Evaluation of pairwise similarity

In order to evaluate a model's performance for a particular word, we calculate the pairwise similarity between that word and its top k similar words. For each target word, the top k similar words are extracted. We then calculate the similarity between the target word and each of its top k nouns (using one of the similarity measures described above). The pairwise similarity value for a particular word at k is then the average of the pairwise similarity values. Formally,

$$sim_{x,k}(w_0) = \frac{\sum_{i=1}^{k} sim_x(w_0, w_i)}{k} \tag{5.9}$$

where $w_0, \ldots, w_k$ is a list of top similar words (the first word w_0 being the target word for which the similar words are calculated), sim_x is either sim_{wp} or sim_{lin}, and k is the number of similar words over which the average pairwise similarity is calculated. The similarity score for a particular model is then again the average of the average pairwise similarity for all target words in the test set.

Let us illustrate this procedure with an example target word. In (1), the 5 most similar words for the target word *wedstrijd* 'game' are given (using a syntax-based model):

(1) *wedstrijd* 'game': *duel* 'duel', *toernooi* 'tournament', *race* 'race', *finale* 'final', *ronde*, 'round'

It is now straightforward to calculate the average pairwise similarity using a particular similarity measure and a particular value of k, e.g. $sim_{lin,3}(wedstrijd)$

$$= \frac{sim_{lin}(wedstrijd, duel) + sim_{lin}(wedstrijd, toernooi) + sim_{lin}(wedstrijd, race)}{3}$$
$$= \frac{.796 + .799 + .793}{3} = .796$$

As mentioned before, the score for the whole test set is then the average of these individual average pairwise similarities for all target words in the test.

5.3.4 Test set

For the construction of the evaluation's test set, we extracted the 10K most frequent nouns from the TWNC. From those nouns, we selected the ones that also appeared in the CORNETTO database. This amounts to a total test set of 5478 nouns.

A random baseline for the evaluation is provided in table 5.1. The baseline has been created by combining each of the 5478 nouns with 5 other nouns randomly selected from the same test set. The table provides means and standard deviations calculated over 5 different random runs.

	similarity	$k = 1$	$k = 3$	$k = 5$
1	sim_{wp}	$.128 \pm 2.60 \times 10^{-3}$	$.126 \pm 1.53 \times 10^{-3}$	$.126 \pm 8.08 \times 10^{-4}$
2	sim_{lin}	$.164 \pm 9.62 \times 10^{-4}$	$.163 \pm 6.02 \times 10^{-4}$	$.163 \pm 7.17 \times 10^{-4}$

Table 5.1: CORNETTO similarity scores for the random baseline

		wu & palmer's similarity			lin's similarity		
	model	$k = 1$	$k = 3$	$k = 5$	$k = 1$	$k = 3$	$k = 5$
1	w_{par}	.337	.302	.284	.330	.302	.287
2	w_{art}	**.379**	**.331**	**.309**	**.354**	**.320**	**.304**

Table 5.2: Evaluation results for different document sizes in the document-based model

5.4 Results

5.4.1 Document-based model

In this section, we investigate the aptness of document-based models for the induction of semantic similarity. First, the influence of different document sizes is examined. In the second part, we inspect the influence of different weighting parameters. The third part then looks at the influence of dimensionality reduction algorithms.

For the construction of document-based models, a smaller subset of the Twente Nieuws Corpus has been used, amounting to ± 3-6 years of newspaper text (dependent on the document size). When more data is used, the models become too large, so that computations become prohibitively expensive. All models have been constructed using 30K nouns as target words, and 500K documents as features. No cut-offs have been applied, so all frequency data is taken into account. Cosine has been used as a similarity measure (this holds for all evaluations performed in this chapter).

Document size

Table 5.2 gives the result for two different document sizes – paragraph (w_{par}) and article (w_{art}). The models both use a combination of local logarithmic weighting and global entropy weighting, and an SVD dimensionality reduction to 300 dimensions. These are the best scoring parameters for both document sizes in the document based model (cfr. table 5.5 and 5.6).

The results indicate that a model using a larger document size is able to achieve better results. This is most likely due to data sparseness in the model with small

		wu & palmer's similarity			lin's similarity		
	model	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
1	A_{freq}	.315	.272	.259	.318	.284	.272
2	$A_{logfreq}$	**.323**	**.285**	**.267**	**.325**	**.294**	**.279**
3	A_{ent}	.315	.271	.259	.318	.284	.272
4	A_{logent}	.323	.285	.267	.325	.294	.279

Table 5.3: Evaluation results for different weighting parameters in the document-based model with paragraph size

document size: the model uses a (rather small) paragraph size, and is not able to encode as many collocations as the model with a larger document size. Also, the model with a small document size uses less corpus data due to the feature limit of 500K documents.

Weighting

Table 5.3 and 5.4 give the results for the document-based model with regard to different weighting parameters; the former table makes use of paragraphs as document size, whereas the latter uses articles. Four different weighting options are considered. The first model uses no weighting whatsoever (i.e. the identity function is applied as a weighting function). The second model uses local logarithmic weighting, and no global weighting. The third model uses global entropy weighting, and no local weighting. And the fourth model uses a combination of a local logarithmic weighting and a global entropy weighting. The four different models can be formalized as follows:

$$\begin{aligned} A_{freq}(i,j) &= f_{ij} \\ A_{logfreq}(i,j) &= 1+\log(f_{ij}) \\ A_{ent}(i,j) &= f_{ij}(1+\textstyle\sum_{k=1}^{n}\frac{p_{ik}\log(p_{ik})}{log(n)}), p_{ik}=\frac{f_{ik}}{\sum_{l=1}^{n} f_{il}} \\ A_{logent}(i,j) &= f_{ij}(1+\textstyle\sum_{k=1}^{n}\frac{p_{ik}\log(p_{ik})}{log(n)}), p_{ik}=\frac{1+\log(f_{ik})}{\sum_{l=1}^{n} 1+\log(f_{il})} \end{aligned} \tag{5.10}$$

For both document sizes, we see that logarithmic weighting improves the results a little. The application of global entropy weighting does not improve the results – neither with the bare frequency counts nor with the logarithmically weighted frequency counts. This is probably due to the vast sparseness of the document-based feature vectors. The entropy weighting is able to adapt the frequency value according to a word's importance in the document, but similar words simply do not co-occur sufficiently frequently to be able to make proper similarity calculations.

		wu & palmer's similarity			lin's similarity		
	model	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
1	A_{freq}	.331	.298	.280	.335	.310	.296
2	$A_{logfreq}$	**.348**	**.302**	**.285**	**.350**	**.317**	**.304**
3	A_{ent}	.331	.298	.280	.335	.310	.296
4	A_{logent}	.348	.302	.285	.350	.317	.304

Table 5.4: Evaluation results for different weighting parameters in the document-based model with article size

			wu & palmer's similarity			lin's similarity		
A	M	d	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
A_{freq}	M_{svd}	50	.283	.257	.243	.292	.270	.259
		100	.302	.275	.261	.308	.285	.274
		300	.327	.287	.274	.324	.292	.282
$A_{logfreq}$	M_{svd}	50	.282	.260	.248	.291	.273	.263
		100	.310	.282	.269	.312	.290	.279
		300	.331	.295	.279	.327	.297	.284
A_{logent}	M_{svd}	50	.303	.279	.263	.306	.283	.272
		100	.325	.293	.276	.324	.297	.284
		300	**.337**	**.302**	**.284**	**.330**	**.302**	**.287**

Table 5.5: Evaluation results for different dimensionality reduction parameters in the document-based model with paragraph size

Dimensionality reduction

Table 5.5 gives the results for the document-based model with paragraph size (weighted according to three different weighting functions) combined with singular value decomposition. Table 5.6 gives the same result for the document-based model with article size. Non-negative matrix factorization has not been applied, because the computation of those factorizations was computationally too expensive.[10]

The results indicate that the models using singular value decomposition are able to improve upon the original models (with $d = 300$), albeit only to a small extent. The improvement is visible for all different weighting parameters and for both document

[10]Computationally too expensive in this case means that the factorization could not be carried out within the memory limit of 4GB.

			wu & palmer's similarity			lin's similarity		
A	M	d	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
A_{freq}	M_{svd}	50	.311	.278	.264	.307	.279	.269
		100	.323	.288	.273	.318	.289	.277
		300	.349	.304	.285	.338	.303	.288
$A_{logfreq}$	M_{svd}	50	.322	.291	.274	.313	.288	.275
		100	.341	.305	.286	.330	.301	.286
		300	.357	.317	.298	.341	.311	.296
A_{logent}	M_{svd}	50	.320	.289	.274	.312	.286	.275
		100	.344	.308	.289	.333	.304	.290
		300	**.379**	**.331**	**.309**	**.354**	**.320**	**.304**

Table 5.6: Evaluation results for different dimensionality reduction parameters in the document-based model with article size

sizes, although some parameters bring about a larger improvement. In general, the models using articles as a document size get a better improvement. Also, the model using logarithmic weighting combined with entropy weighting is improving more than the other models. This makes the model using log-entropy weighting combined with SVD the best scoring model for the document-based approach.[11]

Summary

The performance of the document-based model for the extraction of semantic similarity is rather low. The use of larger document sizes seems to make the model more robust with respect to data sparseness. Dimensionality reduction (SVD) improves the results a little, but the results remain rather low. Log-entropy weighting achieves the best performance.

5.4.2 Window-based models

Introduction

In this section, we will investigate the performance of different window-based models on the task of extracting semantic similarity.[12] In the first part, we investigate the

[11] Note that these are the parameters that are commonly used in latent semantic analysis, though latent semantic analysis takes all words into consideration. In the results above, we only consider nouns.

[12] Recall that in these models, we use the words that appear within a certain window size around the target word. These models are also known as bag of words models.

relation between the size of the model (in terms of features) and the model quality. The second part will focus on the influence of the context window size. The third part describes the influence of different weighting parameters, and the fourth part evaluates the influence of different dimensionality reduction algorithms. In the last part, we draw some conclusions with regard to the different parameter settings in window-based distributional similarity models.

Model size/feature selection

The performance of word space models improves when more data is taken into account. This holds a fortiori for the model's feature space: when more features are added to the model, the model's grasp of semantic similarity improves. There is, however, one big caveat, particularly when using window-based models. The models tend to become very large quickly when more features are taken into account. Usually, the most frequent words are used as the model's feature set – applying a threshold or cut-off to limit the feature size. When a large feature set is used, the computations and memory requirements often become prohibitively expensive. This is certainly true when high frequency function words are included in the feature set as well. In this section, we therefore want to investigate the influence of the nature and size of the feature set.

Table 5.7 shows the results for seven different bag of words models. The first three models have been built with a window size of 1 (w_1), respectively using a feature set of 100K words (T_{100K}), 2K words (T_{2K}), and 2K words excluding a list of stop words[13] (T_{2K}, $\neg$stop). Model 4 and 5 have been built with a window size of 2 (w_2), using feature sets of 100K words (T_{100K}) and 2K words (T_{2K}). The last two models both use a sentence window (w_{sent}) and a feature set of 2K words (T_{2K}), but the latter's feature set does not include stop words. All models have been weighted with pointwise mutual information.

The results indicate that the models with larger feature sets indeed perform better: the best model is the first model (large feature set and a window size of 1), closely followed by the fourth model (large feature set and context size of 2). Note, however, that the same models with a smaller feature set (models 2 and 5) attain a relatively high score as well; the difference between the respective models gets smaller as the window size increases. At the same time, the models with smaller feature sets are about half the size of their counterparts with large feature sets (in terms of non-zero matrix values).

The removal of function words again hurts performance, but the models are still able to attain reasonable scores, while at the same time again significantly reducing the

[13] The stop list consists of 'contentless' function words.

	description	wu & palmer's similarity $k=1$	$k=3$	$k=5$	lin's similarity $k=1$	$k=3$	$k=5$
1	w_1, T_{100K}	**.633**	**.561**	**.526**	**.541**	**.485**	**.456**
2	w_1, T_{2K}	.607	.540	.507	.514	.463	.436
3	w_1, T_{2K}, ¬stop	.589	.521	.487	.510	.457	.428
4	w_2, T_{100K}	.627	.551	.516	**.541**	.483	.455
5	w_2, T_{2K}	.620	.546	.511	.535	.477	.450
6	w_{sent}, T_{2K}	.547	.465	.428	.477	.420	.391
7	w_{sent}, T_{2K}, ¬stop	.528	.447	.411	.465	.406	.378

Table 5.7: Evaluation results for different model size parameters in the bag of words model

size of the model: the models that filter out stop words are again about half the size of their non-filtered counterparts.

From the results in table 5.7, we conclude that it is best to use as many features as possible; when the use of many features is computationally too expensive, models with smaller feature sizes still perform well compared to their counterparts with larger feature size. The same goes for the removal of high frequency function words: function words should be included if possible, but if the model size grows too large, they may be removed without severely hurting the model's performance.

Window size

In the next paragraphs, we will look at the influence of window size on the model's performance. We will use window sizes of 1, 2, 5, and 10 (both left and right around the target word, but not crossing sentence boundaries), as well as the sentence and the paragraph in which the target word appears. All models use a feature set of size 2K without stop word removal, and all have been weighted with pointwise mutual information. Table 5.8 gives the results for the varying window sizes.

The model that uses a window size of 2 (w_2) scores best. In general, smaller window sizes perform better for the induction of semantic similarity. The model with a window size of 1 (w_1) performs worse than w_2, but this is probably due to data sparseness: the small window combined with the small feature set does not allow the model to be sufficiently informative. The results in table 5.7 indicate that a model with a window size of 1 performs better when more features are taken into account.[14]

[14] A window size of 2K has been used to keep all parameters but the window size equal. Using a larger

		wu & palmer's similarity			lin's similarity		
	window size	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
1	w_1	.607	.540	.507	.514	.463	.436
2	w_2	**.620**	**.546**	**.511**	**.535**	**.477**	**.450**
3	w_5	**.620**	.541	.500	.532	.474	.442
4	w_{10}	.578	.497	.457	.502	.445	.413
5	w_{sent}	.547	.465	.428	.477	.420	.391
6	w_{par}	.442	.377	.349	.404	.357	.336

Table 5.8: Evaluation results for different window sizes in the bag of words model

Weighting

The results below show the influence of different weighting options. Four different weighting options are considered. The first model uses no weighting whatsoever (i.e. the identity function is applied as a weighting function). The second model uses local logarithmic weighting, and no global weighting. The third model uses global pointwise mutual information weighting, and no local weighting. And the fourth model uses a combination of a logarithmic weighting and pointwise mutual information. The four different models can be formalized as follows:

$$\begin{array}{ll} A_{freq}(i,j) = f_{ij} & A_{logfreq}(i,j) = 1+\log f_{ij} \\ A_{pmi}(i,j) = \log \frac{p(f_{ij})}{p(f_i)p(f_j)} & A_{logpmi}(i,j) = \log \frac{p(1+\log f_{ij})}{p(\sum_j 1+\log f_{ij})p(\sum_i 1+\log f_{ij})} \end{array} \tag{5.11}$$

In order to examine the influence of different weighting options on different window sizes, results are given for a small window size (w_2, table 5.9) and a large window size (w_{par}, table 5.10).

For a window size of 2, pointwise mutual information (A_{pmi}) clearly achieves the best result. Both logarithmic weighting ($A_{logfreq}$) and the combination of logarithmic weighting and PMI (A_{logpmi}) improve upon the bare frequency counts, but both methods perform worse than A_{pmi}.

For the models using a paragraph as context window, the results look different. A_{pmi} still achieves the best result, but the improvement on the bare frequency counts is smaller. Furthermore, $A_{logfreq}$ and A_{logpmi} do not improve upon the bare frequency counts.

feature set would have made the models with larger windows prohibitively expensive.

		wu & palmer's similarity			lin's similarity		
	model	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
1	A_{freq}	.425	.372	.351	.360	.318	.300
2	$A_{logfreq}$	.555	.492	.462	.481	.427	.402
3	A_{pmi}	**.620**	**.546**	**.511**	**.535**	**.477**	**.450**
4	A_{logpmi}	.542	.484	.454	.479	.430	.406

Table 5.9: Evaluation results for different weighting parameters in the bag of words model with a window of 2

		wu & palmer's similarity			lin's similarity		
	model	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
1	A_{freq}	.374	.330	.310	.349	.315	.302
2	$A_{logfreq}$	.340	.302	.284	.337	.306	.292
3	A_{pmi}	**.442**	**.377**	**.349**	**.404**	**.357**	**.336**
4	A_{logpmi}	.362	.314	.296	.353	.316	.302

Table 5.10: Evaluation results for different weighting parameters in the bag of words model with a paragraph window

The results indicate that, for the computation of semantic similarity, a weighting function applying pointwise mutual information to bare frequency counts (A_{pmi}) is overall the best weighting function for bag of words models.

Dimensionality reduction

In this section, we look at the influence of different dimensionality reduction algorithms on the models' performance. The two dimensionality reductions explained in chapter 3 – singular value decomposition and non-negative matrix factorization – are used. For each of these, the models are reduced to 50, 100 and 300 dimensions. Furthermore, this setup is combined with three different weighting functions – A_{freq}, $A_{logfreq}$, and A_{pmi} – in order to check whether a particular weighting is more apt to be used with a particular dimensionality reduction. The whole evaluation is carried out for a small window size (w_2, table 5.11) and a large window size (w_{par}, table 5.12).

From table 5.11, we see that the best scoring model is the one using pointwise mutual information as a weighting function, and a singular value decomposition to 300 dimensions as dimensionality reduction algorithm. However, the best performing

			wu & palmer's similarity			lin's similarity		
A	M	d	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
A_{freq}	M_{svd}	50	.395	.350	.329	.333	.299	.282
		100	.411	.361	.339	.347	.307	.291
		300	.421	.369	.347	.355	.316	.297
	M_{nmf}	50	.443	.404	.388	.372	.340	.327
		100	.481	.435	.411	.406	.367	.347
		300	.497	.435	.410	.421	.370	.350
$A_{logfreq}$	M_{svd}	50	.481	.438	.421	.414	.375	.360
		100	.506	.464	.439	.435	.397	.376
		300	.533	.483	.455	.461	.417	.392
	M_{nmf}	50	.434	.400	.385	.377	.346	.333
		100	.451	.414	.397	.395	.361	.346
		300	.442	.407	.391	.387	.355	.341
A_{pmi}	M_{svd}	50	.540	.489	.465	.461	.420	.401
		100	.574	.513	.487	.490	.442	.420
		300	**.606**	**.539**	**.507**	**.518**	**.466**	**.440**
	M_{nmf}	50	.423	.398	.385	.369	.347	.336
		100	.459	.417	.398	.397	.364	.349
		300	.473	.426	.405	.418	.378	.360
A_{pmi}	–	–	.620	.546	.511	.535	.477	.450

Table 5.11: Evaluation results for different dimensionality reduction parameters in the bag of words based model with a window of 2

factorized model does not improve on its non-factorized counterpart (shown in the last line for reasons of comparison). For small windows, semantic similarity calculations do not benefit from dimensionality reduction algorithms.

Table 5.11 shows a number of additional striking characteristics that require our attention. With regard to SVD, the models that perform best are the ones that use PMI as a weighting function. Using logarithmic weighting also improves on the bare frequency data; the use of bare frequency data yields the worst performing models. With regard to NMF, the situation is quite different. The best performing models are the ones that do not apply weighting; using logarithmic weighting or PMI yields models that perform worse. The use of the Kullback-Leibler divergence as the NMF's objective function makes weighted models perform worse than their non-weighted counterparts. In general, though, models using SVD reach better results than models using NMF.

			wu & palmer's similarity			lin's similarity		
A	M	d	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
A_{freq}	M_{svd}	50	.331	.288	.273	.317	.285	.273
		100	.352	.311	.290	.333	.302	.286
		300	.371	.328	.305	.345	.315	.297
	M_{nmf}	50	.337	.304	.284	.315	.290	.277
		100	.360	.316	.291	.338	.304	.285
		300	.377	.324	.304	.350	.310	.295
$A_{logfreq}$	M_{svd}	50	.323	.289	.272	.316	.289	.276
		100	.336	.297	.281	.327	.298	.284
		300	.337	.304	.285	.331	.304	.289
	M_{nmf}	50	.313	.288	.271	.309	.288	.276
		100	.333	.295	.277	.327	.295	.281
		300	.332	.296	.281	.327	.299	.286
A_{pmi}	M_{svd}	50	.389	.333	.313	.354	.313	.298
		100	.418	.357	.332	.379	.333	.314
		300	**.443**	**.376**	**.351**	**.402**	**.351**	**.331**
	M_{nmf}	50	.320	.284	.270	.303	.275	.263
		100	.356	.314	.297	.328	.298	.285
		300	.391	.335	.312	.360	.319	.301
A_{pmi}	–	–	.442	.377	.349	.404	.357	.336

Table 5.12: Evaluation results for different dimensionality reduction parameters in the bag of words based model with a paragraph window

Table 5.12 shows results that are similar to the ones given in table 5.12. The best scoring model is again the one using PMI as a weighting function, and an SVD to 300 dimensions. Again it does not improve on its non-factorized counterpart; in the best case it is able to achieve a similar score. Bag of words models using a large context window do not provide good models for semantic similarity calculations, and the application of dimensionality reduction algorithms does not improve on those results.

The tendencies for SVD are again the same, the only difference being that logarithmic weighting does not bring about an improvement. With regard to NMF, simple frequencies again yield the best models.

Summary

For the extraction of tight semantic similarity, smaller context windows provide the most informative features. The more features that are taken into account, the better performance the models reach. Still, the models are able to attain a reasonable performance when a smaller number of features is taken into account. When the goal is to extract tight semantic similarity, it is better not to exclude a stop list (though it is a possibility that does not severely decrease performance when the models get too large). With regard to dimensionality reduction algorithms, no model improves upon its non-reduced counterpart, but the SVD models combined with A_{pmi} are able to approximate their non-reduced counterparts very closely within a highly reduced vector space.

5.4.3 Syntax-based models

Introduction

In this section, we look at the performance of syntax-based models for the calculation of semantic similarity. All syntax-based models have been constructed using dependency relations extracted from the Twente Nieuws Corpus (parsed with the Dutch dependency parser ALPINO).[15] The dependency relations that are used to construct the models are listed in table 1.2 on page 22.

In the next paragraphs, we extensively examine the influence of different weighting functions models. In all subsequent evaluations, we use a syntax-based model with a noun cutoff of 20 (a noun has to appear with 20 different features) and a triple cutoff of 2 (a particular noun-feature combination has to appear twice). This amounts to a model that contains 30K nouns cross-classified by 170K features. In practice, this gives a good balance between model size and model quality. As a similarity measure, cosine is used.

Weighting

We have tested the syntactic model with the same weighting functions that have been applied to the window-based model: no weighting (A_{freq}), local logarithmic weighting ($A_{logfreq}$), pointwise mutual information (A_{pmi}) and a combination of a logarithmic function and pointwise mutual information (A_{logpmi}). A formal overview of the different functions is given on page 74.

[15] An example feature for the noun *apple* might be 'eat_{obj}'.

		wu & palmer's similarity			lin's similarity		
	model	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
1	A_{freq}	.480	.415	.385	.414	.364	.339
2	$A_{logfreq}$	**.648**	.578	.546	**.555**	.502	.474
3	A_{pmi}	.640	**.584**	**.554**	.546	**.504**	**.480**
4	A_{logpmi}	.631	.576	.546	.540	.499	.474

Table 5.13: Evaluation results for different weighting parameters in the syntax-based model

The results are presented in table 5.13; they indicate that the performance of the different weighting models is mixed. For $k = 1$, simple logarithmic weighting scores best ($sim_{wp} = .648$ and $sim_{lin} = .555$). For the other values of k, the model that uses only pointwise mutual information is the most successful. All weighting models perform better than the model using simple frequency counts, but a combination of logarithmic weighting and PMI does not improve over the models that use either of those.

Dimensionality reduction

To evaluate the influence of dimensionality reduction algorithms on the syntax-based model, we use a setup that is similar to the one used with the window-based model. Again, the two dimensionality reductions explained in chapter 3 – singular value decomposition and non-negative matrix factorization – have been applied. For each of these, the models are reduced to 50, 100 and 300 dimensions. This setup is combined with three different weighting functions – A_{freq}, $A_{logfreq}$, and A_{pmi} – to check whether a particular weighting is more apt to be used with a particular dimensionality reduction.

No dimensionality reduction model is able to improve upon the score of its non-reduced counterpart, so dimensionality reduction does not help in getting better similarity models with regard to CORNETTO similarity. We do notice that some models (notably the SVD models reduced to 300 dimensions using local logarithmic weighting and global pointwise mutual information) are able to closely approach their non-reduced counterparts (for $k = 1$, $sim_{wp} = .610$ ($\leftrightarrow .648$), $sim_{lin} = .520$ ($\leftrightarrow .555$) with $A_{logfreq}$, and $sim_{wp} = .604$ ($\leftrightarrow .640$), $sim_{lin} = .510$ ($\leftrightarrow .546$) with A_{pmi}). The last line of the table shows the non-reduced PMI model for comparison.

With regard to the differences between both dimensionality reduction algorithms, the tendencies are similar to the window-based model. For SVD, the best performing

			wu & palmer's similarity			lin's similarity		
A	M	d	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
A_{freq}	M_{svd}	50	.373	.339	.325	.320	.294	.283
		100	.416	.374	.355	.355	.322	.305
		300	.466	.408	.384	.396	.351	.332
	M_{nmf}	50	.409	.393	.385	.338	.322	.315
		100	.454	.430	.417	.376	.356	.347
		300	.502	.455	.433	.423	.388	.369
$A_{logfreq}$	M_{svd}	50	.552	.503	.482	.468	.425	.408
		100	.576	.527	.499	.487	.448	.426
		300	**.610**	.553	.524	**.520**	.474	.449
	M_{nmf}	50	.407	.392	.388	.342	.327	.322
		100	.463	.441	.427	.389	.367	.355
		300	.508	.468	.452	.432	.400	.385
A_{pmi}	M_{svd}	50	.548	.500	.482	.459	.422	.406
		100	.572	.525	.505	.483	.446	.427
		300	.604	**.560**	**.535**	.510	**.476**	**.453**
	M_{nmf}	50	.368	.359	.353	.318	.304	.298
		100	.413	.393	.383	.350	.331	.322
		300	.443	.414	.405	.382	.356	.347
A_{pmi}	–	–	.640	**.584**	**.554**	.546	**.504**	**.480**

Table 5.14: Evaluation results for different dimensionality reduction parameters in the syntax-based model

model is A_{pmi}, together with $A_{logfreq}$. For NMF, the best scoring models are the A_{freq} and $A_{logfreq}$ models; the A_{pmi} models scores considerably worse.

Summary

The class of syntax-based models gives the best performance for the extraction of semantic similarity. Pointwise mutual information seems to provide the best weighting, although simple logarithmic weighting yields quite good results as well. Dimensionality reduction algorithms do not improve upon the results, but singular value decomposition is able to approximate the original model.

		wu & palmer's similarity			lin's similarity	
model	$k=1$	$k=3$	$k=5$	$k=1$	$k=3$	$k=5$
document-based	.379	.331	.309	.354	.320	.304
window-based	.633	.561	.526	.541	.485	.456
syntax-based	**.648**	**.584**	**.554**	**.555**	**.504**	**.480**

Table 5.15: Comparison of the best scoring models for each class

5.4.4 Conclusion

Table 5.15 compares the best results of the three different classes that have been evaluated in this chapter.

The syntax-based models yield the best results (with $A_{logfreq}$ for $k = 1$ and A_{pmi} for $k = 3,5$) closely followed by the window-based models (using a small window of $w = 1$ and A_{pmi} weighting). The best performing document-based model (using SVD reduced to 300 dimensions with article size and A_{logent} weighting) scores worse than the other two models.

Dimensionality reduction algorithms are - to a small extent - beneficial for document-based models. Singular value decomposition (in combination with logentropy-weighting) is able to improve upon the original data. The window-based and syntax-based models do not benefit from dimensionality reduction algorithms; the window-based model with dimensionality reduction achieves results similar to its non-factorized counterpart, while the application of dimensionality reduction is detrimental when using a syntax-based model. The results of the models using non-negative matrix factorization are disappointing: in all cases, either SVD or the non-factorized models reach better results.

For the extraction of tight, synonym-like similarity, we conclude that a syntax-based model is the preferred model.

Chapter 6

Evaluation of Cluster Quality

6.1 Introduction

In this chapter, we evaluate the quality of a word space model by comparing the output of a clustering algorithm (that calculates the similarity between words through the word space model) to a gold standard categorization. For this purpose, we will make use of the lexical categorization tasks that have been defined for the workshop 'Bridging the gap between semantic theory and computational simulations', organized at ESSLLI 2008 in Hamburg.[1]

6.2 Methodology

6.2.1 Clustering tasks

The workshop provides two different noun categorization (clustering) tasks:

1. CONCRETE NOUN CATEGORIZATION – In the concrete noun categorization task, the goal is to cluster 44 concrete nouns in a number of classes on various levels of generality:
    - 2-WAY CLUSTERING – cluster nouns in two top classes *natural* and *artefact*;
    - 3-WAY CLUSTERING – cluster nouns in three classes *animal*, *vegetable* and *artefact*;

[1] `http://wordspace.collocations.de/doku.php/esslli:start`

- 6-WAY CLUSTERING – cluster nouns in six classes *bird*, *groundAnimal*, *fruitTree*, *green*, *tool* and *vehicle*.

2. ABSTRACT/CONCRETE NOUN DISCRIMINATION – The evaluation of algorithms discriminating between abstract and concrete nouns consists of three parts:

   - In the first part, 30 nouns (15 with high concreteness value and 15 with low concreteness value) are clustered in two clusters, HI and LO;
   - in the second part, 10 nouns with average concreteness value are added to the two-way clustering, to see whether they end up in the HI or the LO cluster;
   - in the third part, a three-way clustering of the 40 nouns (15 HI, 10 ME, 15 LO) is performed.

We will investigate the performance of the different models with regard to these tasks. With regard to the second task, we will only assess the first part (the ability of the models to discriminate between concrete and abstract nouns). Part 2 and 3 will not be considered here.[2]

The test sets provided by the workshop are in English. They have been translated into Dutch by three translators, and – when multiple translations were found – the majority translation has been taken as the final one. The frequencies of the Dutch words are by and large comparable to the frequencies of their English counterparts. The gold standard categorization for both tasks is given in appendix A (page 159).

6.2.2 Clustering algorithm

The clustering solutions have been computed with the clustering program CLUTO (Karypis, 2003). The k-way solution is computed by performing a sequence of $k-1$ repeated bisections. This means that the matrix is first clustered into two groups, and then one of the groups is selected and bisected further. This process continues until the desired number of clusters is found. In each step, the cluster is bisected so that the resulting 2-way clustering solution maximizes the clustering optimization function given in equation 6.1.

$$I_2 = \sum_{k}^{i=1} \sqrt{\sum_{v,u \in S_i} \cos(v,u)} \tag{6.1}$$

[2]The interested reader is referred to Van de Cruys (2008a).

where k is the number of clusters, S_i is the set of nouns assigned to the ith cluster, v and u are two nouns, and $\cos(v,u)$ is the cosine similarity of those nouns. After computing the initial clustering, the overall solution is globally optimized. In practice, this ensures the algorithm computes the same clustering solution for the same set of objects.[3]

6.3 Evaluation framework

For the evaluation of cluster quality, we use two different measures: ENTROPY and PURITY (Zhao and Karypis, 2001). These are the standard measures of cluster quality available in CLUTO. Entropy measures how the various semantic classes are distributed within each cluster. Given a particular cluster S_r of size n_r, the entropy of this cluster is defined to be

$$E(S_r) = -\frac{1}{\log q}\sum_{i=1}^{q}\frac{n_r^i}{n_r}\log\frac{n_r^i}{n_r} \tag{6.2}$$

where q is the number of classes in the dataset, and n_r^i is the number of documents of the ith class that were assigned to the rth cluster. The entropy of the entire clustering is then the sum of the individual cluster entropies weighted according to the cluster size:

$$entropy = \sum_{r=1}^{k}\frac{n_r}{n}E(S_r) \tag{6.3}$$

The clustering solution is perfect if clusters only contain words from one single class; in that case the entropy of the clustering solution is zero. Smaller entropy values indicate better clustering solutions.

Using the same mathematical definitions, the purity of a cluster is defined as:

$$Pu(S_r) = \frac{1}{n_r}\max_i n_r^i \tag{6.4}$$

The purity gives the fraction of the overall cluster size that the largest class of words assigned to that cluster represents. The purity of the clustering solution is then again the weighted sum of the individual cluster purities:

$$purity = \sum_{r=1}^{k}\frac{n_r}{n}Pu(S_r) \tag{6.5}$$

Larger purity values indicate better clustering solutions.

[3]The algorithm is selected by choosing the 'rbr' clustering method option in CLUTO.

6.4 Results

The models in this chapter are built in exactly the same way as the models presented in chapter 5, in which the semantic similarity is evaluated. After building the models, the feature vectors of the nouns involved in the categorization tasks have been extracted from the models, and fed to the clustering algorithm.

6.4.1 Concrete noun categorization

Document based

Table 6.1 shows the clustering results for the document-based model with regard to different weighting parameters. Results are given for smaller paragraph (w_{par}) and larger article (w_{art}) document sizes.

		2-way		3-way		6-way	
method	docsize	entropy	purity	entropy	purity	entropy	purity
A_{freq}	w_{par}	.970	.591	.739	.591	.548	.523
	w_{art}	**.505**	**.886**	**.587**	**.727**	**.318**	**.750**
$A_{logfreq}$	w_{par}	.562	.864	.757	.614	.408	.659
	w_{art}	.611	.841	.685	.614	.392	.659
A_{ent}	w_{par}	.970	.591	.739	.591	.548	.523
	w_{art}	**.505**	**.886**	**.587**	**.727**	**.318**	**.750**
A_{logent}	w_{par}	.562	.864	.757	.614	.408	.659
	w_{art}	.611	.841	.685	.614	.392	.659

Table 6.1: Clustering results for concrete noun categorization: bag of words approach with different weighting parameters

The best results are reached with simple frequency information (using articles as document size); none of the weightings is able to improve upon the simple frequency values. Still, the best scoring model does not yield a very good clustering solution.

Table 6.2 gives the results for the document-based model combined with dimensionality reduction (SVD). Results are given for three different weighting functions (A_{freq}, $A_{logfreq}$ and A_{logent}), using both paragraphs and articles as document sizes.

The results for the document-based model in combination with dimensionality reduction are mixed. On the one hand, the models are able to attain better results for the 3-way and 6-way clustering. The model using a combination of logarithmic smoothing

			2-way		3-way		6-way	
A	M	w	entropy	purity	entropy	purity	entropy	purity
A_{freq}	M_{svd}	w_{par}	.983	.545	.241	.886	.330	.682
		w_{art}	.930	.614	**.179**	**.932**	.318	.682
$A_{logfreq}$	M_{svd}	w_{par}	.991	.545	.241	.886	.343	.682
		w_{art}	.954	.591	.213	.909	.310	.659
A_{logent}	M_{svd}	w_{par}	**.994**	**.545**	.265	.864	.354	.659
		w_{art}	.930	.614	**.179**	**.932**	**.292**	**.682**

Table 6.2: Clustering results for concrete noun categorization: document-based approach with dimensionality reduction

and entropy weighting generally gets the best results. The two-way clustering, however, is consistently worse.

The general tendency of these results is similar to the document-based results presented in the last chapter: log-entropy weighting combined with singular value decomposition reaches the best results, but generally speaking document-based models do not attain a very good performance.

Bag of words

Table 6.3 gives the clustering results for the bag of words model with regard to different weighting parameters, for a small context window of two words (w_2) as well as a large paragraph window (w_{par}).

The model that uses a small context window together with pointwise mutual information clearly performs best. These results are in line with the results that are presented in the last chapter: small context windows tend to attain better results for semantic similarity extraction, and the combination with PMI shows the best performance.

It is interesting to compare the different clusters found by the models using a small window and the models using a large window. Here we compare the models using PMI for the 6-way clustering. For the model that uses the small window (w_2), the confusion mostly arises from the ANIMAL class that is wrongly split up:

- *eend* 'duck', *hond* 'dog', *kat* 'cat', *kip* 'chicken', *koe* 'cow', *leeuw* 'lion', *olifant* 'elephant', *pauw* 'peacock', *varken* 'pig', *zwaan* 'swan'
- *arend* 'eagle', *pinguïn* 'penguin', *schildpad* 'turtle', *slak* 'snail', *uil* 'owl'

		2-way		3-way		6-way	
method	window	entropy	purity	entropy	purity	entropy	purity
A_{freq}	w_2	.714	.715	.422	.727	.556	.477
	w_{par}	.966	.544	.638	.659	.380	.659
$A_{logfreq}$	w_2	.924	.659	.887	.477	.375	.659
	w_{par}	.922	.636	.866	.523	.729	.409
A_{pmi}	w_2	**.000**	**1.000**	**.213**	**.909**	**.206**	**.773**
	w_{par}	.911	.659	.541	.705	.377	.591
A_{logpmi}	w_2	.924	.659	.648	.614	.277	.727
	w_{par}	.871	.705	.650	.659	.472	.568

Table 6.3: Clustering results for concrete noun categorization: bag of words approach with different weighting parameters

The model using the large window (w_{par}), on the other hand, seems to find more topically related clusters:

- *koe* 'cow', *maïs* 'corn', *varken* 'pig'
- *aardappel* 'potato', *ananas* 'pineapple', *banaan* 'banana', *champignon* 'mushroom', *fles* 'bottle', *kers* 'cherry', *ketel* 'kettle', *kip* 'chicken', *kom* 'bowl', *lepel* 'spoon', *peer* 'pear', *sla* 'lettuce', *ui* 'onion'

The first cluster seems related to stock breeding, containing stock animals (*koe* 'cow' and *varken* 'pig') and a crop often used as feed (*maïs* 'corn'). The second cluster looks like a food-related cluster, containing fruit and vegetables, but also an animal often consumed as meat (*kip* 'chicken') and a number of kitchen utensils (*fles* 'bottle', *ketel* 'kettle', *kom* 'bowl', *lepel* 'spoon').

Table 6.4 gives the results for the bag of words model combined with dimensionality reduction algorithms (SVD and NMF). Results are given for three different weighting functions (A_{freq}, $A_{logfreq}$ and A_{pmi}), both for a small context window (w_2) and a large context window (w_{par}).

Again, the model using a small context window and PMI (in combination with SVD) generally seems to give the best performance. The result, however, does not exceed its non-factorized counterpart.

Another remarkable result is that the NMF model – in combination with simple frequency counts and a large window – is able to reach the best specific result for

			2-way		3-way		6-way	
A	M	w	entropy	purity	entropy	purity	entropy	purity
A_{freq}	M_{svd}	w_2	.714	.750	.422	.727	.549	.477
		w_{par}	.966	.545	.271	.886	.337	.682
	M_{nmf}	w_2	.492	.864	.431	.773	.329	.750
		w_{par}	.930	.614	**.179**	**.932**	.310	.705
$A_{logfreq}$	M_{svd}	w_2	.924	.659	.887	.477	.440	.614
		w_{par}	.922	.636	.719	.659	.607	.432
	M_{nmf}	w_2	.944	.636	.679	.591	.492	.591
		w_{par}	.896	.682	.617	.659	.617	.432
A_{pmi}	M_{svd}	w_2	**.000**	**1.000**	.223	.909	**.233**	**.818**
		w_{par}	.972	.568	.541	.705	.334	.682
	M_{nmf}	w_2	.881	.682	.550	.705	.414	.659
		w_{par}	.964	.591	.513	.727	.384	.614

Table 6.4: Clustering results for concrete noun categorization: bag of words approach with dimensionality reduction

the three-way clustering. Clearly, NMF is able to induce general semantic traits from the data, yielding a good clustering result. Upon inspection of the data, we see that good ANIMAL and ARTEFACT clusters are formed. A third clustering contains mostly VEGETABLEs, but again also food related ARTEFACT nouns such as *kom* 'bowl' and *lepel* 'spoon', and the food related ANIMAL noun *kip* 'chicken'.

Syntax-based

Table 6.5 gives the clustering results for the syntax-based model, again according to four different weighting parameters.

The results indicate that the models using PMI and LOGPMI perform best – though it should be noted that all syntax-based models attain quite good results. Remarkably, the model that uses only logarithmic weighting performs not so well (particularly with regard to the two-way clustering). This result is different compared to the semantic similarity results presented in the last chapter. Overall, using PMI seems to provide the most stable performance for the induction of semantic similarity using the syntax-based model.

Figure 6.1 shows the confusion matrix of the 6-way clustering for the models using PMI and LOGPMI; the matrix is the same for both models.

	2-way		3-way		6-way	
method	entropy	purity	entropy	purity	entropy	purity
A_{freq}	**.000**	**1.000**	.067	.977	.264	.750
$A_{logfreq}$	.784	.659	**.000**	**1.000**	.245	.750
A_{pmi}	**.000**	**1.000**	**.000**	**1.000**	**.153**	**.864**
A_{logpmi}	**.000**	**1.000**	**.000**	**1.000**	**.153**	**.864**

Table 6.5: Clustering results for concrete noun categorization: syntactic approach with different weighting parameters

cluster	bird	grou	frui	gree	tool	vehi
1	0	0	4	5	0	0
2	0	0	0	0	7	0
3	6	0	0	0	0	0
4	0	0	0	0	0	7
5	0	0	0	0	6	0
6	1	8	0	0	0	0

Figure 6.1: Confusion matrix of the 6-way clustering for the syntax-based model (PMI and LOGPMI).

Upon examining the results, the decisions made by the algorithm look quite reasonable:

- One bird ('chicken') is classified as *groundAnimal*;
- fruits and vegetables are assigned to one single cluster;
- the *tools* class is split up into two different clusters, so that a division is made between 'active' and 'passive' tools:
  - *beitel* 'chisel', *hamer* 'hammer', *mes* 'knife', *pen* 'pen', *potlood* 'pencil', *schaar* 'scissors', *schroevendraaier* 'screwdriver';
  - *beker* 'cup', *fles* 'bottle', *ketel* 'kettle', *kom*, 'bowl', *lepel* 'spoon', *telefoon* 'telephone'.

The only noun that may be considered wrongly classified is *slak* 'snail', which is classified as a bird.

It is interesting to note that the difference between fruit and vegetables nonetheless is present in the data. When clustering the words from the subsets *fruitTree* and *green* into two classes, they are properly split up:

- *kers* 'cherry', *banaan* 'banana', *peer* 'pear', *ananas* 'pineapple';
- *champignon* 'mushroom', *maïs* 'corn', *sla* 'lettuce', *aardappel* 'potatoe', ui 'oignon'.

In table 6.6, the results for the syntax-based model combined with dimensionality reduction algorithms (SVD and NMF) are shown. Again, three different weighting functions (A_{freq}, $A_{logfreq}$ and A_{pmi}) are evaluated.

		2-way		3-way		6-way	
A	M	entropy	purity	entropy	purity	entropy	purity
A_{freq}	M_{svd}	.902	.682	.457	.795	.430	.636
	M_{nmf}	.917	.659	.260	.864	.244	.750
$A_{logfreq}$	M_{svd}	.784	.659	**.000**	**1.000**	.238	.773
	M_{nmf}	.589	.795	**.000**	**1.000**	.235	.750
A_{pmi}	M_{svd}	**.000**	**1.000**	**.000**	**1.000**	**.206**	**.773**
	M_{nmf}	.589	.795	.067	.977	.266	.727

Table 6.6: Clustering results for concrete noun categorization: syntactic approach with dimensionality reduction

Again, we see the same tendencies that became apparent in the former chapter: SVD in combination with PMI gives the best performance, but the results are not as good as the PMI model that does not use any dimensionality reduction (notably, the 6-way clustering performs worse).

6.4.2 Abstract/concrete noun discrimination

In the abstract/concrete noun discrimination task, we try to separate 15 abstract nouns and 15 concrete nouns into two different clusters.

Document-based model

Table 6.7 presents the clustering results for the document-based model. Four different weighting functions have been applied, each with two different document sizes.

method	docsize	entropy	purity
A_{freq}	w_{par}	.362	.931
	w_{art}	**.296**	**.933**
$A_{logfreq}$	w_{par}	.579	.862
	w_{art}	.463	.900
A_{ent}	w_{par}	.362	.931
	w_{art}	**.296**	**.933**
A_{logent}	w_{par}	.579	.862
	w_{art}	.463	.900

Table 6.7: Clustering results for abstract/concrete noun discrimination: document-based approach with different weighting parameters

The document-based model is not able to make a good division between abstract and concrete nouns. Using a large document size (w_{art}) yields slightly better results. No particular weighting function gets better results than the simple frequency data.

Table 6.8 presents the results for the document-based model in combination with singular value decomposition. Using a singular value decomposition yields worse results than the use of the original models.

A	M	w	entropy	purity
A_{freq}	M_{svd}	w_{par}	.739	.724
		w_{art}	**.470**	**.867**
$A_{logfreq}$	M_{svd}	w_{par}	.739	.724
		w_{art}	.662	.767
A_{logent}	M_{svd}	w_{par}	.625	.793
		w_{art}	.604	.800

Table 6.8: Clustering results for abstract/concrete noun discrimination: document-based approach with dimensionality reduction

Window-based model

Table 6.9 shows the clustering results for the window-based approach, with regard to four different weighting parameters. Again, we see that PMI (and LOGPMI) – combined with a small context window – is able to reach the best performance. Also notice that PMI combined with a larger window is able to attain a reasonable result, with only one concrete noun (*leeuw* 'lion') being misclassified as an abstract noun.

method	window	entropy	purity
A_{freq}	w_2	.914	.633
	w_{par}	.715	.733
$A_{logfreq}$	w_2	1.000	.500
	w_{par}	.930	.567
A_{pmi}	w_2	**.000**	**1.000**
	w_{par}	.180	.967
A_{logpmi}	w_2	**.000**	**1.000**
	w_{par}	.470	.867

Table 6.9: Clustering results for abstract/concrete noun discrimination: bag of words approach with different weighting parameters

In table 6.10, three different weighting parameters are again combined with two different dimensionality reduction algorithms – both for small and large context windows.

A model with small context size, weighted with PMI and combined with SVD is able to attain a perfect result. Remarkable in these results is also the performance of the models factorized with non-negative matrix factorization. The NMF model with simple frequency data attains good results, with a perfect division for the model using a large context window; the NMF models with small context window using logarithmic weighting and pointwise mutual information are able to make a perfect divsion between abstract and concrete nouns as well.

Syntax-based model

In table 6.11, the clustering results for the syntax-based model are presented with regard to four different weighting parameters. The results indicate that the syntax-based model is able to make a good division between abstract and concrete nouns. Only the model

A	M	w	entropy	purity
A_{freq}	M_{svd}	w_2	.914	.633
		w_{par}	.604	.800
	M_{nmf}	w_2	.180	.967
		w_{par}	**.000**	**1.000**
$A_{logfreq}$	M_{svd}	w_2	1.000	.500
		w_{par}	.930	.567
	M_{nmf}	w_2	**.000**	**1.000**
		w_{par}	.715	.733
A_{pmi}	M_{svd}	w_2	**.000**	**1.000**
		w_{par}	.180	.967
	M_{nmf}	w_2	**.000**	**1.000**
		w_{par}	.296	.933

Table 6.10: Clustering results for abstract/concrete noun discrimination: bag of words approach with dimensionality reduction

based on simple frequencies is not able to make a decent division; the other models are able to induce a perfect clustering.

method	entropy	purity
A_{freq}	.985	.567
$A_{logfreq}$	**.000**	**1.000**
A_{pmi}	**.000**	**1.000**
A_{logpmi}	**.000**	**1.000**

Table 6.11: Clustering results for abstract/concrete noun discrimination: syntax-based approach with different weighting parameters

In table 6.12, the syntax-based model (weighted with three different weighting functions) is again combined with two different dimensionality reduction algorithms. The results of the factorized models reflect the performance of their non-factorized counterparts. The models that make a perfect cut without dimensionality reductions also make a perfect cut with the use of dimensionality reduction. Notice again, however, that NMF is able to strongly improve the results of the model using simple frequency

data.

A	M	entropy	purity
A_{freq}	M_{svd}	1.000	.500
	M_{nmf}	.180	.967
$A_{logfreq}$	M_{svd}	**.000**	**1.000**
	M_{nmf}	**.000**	**1.000**
A_{pmi}	M_{svd}	**.000**	**1.000**
	M_{nmf}	**.000**	**1.000**

Table 6.12: Clustering results for abstract/concrete noun discrimination: syntax-based approach with dimensionality reduction

6.5 Comparison of the different models

Examining the differences in cluster quality between the different models, we can draw some conclusions with regard to their ability to extract semantic similarity. Clearly, the document-based models are do not perform well. The models are not able to correctly categorize concrete nouns, nor are they able to make a proper division between abstract and concrete nouns. The window-based model performs better than the document-based model. The window size plays a significant role: models using a small window tend to perform better than the ones using a large window (using PMI as weighting function).[4] But the model that performs best is the syntax-based model. Different weighting functions perform well, but PMI consistently seems to yield good results. In general, the application of dimensionality reduction models does not yield better cluster quality.

[4]Notice however that the models with a large window size seem to grasp a kind of 'topical' similarity.

Chapter 7

Evaluation of Domain Coherence

7.1 Introduction

In this chapter, we will evaluate the ability of the various models to induce more loosely related, topically similar words. This will be done by evaluating the ability of the models to extract similar words that belong to the same semantic domains. The CORNETTO database has been augmented with semantic domain tags. By evaluating the models' ability to extract similar words that belong to the same semantic domain as the target word, we are able to evaluate a model's ability to induce topical similarity.

7.2 Evaluation framework

7.2.1 Semantic domains

Semantic domains are particular areas of human knowledge (such as POLITICS, MEDICINE or SPORTS) according to which words and senses can be classified. Words that belong to the same semantic domain (e.g. *doctor* and *hospital* are both in the MEDICINE domain) are topically related. This makes semantic domains perfectly suitable to investigate topical similarity or domain coherence.

Semantic domains are a part of the original English WordNet: it has been augmented with Wordnet Domain labels – a collection of 200 domain labels (tags) organized in a hierarchical structure (Magnini and Cavagli, 2000; Bentivogli et al., 2004). Each synset in WordNet is labeled with one or more labels. When there are no suitable labels for a particular synset, the label 'factotum' is assigned.

The CORNETTO database, used in our evaluation framework, has been augmented with domain labels as well. Domain labels from the English WordNet have automatically been added.[1] Moreover, domain labels from the VLIS database[2] have been automatically converted to WordNet domain labels. The CORNETTO domain labels are consulted with the PYCORNETTO interface.

7.2.2 Similarity measure

For each target word in the test set, the 10 most similar words according to a particular model are extracted. All possible domain labels for the target word are extracted from the CORNETTO database, and compared to all possible domain tags for the extracted similar words. The most frequent domain tag – that is also present in the tagset of the target word – is taken as the correct domain tag. The number of words that is topically coherent to the target word is then the number of words that have the correct tag, divided by all words for which a domain tag has been extracted. Formally,

$$sim_{topic} = \frac{\max_{t \in \text{tags}(s_0)} \sum_i \begin{cases} 1 & \text{if t in tags}(s_i) \\ 0 & \text{otherwise} \end{cases}}{i} \tag{7.1}$$

where s_0 is the target word, $s_{1...i}$ is the list of similar words present in CORNETTO with at least one domain tag, and $\text{tags}(s_i)$ is the set of domain tags for word s_i. We ignore words that are not found in the CORNETTO database.

Let us illustrate the topical similarity measure with an example. In table 7.1, 10 words are given that are topically related to the target word *zebrapad* 'pedestrian crossing', together with the domain tags extracted from CORNETTO. The target word *zebrapad* has one possible tag: TRANSPORT. We look up which of the similar words also has this tag as one of its domain tags, and divide the number of correctly tagged words by the total number of similar words. This gives us a topic similarity sim_{topic} of $\frac{8}{10} = .80$.

Note that, among the similar words of *zebrapad*, there are many words that are only loosely, topically related (e.g. *voetganger* 'pedestrian' and *fietser* 'cyclist'). Still, those words belong to the same TRANSPORT domain, so that a high sim_{topic} value is attributed to the set of words.

Another important fact to note is that words that are tightly similar are usually also topically related. In table 7.2, 10 words are given that are tightly similar (synonymous

[1] CORNETTO's synsets are aligned with the English WordNet synsets, which makes the mapping straightforward.

[2] Van Dale Lexical Information System; a database developed by Van Dale lexicographers.

	word	translation	domain tags
0	**zebrapad**	'pedestrian crossing'	TRANSPORT
1	voetganger	'pedestrian'	TRANSPORT
2	fietser	'cyclist'	CYCLING, TRANSPORT
3	stoplicht	'traffic light'	TOWN PLANNING, TRANSPORT
4	trottoir	'pavement'	TRANSPORT
5	stoep	'pavement'	BUILDING INDUSTRY, TRANSPORT
6	kruispunt	'crossing'	TRANSPORT
7	fietspad	'cycle track'	TRANSPORT
8	rotonde	'roundabout'	BUILDING INDUSTRY, TRANSPORT
9	lantaarnpaal	'lamppost'	BUILDING INDUSTRY, TOWN PLANNING
10	berm	'verge'	TOWN PLANNING

Table 7.1: Domain tags for the similar words of *zebrapad*, 'pedestrian crossing'

	word	translation	domain tags
0	**huis**	'house'	BUILDING INDUSTRY
1	woning	'house'	BUILDING INDUSTRY
2	pand	'building'	BUILDING INDUSTRY, TOWN PLANNING, ...
3	gebouw	'building'	BUILDING INDUSTRY, TOWN PLANNING
4	villa	'villa'	BUILDING INDUSTRY
5	kamer	'room'	ADMINISTRATION, BUILDING INDUSTRY, ...
6	huisje	'small house'	–
7	boerderij	'farm'	AGRICULTURE
8	appartement	'appartment'	BUILDING INDUSTRY, TOWN PLANNING
9	flat	'flat	FASHION, BUILDING INDUSTRY
10	schuur	'shed'	BUILDING INDUSTRY

Table 7.2: Domain tags for the similar words of *huis*, 'house'

or hyponymous) to the target word *huis* 'house'. Most of those words also belong to the same semantic domain BUILDING INDUSTRY, yielding a high sim_{topic} score of $\frac{8}{9} = .89$.[3]

One last important fact to be considered is that – due to the automatic procedure – a significant number of words in CORNETTO is erroneously tagged, or has an incomplete

[3] *huisje* 'small house' is not present in the CORNETTO database, so only 9 words are taken into account.

	word	translation	domain tags
0	film	'movie'	CINEMA, PHOTOGRAPHY, TV, ...
1	regisseur	'director'	MUSIC
2	speelfilm	'movie'	PAINTING
3	filmmaker	'filmmaker'	PERSON
4	hoofdrolspeler	'main actor'	THEATRE
5	acteur	'actor'	PERSON
6	hoofdrol	main part	THEATRE
7	script	script	THEATRE
8	hoofdrolspeelster	main actress	–
9	cinema	cinema	BUILDING INDUSTRY, THEATRE, ...
10	scenarioschrijver	scriptwriter	LITERATURE

Table 7.3: Domain tags for the words topically related to *film*, 'movie'

tagset. Take, for example, the topically related words to the target word *film* 'movie', presented in table 7.3. All words clearly seem topically related to the CINEMA domain. Yet, none of the similar words is tagged with the actual CINEMA tag, so that a topical similarity score sim_{topic} of .00 is attributed. As a result, many sets may in reality be more topically related than the evaluation score suggests.

7.3 Results

The models in this chapter are built in exactly the same way as the models presented in chapters 5 and 6, in which the wordnet-based similarity and cluster quality are evaluated. The baseline topical similarity score (using the same random models that were used for the creation of the CORNETTO baseline in chapter 5) amounts to $.0482 \pm 1.09 \times 10^{-3}$ (mean and standard deviation).

7.3.1 Document-based models

Weighting

In table 7.4 the topical similarity scores for the document-based model are given, according to different weighting functions. The results indicate that the model using a large document size performs better. Also, using logarithmic weighting improves the results. Entropy weighting does not improve the results. The different weighting scores

	model	sim_{topic} w_{par}	w_{art}
1	A_{freq}	.315	.349
2	$A_{logfreq}$	**.329**	**.359**
3	A_{ent}	.315	.349
4	A_{logent}	**.329**	**.359**

Table 7.4: Topical similarity results for different weighting parameters in the document-based model with two different document sizes

presented here show the same tendencies as the weighting scores for the document-based model presented in chapter 5.

Dimensionality reduction

Table 7.5 gives the results for the document-based model in combination with singular value decomposition. The results show that SVD as a dimensionality reduction is able to improve the results. All models tend to improve with SVD (reduced to 300 dimensions); in combination with log-entropy weighting, the similarity scores improve most. These results are again similar to the results for dimensionality reduction presented in chapter 5: SVD improves the results, and the improvement is the largest with log-entropy weighting.

7.3.2 Bag of words models

Window size

In table 7.6, the topical similarity scores for different window sizes are given. If we compare the results to those in table 5.8 on page 74, we notice some striking differences. Whereas the CORNETTO similarity is clearly best with smaller context windows – and heavily deteriorates with larger context windows – this is not the case for the topical similarity score. The models with a smaller context window (w_2, w_5) still get a good score, but the results for the models with large context window (w_{10}, w_{sent} and to a lesser extent w_{par}) are clearly within the same range. The results indicate that the window-based models are able to extract topical similarity with small windows as well as with large windows.

			sim_{topic}	
A	M	d	w_{par}	w_{art}
A_{freq}	M_{svd}	50	.262	.320
		100	.300	.347
		300	.340	.370
$A_{logfreq}$	M_{svd}	50	.263	.335
		100	.306	.359
		300	.343	.379
A_{logent}	M_{svd}	50	.302	.346
		100	.329	.372
		300	**.357**	**.394**
A_{logent}	–	–	.329	.359

Table 7.5: Evaluation results for different dimensionality reduction parameters in the document-based model with two different document sizes

	window size	sim_{topic}
1	w_1	.380
2	w_2	.414
3	w_5	**.423**
4	w_{10}	.422
5	w_{sent}	.418
6	w_{par}	.399

Table 7.6: Evaluation results for different window sizes in the bag of words model

Weighting

In table 7.7, the topical similarity scores for the four different weighting functions are presented, both for a context window of 2 and a paragraph window. One striking fact to notice is that a simple frequency model in combination with a paragraph window performs quite well. The other scores are in line with the semantic similarity scores presented in chapter 5: weighting with pointwise mutual information scores best for both window sizes, and the other weighting parameters $A_{logfreq}$ and A_{logpmi} perform worse than A_{pmi} with a small window (w_2) and even perform worse with a large paragraph window (w_{par}).

	model	sim_{topic} w_2	w_{par}
1	A_{freq}	.209	.349
2	$A_{logfreq}$	.350	.309
3	A_{pmi}	**.414**	**.399**
4	A_{logpmi}	.370	.325

Table 7.7: Evaluation results for different weighting parameters in the bag of words model with two different context windows

Dimensionality reduction

In table 7.8, the topical similarity score for different dimensionality reduction algorithms is given, with regard to two different context window sizes (w_2 and w_{par}). The tendencies with regard to dimensionality reduction for bag of words models again do not differ tremendously from the tendencies we have seen in chapter 5. The best scoring models are the ones using pointwise mutual information with a singular value decomposition. For small windows (w_2), the best scoring model does not improve upon its non-reduced counterpart. This is different, however, for large windows (w_{par}): pointwise mutual information combined with singular value decomposition to 100 and 300 dimensions brings about an improvement compared to its non-reduced counterpart. Note that this improvement is larger for the topical similarity presented here, compared to the wordnet-based similarity (table 5.12 on page 77).

Note that, for non-negative matrix factorization, the best scoring models are also the ones using pointwise mutual information. In chapter 5, the best scoring NMF models were the ones using simple frequency information.

7.3.3 Syntax-based models

Weighting

Table 7.9 gives the topical similarity scores for different weighting functions for syntax-based models. The model using only pointwise mutual information again performs best, followed by the model using a combination of logarithmic weighting and PMI, and the model using only logarithmic weighting. Simple frequency information again performs worst.

Compared to the wordnet-based similarity results, the topical similarity scores are a little bit different. Whereas with the wordnet-based similarity, both PMI and

			sim_{topic}	
A	M	d	w_2	w_{par}
A_{freq}	M_{svd}	50	.190	.302
		100	.200	.328
		300	.207	.349
	M_{nmf}	50	.260	.341
		100	.284	.367
		300	.285	.371
$A_{logfreq}$	M_{svd}	50	.301	.295
		100	.320	.305
		300	.339	.311
	M_{nmf}	50	.283	.293
		100	.300	.305
		300	.296	.308
A_{pmi}	M_{svd}	50	.380	.384
		100	.404	.400
		300	**.416**	**.409**
	M_{nmf}	50	.306	.339
		100	.316	.373
		300	.329	.388
A_{pmi}	–	–	.442	.399

Table 7.8: Evaluation results for different dimensionality reduction parameters in the bag of words based model with two different context windows

logarithmic weighting perform quite well, PMI is clearly the winner with regard to topical similarity.

Dimensionality reduction

Table 7.10 gives the results for different dimensionality reduction algorithms in the syntax-based models. Singular value decomposition combined with PMI again performs best, but dimensionality reduction algorithms do not bring about any improvement, compared to the models without dimensionality reduction.

	model	sim_{topic}
1	A_{freq}	.259
2	$A_{logfreq}$	.404
3	A_{pmi}	**.441**
4	A_{logpmi}	.432

Table 7.9: Evaluation results for different weighting parameters in the syntax-based model

A	M	d	sim_{topic}
A_{freq}	M_{svd}	50	.185
		100	.214
		300	.253
	M_{nmf}	50	.266
		100	.310
		300	.342
$A_{logfreq}$	M_{svd}	50	.345
		100	.370
		300	.391
	M_{nmf}	50	.270
		100	.340
		300	.379
A_{pmi}	M_{svd}	50	.377
		100	.405
		300	**.434**
	M_{nmf}	50	.256
		100	.303
		300	.337
A_{pmi}	–	–	.441

Table 7.10: Topical similarity for different dimensionality reduction parameters in the syntax-based model

7.4 Discussion & comparison

The results presented in this chapter provide mixed results. On the one hand, models that score well in the two former tasks (semantic similarity in chapter 5 and cluster quality in chapter 6) also perform well on the task of extracting topically similar words. This is in particular the case for the syntax-based approach, and to a lesser extent for the window-based approach with small window size.

On the other hand, we notice that models that do not perform well on the two former evaluation tasks (the document-based model and the window-based model with large window) still perform reasonably well with regard to the extraction of topical similarity. We may therefore conclude that those models are not extracting tight, synonym-like semantic similarity, but are still extracting semantically related, topically similar words.

Let us illustrate this fact with a number of examples. In the examples below, we compare the 10 most similar words according to:

a. the syntax-based model, using PMI weighting: this model performs best both with regard to wordnet-based similarity/cluster quality and topical similarity;

b. the window-based model, using PMI weighting and paragraph window: this model performs worse with regard to wordnet-based similarity/cluster quality, but performs reasonably well in the topical similarity evaluation.

(1) a. **krant** 'newspaper': *dagblad* 'newspaper', *zakenkrant* 'business paper', *kwaliteitskrant* 'quality newspaper', *blad* 'paper', *tabloid* 'tabloid', *zondagskrant* 'sunday newspaper', *weekblad* 'magazine', *ochtendblad* 'morning paper', *boulevardblad* 'tabloid, *sportkrant* 'sports paper'

b. **krant** 'newspaper': *dagblad* 'newspaper', *voorpagina* 'front page', *hoofdredacteur* 'chief editor', *redactie* 'editors', *weekblad* 'magazine', *kwaliteitskrant* 'quality newspaper', *redacteur* 'editor', *berichtgeving* 'coverage', *hoofdredactie* 'chief editors', *journalist* 'journalist'

(2) a. **film** 'movie': *speelfilm* 'movie', *thriller* 'thriller', *komedie* 'comedy', *documentaire* 'documentary', *musical* 'musical', *drama* 'tragedy', *boek* 'book', *roman* 'novel', *muziek* 'music', *video* 'video'

b. **film** 'movie': *regisseur* 'director', *speelfilm* 'movie', *filmmaker* 'filmmaker', *hoofdrolspeler* 'main actor', *acteur* 'actor', *hoofdrol* 'main part', *script* 'script', *hoofdrolspeelster* 'main actress', *cinema* 'cinema', *scenarioschrijver* 'script writer'

(3) a. **politie** 'police': *brandweer* 'fire brigade', *justitie* 'justice', *douane* 'customs', *leger* 'army', *politiekorps* 'police force', *Politie* 'Police', *agent*

'officer', *autoriteit* 'authority', *recherche* 'criminal investigation department', *marechaussee* 'military police'

b. **politie** 'police': *politiewoordvoerder* 'police spokesman', *agent* 'officer', *politiebureau* 'police station', *politiemens* 'police person', *aanhouding* 'arrest', *politieagente* 'police woman', *vuurwapen* 'firearm', *arrestant* 'arrest person', *arrestatieteam* 'special squad', *recherche* 'criminal investigation department

Clearly, both approaches capture a different kind of semantic similarity. The first approach captures a tight, synonym-like similarity, whereas the second approach captures a broader, topical relatedness. In example (1), the syntax-based model (a) yields synonyms (*dagblad* 'newspaper') and hyponyms (*tabloid* 'tabloid') as semantically similar words for *krant* 'newspaper'. The window-based model (b), on the other hand, also yields more loosely related words, such as meronyms (*voorpagina* 'front page') and associations (*redacteur* 'editor', *journalist* 'journalist'). The same tendency is present in the two other examples. In example (2), the syntax-based model yields synonyms and hyponyms, and a number of co-hyponyms (*boek* 'book', *muziek* 'music') as different kinds of media/art expressions. The window-based model again also yields more loosely, topically related words such as *acteur* 'actor' and *script* 'script'. And in example (3), the similar words in (a) consists mostly of co-hyponyms designating authoritative organizations similar to *politie* 'police', whereas in (b), more topically related words such as *aanhouding* 'arrest' and *vuurwapen* 'firearm' show up again.

Changing the context also changes the semantic characteristics captured by the models. Syntax-based models and window-based models with small window size are particularly apt for the extraction of tight, synonym-like similarity, as has been shown in chapter 5, in which the wordnet-based similarity was evaluated by comparing them to the semantic hierarchical database CORNETTO, and in chapter 6, in which the models' cluster quality was compared to a gold standard. In this chapter, we have shown that those models also perform well with regard to the extraction of topical similarity: words that are tightly similar (such as synonyms and hyponyms) are usually also topically related.

Document-based models and window-based models with large window size, on the other hand, are not suitable for the extraction of tight, synonym-like similarity. As we have shown in chapters 5 and 6, these models perform far worse with regard to wordnet-based similarity evaluation and the evaluation of cluster quality. Still, the models perform reasonably well in the extraction of topical similarity, as we have shown in this chapter. The examples shown above are also exemplary for the different semantics extracted by the different models. Clearly, a tight context (syntax-based or small window-based) leads to tight, synonym-like similarity, whereas a broad context

(document-based or large window-based captures topically similar words that are often more loosely related to the target word.

Part III

Applications

Chapter 8

Lexico-Semantic Multiword Expression Extraction[1]

8.1 Introduction

In the previous chapters, we have shown that it is possible to automatically extract a database of semantically similar words. More specifically, we have shown that we are able to extract a high quality semantic clustering of nouns, using a syntax-based word space model. In this chapter, we will explain how the extracted semantic similarity database can in turn be used to automatically extract multi-word expressions (MWEs) from text.

MWEs are expressions whose linguistic behaviour is not predictable from the linguistic behaviour of their component words. Baldwin (2006) characterizes the idiosyncratic behavior of MWEs as 'a lack of compositionality manifest at different levels of analysis, namely, lexical, morphological, syntactic, semantic, pragmatic and statistical'. One property that seems to affect MWEs the most is semantic non-compositionality. MWEs are typically non-compositional. As a consequence, it is not possible to replace the content words of a MWE by semantically similar words. Take for example the expressions in (1) and (2):

(1) a. break the vase
    b. break the cup
    c. break the dish

[1] This chapter presents joint work with Begoña Villada Moirón. The research presented in this chapter has been published as Van de Cruys and Villada Moirón (2007a) and Van de Cruys and Villada Moirón (2007b).

(2) a. break the ice
b. *break the snow
c. *break the hail

Expression (1) is fully compositional. Therefore, *vase* can easily be replaced with semantically similar nouns such as *cup* and *dish*. Expression (2), on the contrary, is non-compositional; it is impossible to replace *ice* with semantically related words, such as *snow* and *hail*. Note that we assume a dual classification of expressions into compositional and non-compositional instances; we ignore the possibility that expressions fall in a continuum between compositionality and non-compositionality with many fuzzy cases in between. By 'fuzzy cases' we refer to expressions that are neither fully compositional nor fully non-compositional; such expressions may involve metaphoricity or figurative language.

Due to their non-compositionality, current proposals argue that MWEs need to be described in the lexicon (Sag et al., 2002). In most languages, electronic lexical resources (such as dictionaries, thesauri, ontologies) suffer from a limited coverage of MWEs. To facilitate the update and expansion of language resources, the NLP community would clearly benefit from automated methods that extract MWEs from large text collections. This is the main motivation to pursue an automated and fully unsupervised MWE extraction method.

8.2 Previous work

Recent proposals that attempt to capture semantic compositionality (or lack thereof) employ various strategies. Approaches evaluated so far make use of dictionaries with semantic annotation (Piao et al., 2006), WordNet (Pearce, 2001), automatically generated thesauri (Lin, 1999; Fazly and Stevenson, 2006; McCarthy, Keller, and Carroll, 2003), vector-based methods that measure semantic distance (Baldwin et al., 2003; Katz and Giesbrecht, 2006), translations extracted from parallel corpora (Villada Moirón and Tiedemann, 2006) or hybrid methods that use machine learning techniques informed by features coded using some of the above methods (Venkatapathy and Joshi, 2005).

Pearce (2001) describes a method to extract collocations from corpora by measuring semantic compositionality. The underlying assumption is that a fully compositional expression allows synonym replacement of its component words, whereas a collocation does not. Pearce measures to what degree a collocation candidate allows synonym replacement. The measurement is used to rank candidates relative to their compositionality.

Building on Lin (1998a), McCarthy, Keller, and Carroll (2003) measure the semantic similarity between expressions (phrasal verbs) as a whole and their component words. They exploit contextual features and frequency information in order to assess meaning overlap. They established that human compositionality judgements correlate well with those measures that take into account the semantics of the particle. Contrary to these measures, multiword extraction statistics (log-likelihood, mutual information) correlate poorly with human judgements.

A different approach proposed by Villada Moirón and Tiedemann (2006) measures translational entropy as a sign of meaning predictability, and therefore non-compositionality. The entropy observed among word alignments of a potential MWE varies: highly predictable alignments show less entropy and probably correspond to compositional expressions. Data sparseness and polysemy pose problems because the translational entropy cannot be accurately calculated.

Fazly and Stevenson (2006) use lexical and syntactic fixedness as partial indicators of non-compositionality. Their method uses Lin's (1998) automatically generated thesaurus to compute a metric of lexical fixedness. Lexical fixedness measures the deviation between the pointwise mutual information of a verb-object phrase and the average pointwise mutual information of the expressions resulting from substituting the noun by its synonyms in the original phrase. This measure is similar to Lin's (1999) proposal for finding non-compositional phrases. Separately, a syntactic flexibility score measures the probability of seeing a candidate in a set of pre-selected syntactic patterns. The assumption is that non-compositional expressions score high in idiomaticity, that is, a score resulting from the combination of lexical fixedness and syntactic flexibility. The authors report an 80% accuracy in distinguishing literal from idiomatic expressions in a test set of 200 expressions. The performance of both metrics is stable across all frequency ranges.

In this study, we are interested in establishing whether a fully unsupervised method can capture the (partial or) non-compositionality of MWEs. The method should not depend on the existence of large (open domain) parallel corpora or sense tagged corpora. Also, the method should not require numerous adjustments when applied to new subclasses of MWEs, for instance, when coding empirical attributes of the candidates. Similar to Lin (1999), McCarthy, Keller, and Carroll (2003) and Fazly and Stevenson (2006), our method makes use of automatically generated thesauri; the technique used to compile the thesauri differs from previous work. Aiming at finding a method of general applicability, the measures to capture non-compositionality differ from those employed in earlier work.

8.3 Methodology

In the description and evaluation of our algorithm, we focus on the extraction of verbal MWEs that contain prepositional complements, although the method could easily be generalized to other kinds of MWEs.

In our semantics-based approach, we want to formalize the intuition of non-compositionality, so that MWE extraction can be done in a fully automated way. A number of statistical measures are developed that try to capture the MWE's non-compositional bond between a verb-preposition combination and its noun by comparing the particular noun of an MWE candidate to other semantically related nouns.

8.3.1 Data extraction

The MWE candidates (verb + prepositional phrase) are automatically extracted from the *Twente Nieuws Corpus* (Ordelman, 2002), a large corpus of Dutch newspaper texts (500 million words), which has been automatically parsed by the Dutch dependency parser Alpino (van Noord, 2006). Next, a matrix is created of the 5,000 most frequent verb-preposition combinations by the 10,000 most frequent nouns, containing the frequency of each MWE candidate.[2] To this matrix, a number of statistical measures are applied to determine the non-compositionality of the candidate MWEs. These statistical measures are explained in §8.3.3.

8.3.2 Clustering

In order to compare a noun to its semantically related nouns, a noun clustering is created. These clusters are automatically extracted using standard distributional similarity techniques.[3] First, dependency triples are extracted from the *Twente Nieuws Corpus*. Next, feature vectors are created for each noun, containing the frequency of the dependency relations in which the noun occurs. A frequency matrix of 10K nouns by 100K dependency relations is constructed. The cell frequencies are weighted with pointwise mutual information. clusters using a K-means clustering algorithm (MacQueen, 1967) using cosine similarity. During development, several other clustering algorithms and parameters have been tested, but the settings described above gave us the best EuroWordNet similarity score (using Wu and Palmer (1994)).

Note that our clustering algorithm is a hard clustering algorithm, which means that a certain noun can only be assigned to one cluster. This may pose a problem for

[2] The lowest frequency verb-preposition combination (with regard to the 10,000 nouns) appears 3 times

[3] A detailed description of distributional similarity techniques is given in chapter 2.

polysemous nouns. On the other hand, this makes the computation of our metrics straightforward, since we do not have to decide among various senses of a word.

8.3.3 Measures

The measures used to find MWEs are inspired by Resnik's method to find selectional preferences (Resnik, 1993; Resnik, 1996). Resnik uses a number of measures based on the Kullback-Leibler divergence, to measure the difference between the prior probability of a noun class $p(c)$ and the probability of the class given a verb $p(c|v)$. We adopt the method for particular nouns, and add a measure for determining the 'unique preference' of a noun given other nouns in the cluster, which, we claim, yields a measure of non-compositionality. In total, four measures are used, the latter two being the symmetric counterpart of the former two.

Verb preference

The first two measures, $A_{v\rightarrow n}$ (equation 8.2) and $R_{v\rightarrow n}$ (equation 8.3), formalize the unique preference of the verb[4] for the noun. Equation 8.1 gives the Kullback-Leibler divergence between the overall probability distribution of the nouns and the probability distribution of the nouns given a verb; it is used as a normalization constant in equation 8.2. Equation 8.2 models the actual preference of the verb for the noun.

$$S_v = \sum_n p(n \mid v)\ log\frac{p(n \mid v)}{p(n)} \tag{8.1}$$

$$A_{v\rightarrow n} = \frac{p(n \mid v)\ log\frac{p(n|v)}{p(n)}}{S_v} \tag{8.2}$$

When $p(n|v)$ is 0, $A_{v\rightarrow n}$ is undefined. In this case, we assign a score of 0.

Equation 8.3 gives the ratio of the verb preference for a particular noun, compared to the other nouns that are present in the cluster.

$$R_{v\rightarrow n} = \frac{A_{v\rightarrow n}}{\sum_{n'\varepsilon C} A_{v\rightarrow n'}} \tag{8.3}$$

where C is the cluster in which the noun n appears. When $R_{v\rightarrow n}$ is more or less equally divided among the different nouns in the cluster, there is no preference of the verb for a particular noun in the cluster, whereas scores close to 1 indicate a 'unique' preference of the verb for a particular noun in the cluster. Candidates whose $R_{v\rightarrow n}$ value approaches 1

[4] We will use 'verb' to designate a prepositional verb, i.e. a combination of a verb and a preposition.

are likely to be non-compositional expressions, since the noun of the expression cannot be substituted with a semantically similar noun.

Noun preference

In the latter two measures, $A_{n\rightarrow v}$ and $R_{n\rightarrow v}$, the direction of preference is changed: they model the unique preference of the noun for the verb. Equation 8.4 models the Kullback-Leibler divergence between the overall probability distribution of verbs, and the distribution of the verbs given a certain noun. It is used again as a normalization constant in equation 8.5, which models the preference of the noun for the verb.

$$S_n = \sum_v p(v \mid n)\ log \frac{p(v \mid n)}{p(v)} \tag{8.4}$$

$$A_{n\rightarrow v} = \frac{p(v \mid n)\ log \frac{p(v|n)}{p(v)}}{S_n} \tag{8.5}$$

When $p(v|n)$ is 0, $A_{n\rightarrow v}$ is undefined. In this case, we again assign a score of 0.

Equation 8.6 gives the ratio of noun preference for a particular verb, compared to the other nouns that are present in the cluster.

$$R_{n\rightarrow v} = \frac{A_{n\rightarrow v}}{\sum_{n' \varepsilon C} A_{n'\rightarrow v}} \tag{8.6}$$

Both measures have the same characteristics as the ones that model verb preference. If a noun shows a much higher preference for a verb than the other nouns in the cluster, we expect that the candidate expression tends towards non-compositionality.

Note that the measures for verb preference and the measures for noun preference are different in nature. It is possible that a certain verb only selects a restricted set of nouns, while the nouns themselves can co-occur with many different verbs. This brings about different probability distributions. In our evaluation, we want to investigate the impact of both preferences.

Lexical fixedness measure

For reasons of comparison, we also evaluated the lexical fixedness measure – based on pointwise mutual information – proposed by Fazly and Stevenson (2006).[5] The lexical fixedness is computed following equation 8.7

[5]Fazly and Stevenson (2006) combine the lexical fixedness measure with a measure of syntactic flexibility. Here, we only compare our method to the former measure, concentrating on non-compositionality rather than syntactic rigidity.

$$Fixedness_{lex}(v,n) = \frac{PMI(v,n) - \overline{PMI}}{s} \tag{8.7}$$

where $\overline{PMI}$ stands for the mean given the cluster, and s for the standard deviation. Note that Fazly and Stevenson (2006) use the m most similar nouns given a certain noun, while we use all nouns in a cluster. This means that our value for m varies.

8.3.4 Example

In this section, an elaborated example is presented, to show how our method works. Take for example the two MWE candidates in (3):

(3) a. in de smaak vallen
in the taste fall
to be appreciated
b. in de put vallen
in the well fall
to fall down the well

In the first expression, *smaak* cannot be replaced with other semantically similar nouns, such as *geur* 'smell' and *zicht* 'sight', whereas in the second expression, *put* can easily be replaced with other semantically similar words, such as *kuil* 'hole' and *krater* 'crater'.

The first step in the formalization of this intuition, is the extraction of the clusters in which the words *smaak* and *put* appear from our clustering database. This gives us the clusters in (4).

(4) a. **smaak**: *aroma* 'aroma', *gehoor* 'hearing', *geur* 'smell', *gezichtsvermogen* 'sight', *reuk* 'smell', *spraak* 'speech', *zicht* 'sight'
b. **put**: *afgrond* 'abyss', *bouwput* 'building excavation', *gaatje* 'hole', *gat* 'hole', *hiaat* 'gap', *hol* 'cave', *kloof* 'gap', *krater* 'crater', *kuil* 'hole', *lacune* 'lacuna', *leemte* 'gap', *valkuil* 'pitfall'

Next, the various measures described in §8.3.3 are applied. Resulting scores are given in tables 8.1 and 8.2.

Table 8.1 gives the scores for the MWE *in de smaak vallen*, together with some other nouns that are present in the same cluster. $A_{v\rightarrow n}$ shows that there is a clear preference (.12) of the verb *val in* for the noun *smaak*. $R_{v\rightarrow n}$ shows that there is a unique preference of the verb for the particular noun *smaak*. For the other nouns (*geur, zicht,* ...), the

MWE candidate	$A_{v\rightarrow n}$	$R_{v\rightarrow n}$	$A_{n\rightarrow v}$	$R_{n\rightarrow v}$
val#in smaak	.12	1.00	.04	1.00
val#in geur	.00	.00	.00	.00
val#in zicht	.00	.00	.00	.00

Table 8.1: Scores for MWE candidate *in de smaak vallen* and other nouns in the same cluster

verb has no preference whatsoever. Therefore, the ratio of verb preference for *smaak* compared to the other nouns in the cluster is 1.00.

$A_{n\rightarrow v}$ and $R_{n\rightarrow v}$ show similar behaviour. There is a preference (.04) of the noun *smaak* for the verb *val in*, and this preference is unique (1.00).

MWE candidate	$A_{v\rightarrow n}$	$R_{v\rightarrow n}$	$A_{n\rightarrow v}$	$R_{n\rightarrow v}$
val#in put	.00	.05	.00	.05
val#in kuil	.01	.11	.02	.37
val#in kloof	.00	.02	.00	.03
val#in gat	.04	.71	.01	.24

Table 8.2: Scores for MWE candidate *in de put vallen* and other nouns in the same cluster

Table 8.2 gives the scores for the instance *in de put vallen* – which is not a MWE – together with other nouns from the same cluster. The results are quite different from the ones in table 8.1. $A_{v\rightarrow n}$ – the preference of the verb for the noun – is quite low in most cases, the highest score being a score of .04 for *gat*. Furthermore, $R_{v\rightarrow n}$ does not show a unique preference of *val in* for *put* (a low ratio score of .05). Instead, the preference mass is divided among the various nouns in the cluster, the highest preference of *val in* being assigned to the noun *gat* (.71).[6]

The other two scores show again a similar tendency; $A_{n\rightarrow v}$ – the preference of the noun for the verb – is low in all cases, and when all nouns in the cluster are considered ($R_{n\rightarrow v}$), there is no 'unique' preference of one noun for the verb *val in*. Instead, the preference mass is divided among all nouns in the cluster.

[6]Note that this expression is ambiguous: it can be used in a literal sense (*in een gat vallen*, 'to fall down a hole') and in a metaphorical sense (*in een zwart gat vallen*, 'to get depressed after a joyful or busy period').

After assessing the values of the four different measures, our method would propose *in de smaak vallen* as a non-compositional expression and therefore, MWE; on the other hand, the method would consider *in de put vallen* as compositional, thus a non-MWE.

8.4 Results and evaluation

In this section, our automatic method is extensively evaluated. In the first part, we present the results of our quantitative evaluation – including both an automatic evaluation (using Dutch lexical resources) and a manual evaluation (carried out by human judges). The second part is a qualitative evaluation, indicating the advantages and the drawbacks of our method.

8.4.1 Quantitative evaluation

Automatic evaluation

The MWEs that are extracted with the fully unsupervised method described above are automatically evaluated by comparing them to handcrafted lexical databases. Since we have extracted Dutch MWEs, we are using the two Dutch resources available: the Referentie Bestand Nederlands (RBN, (Martin and Maks, 2005)) and the Van Dale Lexicographical Information System (VLIS) database. Precision and recall are calculated with regard to the MWEs that are present in our evaluation resources. Among the MWEs in our reference data, we consider only those expressions that are present in our frequency matrix: if the verb is not among the 5,000 most frequent verbs, or the noun is not among the 10,000 most frequent nouns, the frequency information is not present in our input data. Consequently, our algorithm would never be able to find those MWEs.

The first six rows of table 8.3 show precision, recall and f-measure for various parameter thresholds with regard to the measures $A_{v\rightarrow n}$, $R_{v\rightarrow n}$, $A_{n\rightarrow v}$ and $R_{n\rightarrow v}$, together with the number of candidates found (n). The last line shows the highest values we were able to reach by using the lexical fixedness score.

Using only two parameters – $A_{v\rightarrow n}$ and $R_{v\rightarrow n}$ – gives the highest f-measure ($\pm$ 14%), with a precision and recall of about 17% and about 12% respectively. Adding parameter $R_{n\rightarrow v}$ increases precision but degrades recall, and this tendency continues when adding both parameters $A_{n\rightarrow v}$ and $R_{n\rightarrow v}$. In all cases, a higher threshold increases precision but degrades recall. When using a high threshold for all parameters, the algorithm is able to reach a precision of $\pm$ 38%, but recall is low ($\pm$ 4%).

parameters					precision	recall	f-measure
$A_{v\to n}$	$R_{v\to n}$	$A_{n\to v}$	$R_{n\to v}$	n	(%)	(%)	(%)
.10	.80	–	–	3175	16.09	13.11	14.45
.10	.90	–	–	2655	17.59	11.98	14.25
.10	.80	–	.80	2225	19.19	10.95	13.95
.10	.90	–	.90	1870	20.70	9.93	13.42
.10	.80	.01	.80	1859	20.33	9.69	13.13
.20	.99	.05	.99	404	38.12	3.95	7.16
$Fixedness_{lex}(v,n)$		3.00		3899	15.14	9.92	11.99

Table 8.3: Evaluation results compared to RBN & VLIS

The lexical fixedness score is able to reach an f-measure of $\pm$ 12% (using a threshold of 3.00). These scores show the best performance that we have reached using lexical fixedness.

Human evaluation

The evaluation procedure described above was applied fully automatically by comparing the output of our method to two existing Dutch lexical databases. We are aware of the fact that the automated annotation process may introduce some errors. There may be extracted expressions wrongly labeled as true MWEs, as well as extracted expressions erroneously labeled as false MWEs. Furthermore, it is known that the lexical databases used are static resources that are likely to miss actual MWEs found in large corpora. This is either because the lexical resources are incomplete, or because the MWEs were not included due to a different understanding of the concept of MWE. With this motivation, we set up a human evaluation experiment. From the dataset that produced the best f-measure ($A_{v\to n}$ = .10 and $R_{v\to n}$ = .80), 200 expressions were randomly selected. To assess the performance of our method across different frequency ranges, we selected 100 highly frequent MWE candidates (frequency $\geq$ 100) and 100 less frequent ones (frequency $<$ 100).

Three human judges were asked to label the expressions as MWE or as non-MWE. The judges were asked to always provide an answer. To investigate if the rankings from the 3 judges agreed, we employed the Kappa statistic (Cohen, 1960). We obtained an average pairwise interannotator agreement of $\kappa = .60$, showing a reasonable correlation between the judges.

The scores assigned by the judges differed severely with regard to frequency range. In the high frequency range, our method was given an average precision of 33.00%. In the low frequency range, precision dropped down to 6.67%. Below, the results of our human evaluation are evaluated more extensively.

8.4.2 Qualitative evaluation

In this section, we elaborate upon advantages and disadvantages of our semantics-based MWE extraction algorithm by examining the output of the procedure, and looking at the characteristics of the correct MWEs found and the errors made by the algorithm.

Advantages of the method

First of all, our algorithm is able to filter out grammatical collocations that cause problems in traditional MWE extraction paradigms. Two examples are given in (5) and (6).

(5) benoemen tot minister, secretaris-generaal
appoint to minister, secretary-general
appoint s.o. {*minister, secretary-general*}

(6) voldoen aan eisen, voorwaarden
meet to demands, conditions
meet the {*demands, conditions*}

In traditional MWE extraction algorithms, based on collocations, highly frequent expressions like the ones in (5) and (6) often get classified as a MWE, even though they are fully compositional. Such algorithms correctly identify a strong lexical affinity between two component words (*voldoen*, *aan*), which make up a grammatical collocation; however, they fail to capture the fact that the noun may be filled in by a semantic class of nouns. Our algorithm filters out those expressions, because semantic similarity is taken into account.

Our quantitative evaluation shows that the algorithm reaches the best results (i.e. the highest f-measures) when only two parameters ($A_{v\rightarrow n}$ and $R_{v\rightarrow n}$) are taken into account. But upon closer inspection of the output, we have noticed that $A_{n\rightarrow v}$ and $R_{n\rightarrow v}$ are often able to filter out non-MWEs like the expressions b in (7) and (8).

(7) a. op toneel verschijnen
on stage appear
to appear

b. op toneel zingen
on stage sing
to sing on the stage

(8) a. in geheugen liggen
in memory lie
be in memory
b. in ziekenhuis liggen
in hospital lie
lie in the hospital

When only taking into account the first two measures (a unique preference of the verb for the noun), the expressions in b do not get filtered out. It is only when the two other measures (a unique preference of the noun for the verb) are taken into account that they are filtered out – either because the preference of the noun for the verb is very low, or the noun preference for the verb is more evenly distributed among the cluster. The b expressions, which are non-MWEs, result from the combination of a verb with a highly frequent PP. These PPs are typically locative, directional or predicative PPs, that may combine with numerous verbs.

Also, expressions like the ones in (9), where the fixedness of the expression lies not so much in the verb-noun combination, but more in the PP part (*naar school*, *naar huis*) are filtered out by the latter two measures. These preposition-noun combinations seem to be institutionalized PPs, so-called determinerless PPs (Baldwin et al., 2006).

(9) a. naar school willen
to school want
want to go to school
b. naar huis willen
to home want
want to go home

Errors of the method

In this section, we give an exhaustive list of the errors made by our algorithm, and quantitatively evaluate the importance of each error category.

1. First of all, our algorithm highly depends on the quality of the noun clustering. If a noun appears in a cluster with unrelated words, the measures will overrate the semantic uniqueness of the expressions in which the noun appears.

2. Syntax might play an important role. Sometimes, there are syntactic restrictions between the preposition and the noun. A noun like *pagina* ‘page’ can only

appear with the preposition *op* 'on', as in *lees op pagina* 'read on page'. Other, semantically related nouns, such as *hoofdstuk* 'chapter', prefer *in* 'in'. Due to these restrictions, the measures will again overrate the semantic uniqueness of the expression.

3. We found many expressions in which the fixedness of the expression lies not so much in the combination of the verb and the prepositional phrase, but rather in the prepositional phrase itself (*naar school*, *naar huis*). Note, however, that our two latter measures were able to filter out many of those expressions (as explained above). But in our error evaluation, we used the result that yields the highest f-measure (and does not take the latter measures into account).

4. Our hard clustering method does not take polysemous nouns into account. A noun can only occur in one cluster, ignoring other possible meanings. *Schaal*, for example, means 'dish' as well as 'scale'. In our clustering, it only appears in a cluster of dish-related nouns. Therefore, expressions like *maak gebruik op [grote] schaal* 'make use of [sth.] on a [large] scale', receive again overrated measures of semantic uniqueness, because the 'scale' sense of the noun is compared to nouns related to the 'dish' sense.

5. Related to the previous error category is the fact that certain nouns – although occurring in a perfectly sound cluster – possess a semantic feature or characteristic that distinguishes them from the other nouns in the cluster, and causes the verb to uniquely prefer that particular noun. An example of this kind of error is the expression *eet in restaurant* 'eat in a restaurant', which is perfectly compositional. But due to the fact that the noun *restaurant* ends up in a cluster with nouns such as *bar* 'bar', *café* 'bar', *kroeg* 'pub', *winkel* 'shop', *hotel* 'hotel' – which are places where one is less likely to eat – the fixedness of the expression is overestimated.

6. The effectiveness of our method is highly dependent on the corpus distribution. Sometimes, expressions that would be effective counterweights for the erroneous classification of compositional expressions as MWE just are not found in the corpus. This might be either due to sparseness of the data, or due to the specific nature of the corpus itself. Examples are *sluit wegens verbouwing* 'close due to alteration', with cluster members such as *restauratie* 'restoration' and *renovatie* 'renovation', and *uit van emotie* 'express emotion', with cluster members such as *agressie* 'agression', *irritatie* 'irritation', *ongeduld* 'impatience'. Expressions such as *sluit wegens renovatie* or *uit van irritatie* are perfectly possible, but are not (sufficiently) attested in the corpus. Therefore, the compositional forms which are attested in the corpus are overestimated as MWE.

7. Finally, misclassifications may be caused by parsing errors or other technical issues.

In order to get a better view of the errors of the method, we manually classified the expressions that were evaluated as non-MWE by our judges. Each expression was assigned to one of the error categories described above. Overall results, and results for high and low frequency expressions are given.

		overall (%)	high freq. (%)	low freq. (%)
1	erroneous clustering	3.6	3.8	3.4
2	specific preposition	6.4	15.4	1.1
3	PP fixedness	26.4	21.2	29.5
4	polysemous word	15.7	13.5	17.0
5	specific semantic feature	22.9	30.8	18.2
6	corpus distribution	21.4	13.5	26.1
7	parsing/other	3.6	1.9	4.5

Table 8.4: Quantitative error evaluation

Misclassifications due to erroneous clustering or parsing errors only constitute a small part of the errors. Also, misclassifications due to syntactic restrictions (specific prepositions) are responsible for only a small part of the errors. More important are misclassifications due to fixedness in the PP, or due to polysemy or specific semantic features of the nouns. The former might be remedied by a more effective use of our measures $A_{n\rightarrow v}$ and $R_{n\rightarrow v}$, the latter by taking on a soft clustering approach. Finally, there are quite some errors due to the specific distribution of MWEs in the corpus. These errors are more common in the low frequency range. Clearly, our method is highly dependent on the corpus that is used, and it should be sufficiently large in order to adequately classify less frequent MWEs.

MWE fuzziness

A last observation to mention is that the status of certain expressions extracted with our method is unclear. Expressions such as *vraag met klem* 'ask with emphasis' or *ga over tot orde [van de dag]* 'pass to the order [of the day]' seem to be on the border of compositionality vs. non-compositionality, and therefore cannot be adequately qualified as MWE or non-MWE. This observation is confirmed by the conflicting views the three judges showed when assessing these kind of expressions.

8.5 Conclusions and further work

Our algorithm based on non-compositionality explores a new approach aimed at large-scale MWE extraction. Using only two parameters, $A_{v\rightarrow n}$ and $R_{v\rightarrow n}$, yields the highest f-measure. Using the two other parameters, $A_{n\rightarrow v}$ and $R_{n\rightarrow v}$, increases precision but degrades recall. Due to the formalization of the intuition of non-compositionality (using an automatic noun clustering), our algorithm is able to rule out various expressions that are coined MWEs by traditional algorithms.

Note that our algorithm has taken on a purely semantics-based approach. 'Syntactic fixedness' of the expressions is not taken into account. Combining our semantics-based approach with other MWE extraction methods that take into account different features might improve the results significantly.

We believe that our method provides a genuine and successful approach to get a grasp of the non-compositionality of MWEs in a fully automated way. We also believe that it is one of the first methods able to extract MWEs based on non-compositionality on a large scale, and that traditional MWE extraction algorithms will benefit from taking this non-compositionality into account.

Chapter 9

Noun Sense Discrimination[1]

9.1 Introduction

The computation of semantic similarity relies on the distributional hypothesis (Harris, 1985), which states that words that occur in similar contexts tend to be similar. In chapter 1, we have seen that there are three different definitions of the notion context: a document-based context, a window-based context, and a syntax-based context. The first approach takes the documents in which the word appears as features. The document context might be the sentence, paragraph or actual document that a word appears in. One of the dominant methods using this method is LATENT SEMANTIC ANALYSIS (LSA). The second approach makes use of the bag of words co-occurrence data; in this approach, a certain window around a word is used for gathering co-occurrence information. The window may either be a fixed number of words, or the paragraph or document that a word appears in. Thus, words are considered similar if they appear in similar windows. The document-based approach is very much related to the bag of words approach. The third approach uses a more fine grained distributional model, focusing on the syntactic relations in which words appear. Typically, a large text corpus is parsed, and dependency triples are extracted.[2] Words are considered similar if they appear in similar syntactic relations. Note that the former approach does not need any kind of linguistic annotation, whereas for the latter, some form of syntactic annotation is needed.

[1] The research presented in this chapter has been published as Van de Cruys (2007) and Van de Cruys (2008c).

[2] e.g. dependency relations that qualify *apple* might be 'object of *eat*' and 'adjective *red*'. This gives us dependency triples like $< apple, obj, eat >$.

In this chapter, we will combine the window-based approach with the syntax-based approach. The results yielded by these two approaches are typically quite different in nature: the first approach typically puts its finger on a broad, thematic kind of similarity,[3] while the latter typically grasps a tighter, synonym-like similarity. Example (1) shows the difference between the two kinds of similarity; for each kind, the top ten most similar nouns to the Dutch noun *muziek* 'music' are given. In (a), the window-based approach is used, while (b) uses the syntax-based approach. (a) shows indeed more thematic similarity, whereas (b) shows tighter similarity.

(1) a. **muziek** 'music': *gitaar* 'guitar', *jazz* 'jazz', *cd* 'cd', *rock* 'rock', *bas* 'bass', *song* 'song', *muzikant* 'musician', *musicus* 'musician', *drum* 'drum', *slagwerker* 'drummer'

b. **muziek** 'music': *dans* 'dance', *kunst* 'art', *klank* 'sound', *liedje* 'song', *geluid* 'sound', *poëzie* 'poetry', *literatuur* 'literature', *popmuziek* 'pop music', *lied* 'song', *melodie* 'melody'

Especially the syntax-based method has been adopted by many researchers in order to find semantically similar words. There is, however, one important problem with this kind of approach: the method is not able to cope with ambiguous words. Take the examples:

(2) een oneven nummer
a odd number
an odd number

(3) een steengoed nummer
a great number
'a great song'

The word *nummer* does not have the same meaning in these examples. In example (2), *nummer* is used in the sense of 'designator of quantity'. In example (3), it is used in the sense of 'musical performance'. Accordingly, we would like the word *nummer* to be disambiguated into two senses, the first sense being similar to words like *getal* 'number', *cijfer* 'digit' and the second to words like *liedje* 'song', *song* 'song'.

While it is relatively easy for a human language user to distinguish between the two senses, this is a difficult task for a computer. Even worse: the results get blurred because the attributes of both senses (in this example *oneven* and *steengoed*) are grouped together into one sense. This is the main drawback of the syntax-based method. On the other hand, methods that capture semantic dimensions are known to

[3] This is true in particular when using a large context window, cfr. chapter 7.

be useful in disambiguating different senses of a word. Particularly, PROBABILISTIC LATENT SEMANTIC ANALYSIS (PLSA) is known to simultaneously encode various senses of words according to latent semantic dimensions (Hofmann, 1999). In this paper, we want to explore an approach that tries to remedy the shortcomings of the former, syntax-based approach with the benefits of the latter. The intuition in this is that the syntactic features of the syntax-based approach can be disambiguated by the 'latent semantic dimensions' found with the window-based approach.

9.2 Previous Work

9.2.1 Distributional Similarity

There have been numerous approaches for computing the similarity between words from distributional data. Here, we mention some of the most important ones. An extensive overview of distributional similarity is given in chapter 2.

With regard to the first approach – using a context window – we already mentioned LSA (Landauer and Dumais, 1997). In LSA, a term-document matrix is created, containing the frequency of each word in a specific document. This matrix is then decomposed into three other matrices with a mathematical technique called SINGULAR VALUE DECOMPOSITION. The most important dimensions that come out of the SVD are interpreted to represent 'latent semantic dimensions', according to which nouns and documents can be presented more efficiently.

LSA has been criticized for not being the most appropriate data reduction method for textual applications. The SVD underlying the method assumes normally-distributed data, whereas textual count data (such as the term-document matrix) can be more appropriately modeled by other distributional models such as Poisson (Manning and Schütze, 2000, §15.4.3). Successive methods such as PROBABILISTIC LATENT SEMANTIC ANALYSIS (PLSA) (Hofmann, 1999), try to remedy this shortcoming by imposing a proper latent variable model, according to which the values can be estimated. The method we adopt in our research – NON-NEGATIVE MATRIX FACTORIZATION – is similar to PLSA, and adequately remedies this problem as well.

The second approach – using syntactic relations – has been adopted by many researchers, in order to acquire semantically similar words. One of the most important is Lin's (1998a). For Dutch, the approach has been applied by Van der Plas & Bouma (2005).

9.2.2 Discriminating senses

Schütze (1998) uses a disambiguation algorithm – called context-group discrimination – based on the clustering of the context of ambiguous words. The clustering is based on second-order co-occurrence: the contexts of the ambiguous word are similar if the words they in turn co-occur with are similar.

Pantel and Lin (2002) present a clustering algorithm – coined CLUSTERING BY COMMITTEE (CBC) – that automatically discovers word senses from text. The key idea is to first discover a set of tight, unambiguous clusters, to which possibly ambiguous words can be assigned. Once a word has been assigned to a cluster, the features associated with that particular cluster are stripped off the word's vector. This way, less frequent senses of the word can be discovered.

The former approach uses a window-based method; the latter uses syntactic data. But none of the algorithms developed so far have combined both sources in order to discriminate among different senses of a word.

9.3 Methodology

9.3.1 Non-negative Matrix Factorization

In this framework, we will adopt non-negative matrix factorization (NMF) (Lee and Seung, 2000) as a dimensionality reduction algorithm. In NMF, a matrix V is factorized into two other matrices, W and H, with $r \ll n, m$.

$$V_{n\times m} \approx W_{n\times r} H_{r\times m} \tag{9.1}$$

An extensive overview of NMF and its computational details is given in section 3.3 on page 42.

In this framework, we will adopt the NMF algorithm that minimizes the Kullback-Leibler divergence as objective function. In our experience, entropy-based measures tend to work well for natural language. Thus, we want to find the matrices W and H for which the Kullback-Leibler divergence between V and WH (the matrix product of W and H) is the smallest.

We can now straightforwardly apply NMF to create semantic word models. NMF is applied to a frequency matrix, containing bags of words co-occurrence data. The additive property of NMF ensures that semantic dimensions emerge according to which the various words can be classified. Two sample dimensions are shown in example (4). For each dimension, the words with the largest value on that dimension are given. Dimension (a) can be qualified as a 'transport' dimension, and dimension (b) as a 'cooking' dimension.

(4) a. *bus* 'bus', *taxi* 'taxi', *trein* 'train', *halte* 'stop', *reiziger* 'traveler', *perron* 'platform', *tram* 'tram', *station* 'station', *chauffeur* 'driver', *passagier* 'passenger'

b. *bouillon* 'broth', *slagroom* 'cream', *ui* 'onion', *eierdooier* 'egg yolk', *laurierblad* 'bay leaf', *zout* 'salt', *deciliter* 'decilitre', *boter* 'butter', *bleekselderij* 'celery', *saus* 'sauce'

9.3.2 Extending Non-negative Matrix Factorization

We now propose an extension of NMF that combines both the bag of words approach and the syntactic approach. The algorithm finds again latent semantic dimensions, according to which nouns, bag of words contexts and syntactic relations are classified.

Since we are interested in the classification of nouns according to both 'bag-of-words' context and syntactic context, we first construct three matrices that capture the co-occurrence frequency information for each mode. The first matrix contains co-occurrence frequencies of nouns cross-classified by dependency relations, the second matrix contains co-occurrence frequencies of nouns cross-classified by words that appear in the noun's context window, and the third matrix contains co-occurrence frequencies of dependency relations cross-classified by co-occurring context words.

We then apply NMF to the three matrices, but we interleave the separate factorizations: the result of the preceding factorization is used to initialize the factorization of the next matrix. This implies that we need to initialize only three matrices at random; the other three are initialized by calculations of the previous step. The process is represented graphically in figure 9.1.

In the example in figure 9.1, matrix H is initialized at random, and the update of matrix W is calculated. The result of update W is then used to initialize matrix V, and the update of matrix G is calculated. This matrix is used again to initialize matrix U, and the update of matrix F is calculated. This matrix can be used to initialize matrix H, and the process is repeated until convergence.

In (5), an example is given of the kind of semantic dimensions found. This dimension may be coined the 'transport' dimension, as is shown by the top 10 nouns (a), context words (b) and syntactic relations (c).

(5) a. *auto* 'car', *wagen* 'car', *tram* 'tram', *motor* 'motorbike', *bus* 'bus', *metro* 'subway', *automobilist* 'driver', *trein* 'trein', *stuur* 'steering wheel', *chauffeur* 'driver'

b. *auto* 'car', *trein* 'train', *motor* 'motorbike', *bus* 'bus', *rij* 'drive', *chauffeur* 'driver', *fiets* 'bike', *reiziger* 'reiziger', *passagier* 'passenger', *vervoer* 'transport'

c. viertraps$_{adj}$ 'four-stage', verplaats_met$_{obj}$ 'move with', toeter$_{adj}$ 'honk', tank_in_houd$_{obj}$ [parsing error], tank$_{subj}$ 'refuel', tank$_{obj}$ 'refuel', rij_voorbij$_{subj}$ 'pass by', rij_voorbij$_{adj}$ 'pass by', rij_af$_{subj}$ 'drive off', peperduur$_{adj}$ 'very expensive'

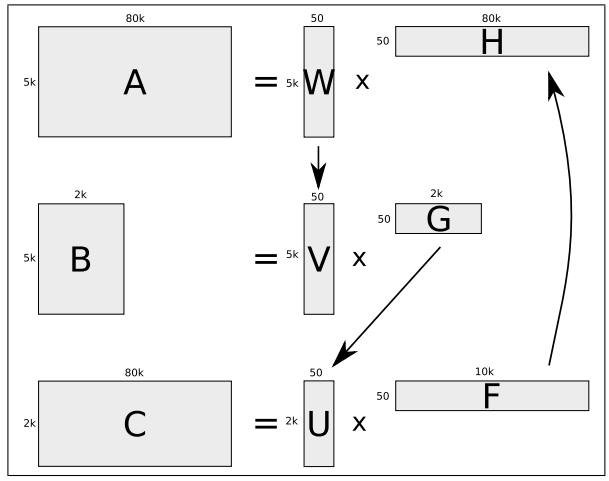

Figure 9.1: A graphical representation of the extended NMF

9.3.3 Sense Subtraction

Next, we want to use the factorization that has been created in the former step for word sense discrimination. The intuition is that we 'switch off' one dimension of an ambiguous word, to reveal possible other senses of the word. From matrix H, we know the importance of each syntactic relation given a dimension. With this knowledge, we can 'subtract' the syntactic relations that are responsible for a certain dimension from the original noun vector:

$$\overrightarrow{v}_{new} = \overrightarrow{v}_{orig}(\overrightarrow{v_1} - \overrightarrow{h}_{dim}) \tag{9.2}$$

Equation 9.2 is an element-wise multiplication that multiplies each feature (syntactic relation) of the original noun vector ($\overrightarrow{v}_{orig}$) with a scaling factor, according to the load of the feature on the subtracted dimension ($\overrightarrow{h}_{dim}$ – the vector of matrix H containing the dimension we want to subtract). $\overrightarrow{v_1}$ is a vector of ones, the size of $\overrightarrow{h}_{dim}$.

9.3.4 A Clustering Framework

The last step is to determine which dimension(s) are responsible for a certain sense of the word. In order to do so, we embed our method in a clustering approach. First, a specific word is assigned to its predominant sense (i.e. the most similar cluster). Next,

the dominant semantic dimension(s) for this cluster are subtracted from the word vector (equation 9.2), and the resulting vector is fed to the clustering algorithm again, to see if other word senses emerge. The dominant semantic dimension(s) can be identified by 'folding in' the cluster centroid into our factorization (so we get a vector $\overrightarrow{w}$ of dimension size r), and applying a threshold to the result (in our experiments a threshold of $\delta = .05$ — so dimensions responsible for $> 5\%$ of the centroid are subtracted).

We used two kinds of clustering algorithms to determine our initial centroids. The first algorithm is a standard K-means algorithm. The second one is the CBC algorithm by Pantel and Lin (2002). The initial vectors to be clustered are weighted using pointwise mutual information (Church and Hanks, 1990).

K-means

First, a standard K-means algorithm is applied to the nouns we want to cluster. This yields a hard clustering, in which each noun is assigned to exactly one (dominant) cluster. In the second step, we try to determine for each noun whether it can be assigned to other, less dominant clusters. First, the salient dimension(s) of the centroid to which the noun is assigned are determined. We compute the centroid of the cluster by averaging the frequencies of all cluster elements except for the target element we want to reassign, and adapt the centroid with pointwise mutual information. After subtracting the salient dimensions from the noun vector, we check whether the vector is reassigned to another cluster centroid (i.e. whether it is more similar to a different centroid). If this is the case, (another instance of) the noun is assigned to the cluster, and we repeat the second step. If there is no reassignment, we continue with the next word. The target element is removed from the centroid to make sure that we only subtract the dimensions associated with the sense of the cluster.

Note that K-means requires setting the number of clusters beforehand, so k is a parameter to be set.

CBC

The second clustering algorithm operates in a similar vein, but instead of using simple K-means, we use Pantel and Lin's CBC algorithm to find the initial centroids (coined COMMITTEES).

In order to find committees, the top k nouns for each noun in the database are clustered with average-link clustering. The clusters are scored and sorted in such a way that preference is given to tight, representative clusters. If the committees do not cover all elements sufficiently, the algorithm recursively tries to find more committees. An elaborate description of the algorithm can be found in Pantel and Lin (2002).

In the second step, we start assigning elements to committees. Once an element is assigned, the salient dimensions are subtracted from the noun vector in the same way as in 9.3.4 (only do we not have to remove any target word from the centroid; committees are supposed to represent tight, unambiguous clusters).

CBC attempts to find the number of committees automatically from the data, so k does not have to be set.

9.4 Examples

9.4.1 Sense Subtraction

In what follows, we will talk about semantic dimensions as, e.g., the 'music' dimension or the 'city' dimension. In the vast majority of the cases, the dimensions are indeed as clear-cut as the transport dimension shown above, so that the dimensions can be rightfully labeled this way.

Two examples are given of how the semantic dimensions that have been found can be used for word sense discrimination. We will consider two ambiguous nouns: *pop*, which can mean 'pop music' as well as 'doll', and *Barcelona*, which can designate either the Spanish city or the Spanish football club.

First, we look up the top dimensions for each noun. Next, we successively subtract the dimensions dealing with a particular sense of the noun, as described in 9.3.3. This gives us three vectors for each noun: the original vector, and two vectors with one of the dimensions eliminated. For each of these vectors, the top ten similar nouns are given, in order to compare the changes brought about.

(6) a. *pop*, *rock*, *jazz*, *meubilair* 'furniture', *popmuziek* 'pop music', *heks* 'witch', *speelgoed* 'toy', *kast* 'cupboard', *servies* '[tea] service', *vraagteken* 'question mark'

b. *pop*, *meubilair* 'furniture', *speelgoed* 'toy', *kast* 'cupboard', *servies* '[tea] service', *heks* 'witch', *vraagteken* 'question mark' *sieraad* 'jewel', *sculptuur* 'sculpture', *schoen* 'shoe'

c. *pop*, *rock*, *jazz*, *popmuziek* 'pop music', *heks* 'witch', *danseres* 'dancer', *servies* '[tea] service', *kopje* 'cup', *house* 'house music', *aap* 'monkey'

Example (6) shows the top similar words for the three vectors of *pop*. In (a), the most similar words to the original vector are shown. In (b), the top dimension (the 'music dimension') has been subtracted from (a), and in (c), the second highest dimension (a 'domestic items' dimension) has been subtracted from (a).

The differences between the three vectors are clear: in vector (a), both senses are mixed together, with 'pop music' and 'doll' items interleaved. In (b), no more music items are present. Only items related to the doll sense are among the top similar words. In (c), the music sense emerges much more clearly, with *rock*, *jazz* and *popmuziek* being the most similar, and a new music term (*house*) showing up among the top ten.

Admittedly, in vector (c), not all items related to the 'doll' sense are filtered out. We believe this is due to the fact that this sense cannot be adequately filtered out by one dimension (in this case, a dimension of 'domestic items' alone), whereas it is much easier to filter out the 'music' sense with only one 'music' dimension. We will try to remedy this in our clustering framework, in which it is possible to subtract multiple dimensions related to one sense.

A second example, the ambiguous proper name *Barcelona*, is given in (7).

(7) a. *Barcelona*, *Arsenal*, *Inter*, *Juventus*, *Vitesse*, *Milaan* 'Milan', *Madrid*, *Parijs* 'Paris', *Wenen* 'Vienna', *München* 'Munich'
b. *Barcelona*, *Milaan* 'Milan', *München* 'Munich', *Wenen* 'Vienna', *Madrid*, *Parijs* 'Paris', *Bonn*, *Praag* 'Prague', *Berlijn* 'Berlin', *Londen* 'London'
c. *Barcelona*, *Arsenal*, *Inter*, *Juventus*, *Vitesse*, *Parma*, *Anderlecht*, *PSV*, *Feyenoord*, *Ajax*

In (a), the two senses of *Barcelona* are clearly mixed up, showing cities as well as football clubs among the most similar nouns. In (b), where the 'football dimension' has been subtracted, only cities show up. In (c), where the 'city dimension' has been subtracted, only football clubs remain.

9.4.2 Clustering Output

In (8), an example of our clustering algorithm with initial K-means clusters is given.

(8) a. *werk* 'work' *beeld* 'image' *foto* 'photo' *schilderij* 'painting' *tekening* 'drawing' *doek* 'canvas' *installatie* 'installation' *afbeelding* 'picture' *sculptuur* 'sculpture' *prent* 'picture' *illustratie* 'illustration' *handschrift* 'manuscript' *grafiek* 'print' *aquarel* 'aquarelle' *maquette* 'scale-model' *collage* 'collage' *ets* 'etching'
b. *werk* 'work' *boek* 'book' *titel* 'title' *roman* 'novel' *boekje* 'booklet' *debuut* 'debut' *biografie* 'biography' *bundel* 'collection' *toneelstuk* 'play' *bestseller* 'bestseller' *kinderboek* 'child book' *autobiografie* 'autobiography' *novelle* 'short story'
c. *werk* 'work' *voorziening* 'service' *arbeid* 'labour' *opvoeding* 'education' *kinderopvang* 'child care' *scholing* 'education' *huisvesting* 'housing' *faciliteit* 'facility' *accommodatie* 'acommodation' *arbeidsomstandigheid* 'working condition'

The example shows three different clusters to which the noun *werk* 'work' is assigned. In (a), *werk* refers to a work of art. In (b), it refers to a written work. In (c), the 'labour' sense of *werk* emerges.

9.5 Evaluation

9.5.1 Methodology

The clustering results have been evaluated according to Dutch EuroWordNet (Vossen, 1998). Precision and recall are calculated by comparing the results to EuroWordNet synsets. The precision is the number of clusters found that correspond to an actual sense of the word. Recall is the number of word senses in EuroWordNet that are found by the algorithm. Our evaluation method is largely the same as the one used by Pantel and Lin (2002).

Both precision and recall are based on wordnet similarity. A number of similarity measures have been developed to calculate semantic similarity in a hierarchical wordnet. Two of those measures – Wu & Palmer's and Lin's – have been presented earlier in chapter 5. In this evaluation, Wu & Palmer's (1994) measure will be adopted. A detailed description of this Wu & Palmer's similarity measure can be found in section 5.3.2 on page 61.

To calculate precision, we apply the same methodology as Pantel and Lin (2002).[4] Let $S(w)$ be the set of EuroWordNet senses. $sim_W(s,u)$, the similarity between a synset s and a word u is then defined as the maximum similarity between s and a sense of u:

$$sim_W(s,u) = \max_{t \varepsilon S(u)} sim(s,t) \tag{9.3}$$

Let c_k be the top k-members of a cluster c, where these are the k most similar members to the centroid of c. $simC(c,s)$, the similarity between s and c, is then defined as the average similarity between s and the top-k members of c:

$$sim_C(s,c) = \frac{\sum_{u \varepsilon c_k} simW(s,u)}{k} \tag{9.4}$$

An assigment of a word w to a cluster c can now be classified as correct if

$$\max_{s \varepsilon S(w)} simC(s,c) > \theta \tag{9.5}$$

and the EuroWordNet sense of w that corresponds to c is

$$\arg\max_{s \varepsilon S(w)} simC(s,c) \tag{9.6}$$

[4] Note, however, that our similarity measure is different. Where Pantel and Lin use Lin's (1998a) measure, we use Wu and Palmer's (1994) measure.

When multiple clusters correspond to the same EuroWordNet sense, only one of them is counted as correct.

Precision of a word w is the percentage of correct clusters to which it is assigned. Recall of a word w is the percentage of senses from EuroWordnet that have a corresponding cluster.[5] Precision and recall of a clustering algorithm is the average precision and recall of all test words.

9.5.2 Experimental Design

We have applied the interleaved NMF presented in section 9.3.2 to Dutch, using the TWENTE NIEUWS CORPUS (Ordelman, 2002), containing $>$ 500M words of Dutch newspaper text. The corpus is consistently divided into paragraphs, which have been used as the context window for the bag-of-words mode. The corpus has been parsed by the Dutch dependency parser Alpino (van Noord, 2006), and dependency triples have been extracted. Next, the three matrices needed for our method have been constructed: one containing nouns by dependency relations (5K $\times$ 80K), one containing nouns by context words (5K $\times$ 2K) and one containing dependency relations by context words (80K $\times$ 2K). We did 200 iterations of the algorithm, factorizing the matrices into 50 dimensions. The NMF algorithm has been implemented in Matlab.

For the evaluation, we use all the words that appear in our original clustering input as well as in EuroWordNet. This yields a test set of 3683 words.

9.5.3 Results

Table 9.1 shows precision and recall figures for four different algorithms, according to two similarity thresholds θ (equation 9.5). kmeans$_{nmf}$ describes the results of our algorithm with K-means clusters, as described in section 9.3.4. CBC describes the results of our algorithm with the CBC committees, as described in section 9.3.4. For comparison, we have also included the results of a standard K-means clustering (kmeans$_{orig}$, $k = 600$), and the original CBC algorithm (CBC$_{orig}$) as described by Pantel and Lin (2002).

The results show the same tendency across all similarity thresholds: kmeans$_{nmf}$ has a high precision, but lower recall compared to CBC$_{orig}$. Still the recall is higher compared to standard K-means, which indicates that the algorithm is able to find multiple senses of nouns, with high precision. The results of CBC$_{nmf}$ are similar to

[5] Our notion of recall is slightly different from the one used by Pantel and Lin, as they use 'the number of senses in which w was used in the corpus' as gold standard. This information, as they acknowledge, is difficult to get at, so we prefer to use the sense information in EuroWordNet.

		threshold θ	
		.40 (%)	.60 (%)
kmeans$_{nmf}$	prec.	78.97	55.16
	rec.	63.90	44.77
CBC$_{nmf}$	prec.	82.70	54.87
	rec.	60.27	40.51
kmeans$_{orig}$	prec.	86.13	58.97
	rec.	60.23	41.80
CBC$_{orig}$	prec.	44.94	29.74
	rec.	69.61	48.00

Table 9.1: Precision and recall percentages for four different algorithms according to two similarity thresholds

the results of kmeans$_{orig}$, indicating that few words are reassigned to multiple clusters when using CBC committees with our method.

Obviously, kmeans$_{orig}$ scores best with regard to precision, but worst with regard to recall. CBC$_{orig}$ finds most senses (highest recall), but precision is considerably worse.

The fact that recall is already quite high with standard K-means clustering indicates that the evaluation is skewed towards nouns with only one sense, possibly due to a lack of coverage in EuroWordNet. In future work, we specifically want to evaluate the discrimination of ambiguous words. Also, we want to make use of the new Cornetto Database[6], a successor of EuroWordNet for Dutch which is currently under development.

Still, the evaluation shows that our method provides a genuine way of finding multiple senses of words, while retaining high precision. Especially the method using a simple K-means clustering performs particularly well. The three way data allows the algorithm to put its finger on the particular sense of a centroid, and adapt the feature vector of a possibly ambiguous noun accordingly.

[6] `http://www.let.vu.nl/onderzoek/projectsites/cornetto/index.html`

9.6 Conclusion & Future Work

In this paper, an extension of NMF has been presented that combines both bag of words data and syntactic data in order to find latent semantic dimensions according to which both words and syntactic relations can be classified. The use of three-way data allows one to determine which dimension(s) are responsible for a certain sense of a word, and adapt the corresponding feature vector accordingly, 'subtracting' one sense to discover another one. When embedded in a clustering framework, the method provides a fully automatic way to discriminate the various senses of words. The evaluation against EuroWordNet shows that the algorithm is genuinely able to disambiguate the features of a given word, and accordingly its word senses.

We conclude with some issues for future work. First of all, we would like to test the method that has been explored in this paper with other evaluation frameworks. We already mentioned the focus on ambiguous nouns, and the use of the new Cornetto database for Dutch. Next, we would like to work out a proper probabilistic framework for the 'subtraction' of dimensions. At this moment, the subtraction (using a cut-off) is somewhat ad hoc. A probabilistic modeling of this intuition might lead to an improvement.

And finally, we would like to use the results of our method to learn selectional preferences. Our method is able to discriminate the syntactic features that are linked to a particular word sense. If we can use the results to improve a parser's performance, this will also provide an external evaluation of the algorithm.

Chapter 10

Selectional Preferences[1]

10.1 Introduction

Distributional similarity methods have proven to be a valuable tool for the induction of semantic similarity. The aggregate of a word's contexts generally provides enough information to compute its meaning, viz. its semantic similarity or relatedness to other words.

Up till now, most algorithms use two-way co-occurrence data to compute the meaning of words. A word's meaning might for example be computed by looking at:

- the various documents that the word appears in (words $\times$ documents);
- a bag of words context window around the word (words $\times$ context words);
- the dependency relations that the word appears with (words $\times$ dependency relations).

The extracted data – representing the co-occurrence frequencies of two different entities – is encoded in a matrix. Co-occurrence frequencies, however, need not be pairwise. One can easily imagine situations where it is desirable to investigate co-occurrence frequencies of three modes and beyond. In an information retrieval context, one such situation might be the investigation of *words* $\times$ *documents* $\times$ *authors*. In an NLP context, one might want to investigate *words* $\times$ *dependency relations* $\times$ *bag of word context words*, or *verbs* $\times$ *subjects* $\times$ *direct objects*.

[1] The research presented in this chapter has been published as Van de Cruys (2009).

Note that it is not possible to investigate the three-way co-occurrences in a matrix representation form. It is possible to capture the co-occurrence frequencies of a verb with its subjects and its direct objects, but one cannot capture the co-occurrence frequencies of the verb appearing with the subject and the direct object *at the same time*. When the actual three-way co-occurrence data is 'matricized', valuable information is thrown-away. To be able to capture the mutual dependencies among the three modes, we will make use of a generalized *tensor* representation.

Two-way co-occurrence models (such as latent semantic analysis) have often been augmented with some form of dimensionality reduction in order to counter noise and overcome data sparseness. We will also make use of a dimensionality reduction algorithm appropriate for tensor representations.

10.2 Previous Work

10.2.1 Selectional Preferences & Verb Clustering

Selectional preferences have been a popular research subject in the NLP community. One of the first to automatically induce selectional preferences from corpora was Resnik Resnik (1996). Resnik generalizes among nouns by using WordNet noun synsets as clusters. He then calculates the *selectional preference strength* of a specific verb in a particular relation by computing the Kullback-Leibler divergence between the cluster distribution of the verb and the aggregate cluster distribution:

$$S_v = \sum_c p(c \mid v)\ log \frac{p(c \mid v)}{p(c)} \tag{10.1}$$

The *selectional association* is then the contribution of a particular cluster to the verb's preference strength:

$$A_{v,c} = \frac{p(c \mid v)\ log \frac{p(c|v)}{p(c)}}{S_v} \tag{10.2}$$

The model's generalization relies entirely on WordNet; there is no generalization among the verbs.[2]

The research in this paper is related to previous work on clustering. Pereira et al. (1993) use an information-theoretic based clustering approach, clustering nouns according to their distribution as direct objects among verbs, conditioned on a set of latent semantic classes.

[2]Other notable approaches using WordNet for selectional preference induction are Abe and Li's (1996) and Clark and Weir's (2001).

$$p(v,n) = \sum_c p(c,v,n) = \sum_c p(c)p(v \mid c)p(n \mid c) \tag{10.3}$$

Their model is an asymmetric, one-sided clustering model: only the direct objects are clustered, there is no clustering among the verbs.

Rooth et al. (1999) use an EM-based technique to induce a clustering based on the co-occurrence frequencies of verbs with their subjects and direct objects. Their clustering model is the same as the one used by Pereira et al. (1993), but they embed the model in a formal EM-clustering framework. Rooth et al.'s (1999) clustering is two-sided: the verbs as well as the subjects/direct objects are clustered. We will use a similar model for evaluation purposes.

Recent approaches using distributional similarity methods for the induction of selectional preferences are the ones by Erk (2007), Bhagat et al. (2007) and Basili et al. (2007). Erk (2007) uses corpus-based similarity measures for the induction of selectional preferences. The selectional preference S_{r_p} for an argument slot r of a particular predicate p and a possible headword w_0 is computed as the weighted sum of similarities between w_0 and the headwords seen as argument fillers for the predicate ($Seen(r_p)$); wt_{r_p} is an appropriate weighting function.

$$S_{r_p}(w_0) = \sum_{w \in Seen(r_p)} sim(w_0,w) \cdot wt_{r_p}(w) \tag{10.4}$$

Both Bhagat et al. (2007) and Basili et al. (2007) investigate the induction of selectional preferences in the context of textual entailment tasks.

This research differs from the approaches mentioned above by its use of multi-way data: where the approaches above limit themselves to two-way co-occurrences, this research will focus on co-occurrences for multi-way data.

10.2.2 Factorization Algorithms

Two-way Factorizations

One of the best known factorization algorithms is principal component analysis (PCA, Pearson (1901)). PCA transforms the data into a new coordinate system, yielding the best possible fit in a least square sense given a limited number of dimensions. Singular value decomposition (SVD) is the generalization of the eigenvalue decomposition used in PCA (Wall, Rechtsteiner, and Rocha, 2003). In information retrieval, singular value decomposition has been applied in latent semantic analysis (LSA, Landauer and Dumais (1997), Landauer et al. (1998)). In LSA, a term-document matrix is created, containing the frequency of each word in a specific document. This matrix is then decomposed

into three other matrices with SVD. The most important dimensions that come out of the SVD allegedly represent 'latent semantic dimensions', according to which nouns and documents can be represented more efficiently.

LSA has been criticized for a number of reasons, one of them being the fact that the factorization contains negative numbers. It is not clear what negativity on a semantic scale should designate. Subsequent methods such as probabilistic latent semantic analysis (PLSA, Hofmann (1999)) and non-negative matrix factorization (NMF, Lee and Seung (2000)) remedy these problems, and indeed get much more clear-cut semantic dimensions.

Three-way Factorizations

To be able to cope with three-way data, several algorithms have been developed as multilinear generalizations of the SVD. In statistics, three-way component analysis has been extensively investigated (for an overview, see Kiers and van Mechelen (2001)). The two most popular methods are parallel factor analysis (PARAFAC, Harshman (1970), Carroll and Chang (1970)) and three-mode principal component analysis (3MPCA, Tucker (1966)), also called higher order singular value decomposition (HOSVD, De Lathauwer et al. (2000)). Three-way factorizations have been applied in various domains, such as psychometry and image recognition (Vasilescu and Terzopoulos, 2002). In information retrieval, three-way factorizations have been applied to the problem of link analysis (Kolda and Bader, 2006).

One last important method dealing with multi-way data is non-negative tensor factorization (NTF, Shashua and Hazan (2005)). NTF is a generalization of non-negative matrix factorization, and can be considered an extension of the PARAFAC model with the constraint of non-negativity (cfr. infra).

One of the few papers that has investigated the application of tensor factorization for NLP is Turney (2007), in which a three-mode tensor is used to compute the semantic similarity of words. The method achieves 83.75% accuracy on the TOEFL synonym questions.

10.3 Methodology

10.3.1 Tensors

Distributional similarity methods usually represent co-occurrence data in the form of a *matrix*. This form is perfectly suited to represent two-way co-occurrence data, but for co-occurrence data beyond two modes, we need a more general representation. The generalization of a matrix is called a *tensor*. A tensor is able to encode co-occurrence

data of any n modes. Figure 10.1 shows a graphical comparison of a matrix and a tensor with three modes – although a tensor can easily be generalized to more than three modes.

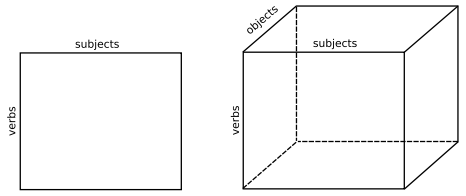

Figure 10.1: Matrix representation vs. tensor representation

10.3.2 Non-negative Tensor Factorization

In order to create a succinct and general model of the extracted data, a statistical dimensionality reduction technique called non-negative tensor factorization (NTF) is applied to the data. The NTF model is similar to the PARAFAC analysis – popular in areas such as psychology and bio-chemistry – with the constraint that all data needs to be non-negative (i.e. ≥ 0).

Parallel factor analysis (PARAFAC) is a multilinear analogue of the singular value decomposition (SVD) used in latent semantic analysis. The key idea is to minimize the sum of squares between the original tensor and the factorized model of the tensor. For the three mode case of a tensor $T \in \mathbb{R}^{D_1 \times D_2 \times D_3}$ this gives equation 10.5, where k is the number of dimensions in the factorized model and $\circ$ denotes the outer product.

$$\min_{x_i \in \mathbb{R}^{D1}, y_i \in \mathbb{R}^{D2}, z_i \in \mathbb{R}^{D3}} \| T - \sum_{i=1}^{k} x_i \circ y_i \circ z_i \|_F^2 \tag{10.5}$$

With non-negative tensor factorization, the non-negativity constraint is enforced, yielding a model like the one in equation 10.6:

$$\min_{x_i \in \mathbb{R}^{D1}_{\geq 0}, y_i \in \mathbb{R}^{D2}_{\geq 0}, z_i \in \mathbb{R}^{D3}_{\geq 0}} \| T - \sum_{i=1}^{k} x_i \circ y_i \circ z_i \|_F^2 \tag{10.6}$$

The algorithm results in three matrices, indicating the loadings of each mode on the factorized dimensions. The model is represented graphically in figure 10.2, visualizing the fact that the PARAFAC decomposition consists of the summation over the outer products of n (in this case three) vectors.

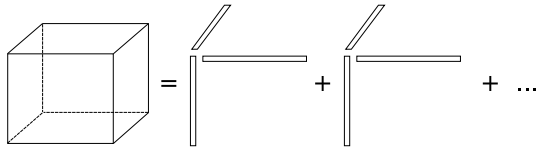

Figure 10.2: Graphical representation of the NTF as the sum of outer products

Computationally, the non-negative tensor factorization model is fitted by applying an alternating least-squares algorithm. In each iteration, two of the modes are fixed and the third one is fitted in a least squares sense. This process is repeated until convergence.[3]

10.3.3 Applied to Language Data

The model can straightforwardly be applied to language data. In this part, we describe the factorization of *verbs* $\times$ *subjects* $\times$ *direct objects* co-occurrences, but the example can easily be substituted with other co-occurrence information. Moreover, the model need not be restricted to 3 modes; it is very well possible to go to 4 modes and beyond — as long as the computations remain feasible.

The NTF decomposition for the *verbs* $\times$ *subjects* $\times$ *direct objects* co-occurrences into the three loadings matrices is represented graphically in figure 10.3.[4] By applying the NTF model to three-way (s,v,o) co-occurrences, we want to extract a generalized selectional preference model, and eventually even induce some kind of frame semantics (in the broad sense of the word).

In the resulting factorization, each verb, subject and direct object gets a loading value for each factor dimension in the corresponding loadings matrix. The original value for a particular (s,v,o) triple x_{svo} can then be reconstructed with equation 10.7.

$$x_{svo} = \sum_{i=1}^{k} s_{si} v_{vi} o_{oi} \tag{10.7}$$

To reconstruct the selectional preference value for the triple $(man, bite, dog)$, for example, we look up the subject vector for *man*, the verb vector for *bite* and the direct

[3]The algorithm has been implemented in MATLAB, using the Tensor Toolbox for sparse tensor calculations (Bader and Kolda, 2009).

[4]Note that the representation of NTF as three loadings matrices (presented in this figure) is equivalent to the representation as the sum of outer products (presented in figure 10.2).

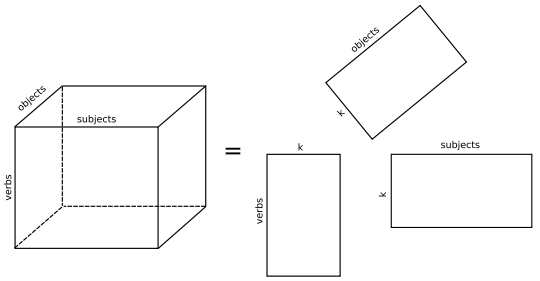

Figure 10.3: Graphical representation of the NTF for language data

object vector for *dog*. Then, for each dimension i in the model, we multiply the ith value of the three vectors. The sum of these values is the final preference value.

10.4 Results

10.4.1 Setup

The approach described in the previous section has been applied to Dutch, using the Twente Nieuws Corpus (Ordelman, 2002), a 500M word corpus of Dutch newspaper texts. The corpus has been parsed with the Dutch dependency parser Alpino (van Noord, 2006), and three-way co-occurrences of verbs with their respective subject and direct object relations have been extracted. As dimension sizes, the 1K most frequent verbs were used, together with the 10K most frequent subjects and 10K most frequent direct objects, yielding a tensor of $1\text{K} \times 10\text{K} \times 10\text{K}$. The resulting tensor is very sparse, with only 0.0002% ($\frac{1}{5000}$th) of the values being non-zero.

The tensor has been adapted with a straightforward extension of pointwise mutual information (Church and Hanks, 1990) for three-way co-occurrences, following equation 10.8. Negative values are set to zero.[5]

$$MI3(x,y,z) = \log \frac{p(x,y,z)}{p(x)p(y)p(z)} \tag{10.8}$$

The resulting matrix has been factorized into k dimensions (varying between 50 and 300) with the NTF algorithm described in section 10.3.2.

[5]This is not just an ad hoc conversion to enforce non-negativity. Negative values indicate a smaller co-occurrence probability than the expected number of co-occurrences. Setting those values to zero proves beneficial for similarity calculations (see e.g. Bullinaria and Levy (2007)).

10.4.2 Examples

Tables 10.1 to 10.5 show example dimensions that have been found by the algorithm with $k = 100$. Each example gives the top 10 subjects, verbs and direct objects for a particular dimension, together with the score for that particular dimension.

Table 10.1 shows the induction of a 'police action' frame, with police authorities as subjects, police actions as verbs and patients of the police actions as direct objects.

subjects	su_s	**verbs**	v_s	**objects**	obj_s
politie 'police'	.99	*houd_aan* 'arrest'	.64	*verdachte* 'suspect'	.16
agent 'policeman'	.07	*arresteer* 'arrest'	.63	*man* 'man'	.16
autoriteit 'authority'	.05	*pak_op* 'run in'	.41	*betoger* 'demonstrator'	.14
Justitie 'Justice'	.05	*schiet_dood* 'shoot'	.08	*relschopper* 'rioter'	.13
recherche 'detective force'	.04	*verdenk* 'suspect'	.07	*raddraaiers* 'instigator'	.13
marechaussee 'military police'	.04	*tref_aan* 'find'	.06	*overvaller* 'raider'	.13
justitie 'justice'	.04	*achterhaal* 'overtake'	.05	*Roemeen* 'Romanian'	.13
arrestatieteam 'special squad'	.03	*verwijder* 'remove'	.05	*actievoerder* 'campaigner'	.13
leger 'army'	.03	*zoek* 'search'	.04	*hooligan* 'hooligan'	.13
douane 'customs'	.02	*spoor_op* 'track'	.03	*Algerijn* 'Algerian'	.13

Table 10.1: Top 10 subjects, verbs and direct objects for the 'police action' dimension

In table 10.2, a legislation dimension is induced, with legislative bodies as subjects,[6] legislative actions as verbs, and mostly law (proposals) as direct objects. Note that some direct objects (e.g. 'minister') also designate persons that can be the object of a legislative act.

Table 10.3 depicts a dimension of 'war deployment'. The dimension contains military powers (countries, military organizations and leaders) that deploy (or remove) particular military forces.

Table 10.4 shows a 'publishing' dimension. The subjects contain writers (persons and bodies), which publish (in a broad sense) textual works.

Table 10.5, finally, is clearly an exhibition dimension, with verbs describing actions of display and trade that art institutions (subjects) can perform on works of art (objects).

These are not the only sensible dimensions that have been found by the algorithm. A quick qualitative evaluation indicates that about 44 dimensions contain similar, framelike semantics. In another 43 dimensions, the semantics are less clear-cut (single verbs or expressions account for one dimension, or different senses of a verb get mixed

[6]Note that VVD, D66, PvdA and CDA are Dutch political parties.

subjects	su_s	verbs	v_s	objects	obj_s
meerderheid 'majority'	.33	*steun* 'support'	.83	*motie* 'motion'	.63
VVD	.28	*dien_in* 'submit'	.44	*voorstel* 'proposal'	.53
D66	.25	*neem_aan* 'pass'	.23	*plan* 'plan'	.28
Kamermeerderheid 'Chamber majority'	.25	*wijs_af* 'reject'	.17	*wetsvoorstel* 'bill'	.19
fractie 'party'	.24	*verwerp* 'reject'	.14	*hem* 'him'	.18
PvdA	.23	*vind* 'think'	.08	*kabinet* 'cabinet'	.16
CDA	.23	*aanvaard* 'accepts'	.05	*minister* 'minister'	.16
Tweede Kamer 'Second Chamber'	.21	*behandel* 'treat'	.05	*beleid* 'policy'	.13
partij 'party'	.20	*doe* 'do'	.04	*kandidatuur* 'candidature'	.11
Kamer 'Chamber'	.20	*keur_goed* 'pass'	.03	*amendement* 'amendment'	.09

Table 10.2: Top 10 subjects, verbs and direct objects for the 'legislation' dimension

subjects	su_s	verbs	v_s	objects	obj_s
regering 'government'	.26	*stuur* 'send'	.72	*troep* 'troop'	.82
VS 'US'	.26	*trek_terug* 'withdraw'	.67	*militair* 'soldier'	.42
Nederland 'Netherlands'	.25	*zet_in* 'deploy'	.14	*soldaat* 'soldier'	.21
president 'president'	.23	*lever* 'supply'	.07	*delegatie* 'delegation'	.12
leger 'army'	.22	*heb* 'have'	.06	*leger* 'army'	.12
land 'country'	.20	*haal_weg* 'remove'	.03	*waarnemer* 'observer'	.08
NAVO 'NATO'	.20	*beschikbaar stel* 'make available'	.03	*marinier* 'marine'	.08
Indonesië 'Indonesia'	.19	*zeg_toe* 'promise'	.03	*grondtroepen* 'ground forces'	.08
Verenigde_Staten 'United States'	.19	*ruim_op* 'clear'	.02	*gezant* 'envoy'	.07
Groot-Brittannië 'Great Britain'	.18	*bied_aan* 'offer'	.02	*versterking* 'reinforcement'	.07

Table 10.3: Top 10 subjects, verbs and direct objects for the 'war movement' dimension

up). 13 dimensions are not so much based on semantic characteristics, but rather on syntax (e.g. fixed expressions and pronomina).

The qualitative evaluation indicates that quite some dimensions indeed do not contain framelike semantics, but those dimensions do contain information that may be useful for selectional preference induction. Such a dimension is shown in table 10.6. It shows an example dimension in which practically all of the dimension's mass is attributed to one particular expression: *een rol spelen* 'to play a role'. The subject slot is more spread out: different kind of things might play a role – each with a fairly low

subjects	su_s	verbs	v_s	objects	obj_s
hij 'he'	.62	*schrijf* 'write'	.87	*boek* 'book'	.30
die 'who'	.41	*publiceer* 'publish'	.33	*roman* 'novel'	.21
ze 'she'	.32	*zing* 'sing'	.14	*brief* 'letter'	.20
ik 'I'	.30	*lees_voor* 'read aloud'	.09	*gedicht* 'poem'	.18
zij 'she'	.19	*wijd* 'devote'	.09	*tekst* 'text'	.17
auteur 'author'	.16	*vertaal* 'translate'	.09	*essay* 'essay'	.17
je 'you'	.14	*bewerk* 'adapt'	.08	*stuk* 'piece'	.16
journalist 'journalist'	.13	*voltooi* 'finish'	.07	*artikel* 'article'	.16
schrijver 'writer'	.13	*componeer* 'compose'	.06	*biografie* 'biography'	.15
krant 'newspaper'	.09	*presenteer* 'present'	.06	*verhaal* 'story'	.14

Table 10.4: Top 10 subjects, verbs and direct objects for the 'publishing' dimension

subjects	su_s	verbs	v_s	objects	obj_s
tentoonstelling 'exhibition'	.50	*toon* 'display'	.72	*schilderij* 'painting'	.47
expositie 'exposition'	.49	*omvat* 'cover'	.63	*werk* 'work'	.46
galerie 'gallery'	.36	*bevat* 'contain'	.18	*tekening* 'drawing'	.36
collectie 'collection'	.29	*presenteer* 'present'	.17	*foto* 'picture'	.33
museum 'museum'	.27	*laat* 'let'	.07	*sculptuur* 'sculpture'	.25
oeuvre 'oeuvre'	.22	*koop* 'buy'	.07	*aquarel* 'aquarelle'	.20
Kunsthal	.19	*bezit* 'own'	.06	*object* 'object'	.19
kunstenaar 'artist'	.15	*zie* 'see'	.05	*beeld* 'statue'	.12
dat 'that'	.12	*koop_aan* 'acquire'	.05	*overzicht* 'overview'	.12
hij 'he'	.10	*in huis heb* 'own'	.04	*portret* 'portrait'	.11

Table 10.5: Top 10 subjects, verbs and direct objects for the 'exhibition' dimension

probability.

10.4.3 Evaluation

The results of the NTF model have been quantitatively evaluated in a pseudo-disambiguation task, similar to the one used by Rooth et al. (1999). It is used to evaluate the generalization capabilities of the algorithm. The task is to judge which subject (s or s') and direct object (o or o') are more likely for a particular verb v, where (s,v,o) is a combination drawn from the corpus, and s' and o' are a subject and direct object randomly drawn from the corpus. A triple is considered correct if the algorithm prefers both s and o over their counterparts s' and o' (so the (s,v,o) triple – that appears in the test corpus – is preferred over the triples (s',v,o'), (s',v,o) and (s,v,o')). Table 10.7 shows three examples from the pseudo-disambiguation task.

subjects	su_s	verbs	v_s	objects	obj_s
naamsbekendheid 'fame'	.04	speel 'play'	1.00	rol 'role'	1.00
aard 'nature'	.04	lever_op 'yield'	.00	hoofdrol 'leading part'	.03
nut 'use'	.04	onderzoek 'research'	.00	die 'who'	.00
hygiëne 'hygiene'	.04	zie 'see'	.00	stroming 'movement'	.00
eer_wraak 'revenge'	.04	neem 'take'	.00	hoofd 'head'	.00
schaamte 'shame'	.04	vertolk 'express'	.00	religie 'religion'	.00
institutie 'institution'	.04	dring_terug 'push back'	.00	werk 'work'	.00
Cultuur 'Culture'	.04	hekel 'aversion'	.00	traditie 'tradition'	.00
verdeling 'division'	.04	krijg 'get'	.00	overheid 'government'	.00
verbinding 'connection'	.04	onderstreep 'underline'	.00	iedereen 'everyone'	.00

Table 10.6: Top 10 subjects, verbs and direct objects for the 'play a role' dimension

s	v	o	s'	o'
jongere	*drink*	*bier*	*coalitie*	*aandeel*
'youngster'	'drink'	'beer'	'coalition'	'share'
werkgever	*riskeer*	*boete*	*doel*	*kopzorg*
'employer'	'risk'	'fine'	'goal'	'worry'
directeur	*zwaai*	*scepter*	*informateur*	*vodka*
'manager'	'sway'	'sceptre'	'informer'	'wodka'

Table 10.7: Three examples from the pseudo-disambiguation evaluation task's test set

Four different models have been evaluated. The first two models are tensor factorization models. The first model is the NTF model, as described in section 10.3.2. The second model is the original PARAFAC model, without the non-negativity constraints.

The other two models are matrix factorization models. The third model is the non-negative matrix factorization (NMF) model, and the fourth model is the singular value decomposition (SVD). For these models, a matrix has been constructed that contains the pairwise co-occurrence frequencies of verbs by subjects as well as direct objects. This gives a matrix of 1K verbs by 10K subjects + 10K direct objects (1K $\times$ 20K). The matrix has been transformed using pointwise mutual information.

The models have been evaluated with 10-fold cross-validation. The corpus contains 298,540 different (s,v,o) co-occurrences. Those have been randomly divided into 10 equal parts. So in each fold, 268,686 co-occurrences have been used for training, and 29,854 have been used for testing. The accuracy results of the evaluation are given in table 10.8.

The results clearly indicate that the NTF model outperforms all the other models.

	dimensions		
	50 (%)	100 (%)	300 (%)
NTF	**89.52** $\pm$ 0.18	**90.43** $\pm$ 0.14	**90.89** $\pm$ 0.16
PARAFAC	85.57 $\pm$ 0.25	83.58 $\pm$ 0.59	80.12 $\pm$ 0.76
NMF	81.79 $\pm$ 0.15	78.83 $\pm$ 0.40	75.74 $\pm$ 0.63
SVD	69.60 $\pm$ 0.41	62.84 $\pm$ 1.30	45.22 $\pm$ 1.01

Table 10.8: Results of the 10-fold cross-validation for the NTF, PARAFAC, NMF and SVD model for 50, 100 and 300 dimensions (averages and standard deviation)

The model achieves the best result with 300 dimensions, but the differences between the different NTF models are not very large – all attaining scores around 90%.

The PARAFAC results indicate the fitness of tensor factorization for the induction of three-way selectional preferences. Even without the constraint of non-negativity, the model outperforms the matrix factorization models, reaching a score of about 85%. The model deteriorates when more dimensions are used.

Both matrix factorization models perform worse than their tensor factorization counterparts. The NMF still scores reasonably well, indicating the positive effect of the non-negativity constraint. The simple SVD model performs worst, reaching a score of about 70% with 50 dimensions.

10.5 Conclusion and Future Work

This paper has presented a novel method that is able to investigate three-way co-occurrences. Other distributional methods deal almost exclusively with pairwise co-occurrences. The ability to keep track of multi-way co-occurrences opens up new possibilities and brings about interesting results. The method uses a factorization model – non-negative tensor factorization – that is suitable for three way data. The model is able to generalize among the data and overcome data sparseness.

The method has been applied to the problem of selectional preference induction. The results indicate that the algorithm is able to induce selectional preferences, leading to a broad kind of frame semantics. The quantitative evaluation shows that use of three-way data is clearly beneficial for the induction of three-way selectional preferences. The tensor models outperform the simple matrix models in the pseudo-disambiguation task. The results also indicate the positive effect of the non-negativity constraint: both models with non-negative constraints outperform their non-constrained counterparts.

The results as well as the evaluation indicate that the method presented here is a promising tool for the investigation of NLP topics, although more research and a more thorough evaluation would be desirable.

There is quite some room for future work. First of all, we want to further investigate the usefulness of the method for selectional preference induction. One of the most important extensions is the inclusion of other dependency relations in our model, apart from subjects and direct objects (thus making use of tensors with more than 3 modes).

Secondly, there is room for improvement and further research with regard to the tensor factorization model. The model presented here minimizes the sum of squared distance. This is, however, not the only objective function possible. Another possibility is the minimization of the Kullback-Leibler divergence. Minimizing the sum of squared distance assumes normally distributed data, and language phenomena are rarely normally distributed. Other objective functions – such as the minimization of the Kullback-Leibler divergence – might be able to capture the language structures much more adequately. We specifically want to stress this second line of future research as one of the most promising and exciting ones.

Finally, the model presented here is not only suitable for selectional preference induction. There are many problems in NLP that involve three-way co-occurrences. In future work, we want to apply the NTF model presented here to other problems in NLP, the most important one being word sense discrimination.

Conclusion

In this dissertation, we have investigated the extraction of semantic similarity using distributional similarity techniques. In the first part, we investigated what kind of information might be able to provide cues about the semantics of words, and we determined that the context of a word is able to inform us about its semantics. We established three different kinds of context – a document-based context, a window-based context, and a syntax-based context – and we provided a formalization of the notion of context, so that the semantics of a word can be formally computed. Next, we gave an overview of two dimensionality reduction algorithms – singular value decomposition and non-negative matrix factorization – two algorithms that reduce the vast number of overlapping feature dimensions to a limited number of generalized dimensions that might be able to capture inherent semantics characteristics present in the data. We also provided an overview of tensor algebra and the multilinear generalization of non-negative matrix factorization in tensor space – non-negative tensor factorization. Using tensor algebra, we are not limited to using co-occurrences in two modes; three-way methods allow us to analyze multi-way co-occurrences, which allow us to capture the semantic information present in the data in a more informed and thorough way.

In the second part of this dissertation, we provided a quantitative overview of three different groups of models of semantic similarity, constructed according to the three different notions of context. We experimented with different parameters, and determined what models and parameters yield the best results according to three different evaluation schemes. In the first two evaluation schemes – the evaluation of wordnet-based similarity and the evaluation of cluster quality – we evaluated the models' ability to extract tight, synonym-like similarity. In the third evaluation scheme – the evaluation of domain coherence – we evaluated the models' ability to extract more loosely related, topical similarity. In all evaluations, we paid special attention to the effects of dimensionality reduction algorithms.

In the third part, we developed a number of applications that make use of the

techniques (and the resulting semantic resources) presented in the first part. In the first application, we used an automatically induced lexico-semantic clustering to extract multi-word expressions based on their non-compositionality. In the second application, we described a technique that combines a syntactic context and a large window-based context in a factorization framework, in order to discriminate among the different senses of nouns. And in the third application, we explored three-way methods in combination with non-negative tensor factorization in order to induce three-way selectional preferences.

We are now able to answer the three research questions that were formulated in the introduction of this dissertation. In the evaluation part of this dissertation, we have shown that different notions of context lead to different kinds of similarity. We concluded that syntax-based models and window-based models with a small window are best suited for the extraction of tight, synonym-like similarity. This was indicated by their performance on the wordnet-based similarity evaluation and the evaluation of cluster quality. By nature, words that are tightly similar are also topically coherent, which was indicated by the good performance of these models on the third evaluation task. On the other hand, the document-based models and window-based models with large windows size – the models that did not perform well on the extraction of tight similarity – still performed reasonably well on the third evaluation task. We concluded that a tight context (syntax-based or small window-based) leads to tight, synonym-like similarity, whereas a broad context (document-based or large window-based) captures more loosely related topically similar words. We also showed that an automatically induced clustering based on syntactic context allows us to extract multi-word expressions based on their non-compositionality, and that a tight, syntactic context and a broad window-based context can be combined in a non-negative matrix factorization framework in order to discriminate the different senses of a noun.

The answer to our second research question is mixed. On the one hand, our quantitative evaluation indicated that dimensionality reduction algorithms do not bring about a large improvement in the extraction of semantic similarity. Dimensionality reduction was only beneficial for the document-based model. In the other models, dimensionality reduction did not improve or even hurt the performance of the models. The performance of the non-negative matrix factorization models in the quantitative evaluation was particularly disappointing.

On the other hand, dimensionality reduction algorithms did prove beneficial in a number of applications. Non-negative matrix factorization allowed us to induce a number of semantic dimensions according to which context words and syntactic relations could be classified. These dimensions in turn allowed us to subtract dimensions related to a specific sense of a noun, so that other senses of the word become apparent. The multilinear generalization of non-negative matrix factorization – non-negative tensor

factorization – was also beneficial for the induction of selectional preferences.

Which brings us to our third research question: we have demonstrated the usefulness of three-way methods for the induction of selectional preferences. The quantitative evaluation shows that the use of three-way data is beneficial for the induction of three-way selectional preferences. The tensor models outperform the simple matrix models in the pseudo-disambiguation task. The results also indicate the positive effect of the non-negativity constraint: both models with non-negative constraints outperform their non-constrained counterparts. The results indicate that three-way methods are a promising tool for the investigation of NLP topics.

Appendix A

Clustering Tasks

A.1 Concrete noun categorization

noun	translation	$class_6$	$class_3$	$class_2$
kip	chicken	bird	animal	natural
arend	eagle	bird	animal	natural
eend	duck	bird	animal	natural
zwaan	swan	bird	animal	natural
uil	owl	bird	animal	natural
pinguïn	penguin	bird	animal	natural
pauw	peacock	bird	animal	natural
hond	dog	groundAnimal	animal	natural
olifant	elephant	groundAnimal	animal	natural
koe	cow	groundAnimal	animal	natural
kat	cat	groundAnimal	animal	natural
leeuw	lion	groundAnimal	animal	natural
varken	pig	groundAnimal	animal	natural
slak	snail	groundAnimal	animal	natural
schildpad	turtle	groundAnimal	animal	natural
kers	cherry	fruitTree	vegetable	natural
banaan	banana	fruitTree	vegetable	natural
peer	pear	fruitTree	vegetable	natural
ananas	pineapple	fruitTree	vegetable	natural
champignon	mushroom	green	vegetable	natural
maïs	corn	green	vegetable	natural

sla	lettuce	green	vegetable	natural
aardappel	potato	green	vegetable	natural
ui	onion	green	vegetable	natural
fles	bottle	tool	artifact	artifact
potlood	pencil	tool	artifact	artifact
pen	pen	tool	artifact	artifact
beker	cup	tool	artifact	artifact
kom	bowl	tool	artifact	artifact
schaar	scissors	tool	artifact	artifact
ketel	kettle	tool	artifact	artifact
mes	knife	tool	artifact	artifact
schroevendraaier	screwdriver	tool	artifact	artifact
hamer	hammer	tool	artifact	artifact
lepel	spoon	tool	artifact	artifact
beitel	chisel	tool	artifact	artifact
telefoon	telephone	tool	artifact	artifact
boot	boat	vehicle	artifact	artifact
auto	car	vehicle	artifact	artifact
schip	ship	vehicle	artifact	artifact
vrachtwagen	truck	vehicle	artifact	artifact
raket	rocket	vehicle	artifact	artifact
motor	motorcycle	vehicle	artifact	artifact
helikopter	helicopter	vehicle	artifact	artifact

A.2 Abstract/concrete noun discrimination

noun	translation	$class_2$
kip	chicken	HI
arend	eagle	HI
leeuw	lion	HI
schildpad	turtle	HI
banaan	banana	HI
ui	onion	HI
aardappel	potato	HI
kom	bowl	HI
potlood	pencil	HI
telefoon	telephone	HI
vrachtwagen	truck	HI
schip	ship	HI
auto	car	HI
fles	bottle	HI
hamer	hammer	HI
jaloezie	jealousy	LO
waarheid	truth	LO
hypothese	hypothesis	LO
hoop	hope	LO
genade	mercy	LO
mysterie	mystery	LO
dankbaarheid	gratitude	LO
concept	concept	LO
verleiding	temptation	LO
trots	pride	LO
geloof	belief	LO
inzicht	insight	LO
wijsheid	wisdom	LO
geluk	luck	LO
afleiding	distraction	LO

List of Abbreviations

CBC	clustering by committee
CGN	Corpus Gesproken Nederlands
JS	Jensen-Shannon
KL	Kullback-Leibler
LSA	latent semantic analysis
MWE	multi-word expression
NMF	non-negative matrix factorization
NTF	non-negative tensor factorization
PARAFAC	parallel factor analysis
PCA	principal component analysis
PLSA	probabilistic latent semantic analysis
PMI	pointwise mutual information
SVD	singular value decomposition
TWNC	Twente Nieuws Corpus

Bibliography

[Abe and Li1996] Abe, Naoke and Hang Li. 1996. Learning word association norms using tree cut pair models. In *Proceedings of the Thirteenth International Conference on Machine Learning*.

[Bader and Kolda2006a] Bader, Brett W. and Tamara G. Kolda. 2006a. Algorithm 862: Matlab tensor classes for fast algorithm prototyping, acm transactions on mathematical software. 32(4), December.

[Bader and Kolda2006b] Bader, Brett W. and Tamara G. Kolda. 2006b. Efficient MATLAB computations with sparse and factored tensors. Technical Report SAND2006-7592, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, December.

[Bader and Kolda2009] Bader, Brett W. and Tamara G. Kolda. 2009. Matlab tensor toolbox version 2.3. `http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/`, July.

[Baldwin et al.2003] Baldwin, T., C. Bannard, T. Tanaka, and D. Widdows. 2003. An Empirical Model of Multiword Expressions Decomposability. In *Proc. of the ACL-2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, pages 89–96, Sapporo, Japan.

[Baldwin et al.2006] Baldwin, T., J. Beavers, L. van der Beek, F. Bond, D. Flickinger, and I.A. Sag, 2006. *In search of a systematic treatment of Determinerless PPs*, pages 163–180. Computational Linguistics Dimensions of Syntax and Semantics of Prepositions. Kluwer Academic.

[Baldwin2006] Baldwin, Tim. 2006. Compositionality and Multiword Expressions: Six of One, Half a Dozen of the Other? Invited talk given at the COLING/ACL'06 Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties, July.

[Basili et al.2007] Basili, Roberto, Diego De Cao, Paolo Marocco, and Marco Pennacchiotti. 2007. Learning selectional preferences for entailment or paraphrasing rules. In *Proceedings of RANLP 2007*, Borovets, Bulgaria.

[Bentivogli et al.2004] Bentivogli, Luisa, Pamela Forner, Bernardo Magnini, and Emanuele Pianta. 2004. Revising wordnet domains hierarchy: Semantics, coverage, and balancing. In *Proceedings of COLING 2004 Workshop on Multilingual Linguistic Resources*, pages 101–108, Geneva, Switzerland.

[Berry1992] Berry, Michael. 1992. Large scale singular value computations. *International Journal of Supercomputer Applications*, 6(3):13–49.

[Bhagat, Pantel, and Hovy2007] Bhagat, Rahul, Patrick Pantel, and Eduard Hovy. 2007. Ledir: An unsupervised algorithm for learning directionality of inference rules. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-07)*, pages 161–170, Prague, Czech Republic.

[Bloomfield1933] Bloomfield, Leonard. 1933. *Language*. Henry Holt, New York.

[Bouma, van Noord, and Malouf2001] Bouma, Gosse, Gertjan van Noord, and Robert Malouf. 2001. Alpino: Wide-coverage computational analysis of dutch. In Walter Daelemans, Khalil Sima'an, J. Veenstra, and J. Zavrel, editors, *Computational Linguistics in the Netherlands 2000*. Rodopi, Amsterdam, New York, pages 45–59.

[Bro and Jong1997] Bro, R. and S. De Jong. 1997. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11:393–401.

[Budanitsky and Hirst2006] Budanitsky, Alexander and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

[Bullinaria and Levy2007] Bullinaria, John A. and Joseph P. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526.

[Carroll and Chang1970] Carroll, J. D. and J.-J. Chang. 1970. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35:283–319.

[Church and Hanks1990] Church, Kenneth Ward and Patrick Hanks. 1990. Word association norms, mutual information & lexicography. *Computational Linguistics*, 16(1):22–29.

[Clark and Weir2001] Clark, Stephen and David Weir. 2001. Class-based probability estimation using a semantic hierarchy. In *Proceedings of NAACL 2001*, Pittsburgh, USA.

[Cohen1960] Cohen, Jacob. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46.

[De Lathauwer, Moor, and Vandewalle2000] De Lathauwer, Lieven, Bart De Moor, and Joos Vandewalle. 2000. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278.

[Deprettere1988] Deprettere, F., editor. 1988. *SVD and signal processing: algorithms, applications and architectures*. North-Holland Publishing Co., Amsterdam, The Netherlands.

[Erk2007] Erk, Katrin. 2007. A simple, similarity-based model for selectional preferences. In *Proceedings of ACL 2007*, Prague, Czech Republic.

[Fazly and Stevenson2006] Fazly, A. and S. Stevenson. 2006. Automatically constructing a lexicon of verb phrase idiomatic combinations. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, Trento, Italy.

[Fellbaum1998] Fellbaum, C., editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Massachusetts.

[Harris1985] Harris, Z. 1985. Distributional structure. In Jerrold J. Katz, editor, *The Philosophy of Linguistics*. Oxford University Press, pages 26–47.

[Harris1954] Harris, Zellig S. 1954. Distributional structure. *Word*, 10(23):146–162.

[Harshman1970] Harshman, R.A. 1970. Foundations of the parafac procedure: models and conditions for an "explanatory" multi-mode factor analysis. In *UCLA Working Papers in Phonetics*, volume 16, pages 1–84, Los Angeles. University of California.

[Hoekstra et al.2001] Hoekstra, Heleen, Michael Moortgat, Bram Renmans, Ineke Schuurman, and Ton van der Wouden. 2001. Syntactic annotation for the spoken dutch corpus project (cgn). In Walter Daelemans, Khalil Sima'an, J. Veenstra, and J. Zavrel, editors, *Computational Linguistics in the Netherlands 2000*, pages 73–87, Amsterdam, New York. Rodopi.

[Hofmann1999] Hofmann, Thomas. 1999. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm.

[Horak, Vossen, and Rambousek2008] Horak, A., P. Vossen, and A. Rambousek. 2008. The development of a complex-structured lexicon based on wordnet. In *Proceedings of the Fourth International GlobalWordNet Conference – GWC 2008*, Szeged, Hungary.

[Karypis2003] Karypis, George. 2003. CLUTO - a clustering toolkit. Technical Report #02-017, nov.

[Katz and Giesbrecht2006] Katz, G. and E. Giesbrecht. 2006. Automatic identification of non-compositional multi-word expressions using Latent Semantic Analysis. In *Proc. of the COLING/ACL'06 Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 12–19, Sydney, Australia.

[Kiers2000] Kiers, H.A.L. 2000. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, 14:105–122.

[Kiers and van Mechelen2001] Kiers, H.A.L and I. van Mechelen. 2001. Three-way component analysis: Principles and illustrative application. *Psychological Methods*, 6:84–110.

[Kolda and Bader2006] Kolda, Tamara and Brett Bader. 2006. The TOPHITS model for higher-order web link analysis. In *Workshop on Link Analysis, Counterterrorism and Security*.

[Kolda and Bader2009] Kolda, Tamara G. and Brett W. Bader. 2009. Tensor decompositions and applications. *SIAM Review*, 51(3), September. In press.

[Landauer, Foltz, and Laham1998] Landauer, Thomas, Peter Foltz, and Darrell Laham. 1998. An Introduction to Latent Semantic Analysis. *Discourse Processes*, 25:295–284.

[Landauer and Dumais1997] Landauer, Thomas K. and Susan T. Dumais. 1997. A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychology Review*, 104:211–240.

[Lawson and Hanson1974] Lawson, C.L. and B.J. Hanson. 1974. *Solving Least Squares Problems*. Prentice-Hall, Englewood Cliffs, NJ.

[Leacock and Chodorow1998] Leacock, Claudia and Martin Chodorow. 1998. Combining local context with wordnet similarity for word sense identification. In Christiane Fellbaum, editor, *WordNet: An Electronic Database*. MIT press, Cambridge, MA, pages 265–283.

[Lee and Seung1999] Lee, Daniel D. and H. Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791.

[Lee and Seung2000] Lee, Daniel D. and H. Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562.

[Lin1998a] Lin, D. 1998a. Automatic retrieval and clustering of similar words. In *Proceedings of* COLING/ACL *98*, Montreal, Canada.

[Lin1998b] Lin, Dekang. 1998b. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann.

[Lin1999] Lin, Dekang. 1999. Automatic identification of non-compositional phrases. In *Proceedings of ACL-99*, pages 317–324. University of Maryland.

[Lowe2001] Lowe, Will. 2001. Towards a theory of semantic space. In *Proceedings of the 23rd Annual Conference of the Cognitive Science Society*, pages 576–581, Edinburgh, UK.

[MacQueen1967] MacQueen, J. B. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley. University of California Press.

[Magnini and Cavagli2000] Magnini, Bernardo and Gabriela Cavagli. 2000. Integrating subject field codes into wordnet. In *Proceedings of LREC-2000, Second International Conference on Language Resources and Evaluation*, pages 1413–1418, Athens, Greece.

[Maks, Martin, and de Meerseman1999] Maks, I., W. Martin, and H. de Meerseman, 1999. *Referentie Bestand Nederlands. Manual*.

[Manning and Schütze2000] Manning, Christopher and Hinrich Schütze. 2000. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachussets.

[Martin and Maks2005] Martin, W. and I. Maks, 2005. *Referentie Bestand Nederlands. Documentatie*, April.

[McCarthy, Keller, and Carroll2003] McCarthy, Diana, Bill Keller, and John Carroll. 2003. Detecting a Continuum of Compositionality in Phrasal Verbs. In *Proc. of the ACL-2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, Sapporo, Japan.

[Nerbonne and Van de Cruys2009] Nerbonne, John and Tim Van de Cruys. 2009. Detecting aspectual relations quantitatively. In Erhard Hinrichs and John Nerbonne, editors, *Theory and Evidence in Semantics(Proceedings of a Symposium in honor of David Dowty)*, pages 159–182, Stanford. CSLI.

[van Noord2006] van Noord, Gertjan. 2006. At Last Parsing Is Now Operational. In Piet Mertens, Cedrick Fairon, Anne Dister, and Patrick Watrin, editors, *TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*, pages 20–42, Leuven.

[Ordelman2002] Ordelman, R.J.F. 2002. Twente Nieuws Corpus (TwNC), August. Parlevink Language Techonology Group. University of Twente.

[Pado and Lapata2007] Pado, Sebastian and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

[Pantel and Lin2002] Pantel, Patrick and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of ACM Conference on Knowledge Discovery and Data Mining (KDD-02)*, pages 613–619, Edmonton, Canada.

[Pearce2001] Pearce, Darren. 2001. Synonymy in collocation extraction. In *WordNet and Other lexical resources: applications, extensions & customizations (NAACL 2001)*, pages 41–46, Pittsburgh. Carnegie Mellon University.

[Pearson1901] Pearson, K. 1901. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572.

[Pereira, Tishby, and Lee1993] Pereira, Fernando, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of English words. In *31st Annual Meeting of the ACL*, pages 183–190.

[Piao et al.2006] Piao, S., P. Rayson, O. Mudraya, A. Wilson, and R. Garside. 2006. Measuring MWE compositionality using semantic annotation. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 2–11, Sydney, Australia. Association for Computational Linguistics.

[van der Plas and Bouma2005] van der Plas, Lonneke and Gosse Bouma. 2005. Syntactic contexts for finding semantically similar words. *Computational Linguistics in the Netherlands 2004. Selected Papers from the Fifteenth CLIN Meeting*, pages 173–184.

[Resnik1993] Resnik, Philip. 1993. *Selection and Information: A Class-Based Approach to Lexical Relationships*. PhD Thesis, University of Pennsylvania.

[Resnik1995] Resnik, Philip. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, Montreal, Canada.

[Resnik1996] Resnik, Philip. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, 61:127–159.

[Resnik1999] Resnik, Philip. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130.

[Rooth et al.1999] Rooth, Mats, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via em-based clustering. In *37th Annual Meeting of the ACL*.

[Sag et al.2002] Sag, Ivan, T. Baldwin, F. Bond, A. Copestake, and D. Flickinger. 2002. Multiword Expressions: a pain in the neck for NLP. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics*, pages 1–15, Mexico City, Mexico.

[Salton, Wong, and Yang1975] Salton, Gerard, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

[Saussure1916] Saussure, Ferdinand De. 1916. Cours de Linguistique Générale. In Charles Bally, Albert Sechehaye, and Albert Riedlinger, editors, . Payot, Lausanne and Paris.

[Schütze1998] Schütze, Hinrich. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

[Shashua and Hazan2005] Shashua, Amnon and Tamir Hazan. 2005. Non-negative tensor factorization with applications to statistics and computer vision. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 792–799, New York, NY, USA. ACM.

[Tucker1966] Tucker, L.R. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311.

[Turk and Pentland1991] Turk, Matthew and Alex Pentland. 1991. Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3(1):71–86.

[Turney2007] Turney, Peter D. 2007. Empirical evaluation of four tensor decomposition algorithms. Technical Report ERB-1152, National Research Council, Institute for Information Technology.

[Van de Cruys2006a] Van de Cruys, Tim. 2006a. The application of singular value decomposition to dutch noun-adjective matrices. In Piet Mertens et al., editors, *Verbum Ex Machina. Actes de RECITAL (TALN)*, pages 767–772, Leuven University Press, Leuven, Belgium.

[Van de Cruys2006b] Van de Cruys, Tim. 2006b. Semantic clustering in dutch. In Khalil Sima'an et al., editors, *Proceedings of the Sixteenth Computational Linguistics in the Netherlands (CLIN)*, pages 17–32, University of Amsterdam, Amsterdam, The Netherlands.

[Van de Cruys2007] Van de Cruys, Tim. 2007. Exploring three way contexts for word sense discrimination. In Marco Baroni, Allesandro Lenci, and Magnus Sahlgren, editors, *Proceedings of the Workshop on Contextual Information in Semantic Space Models: Beyond Words and Documents (COSMO)*, pages 33–41, Roskilde, Denmark.

[Van de Cruys2008a] Van de Cruys, Tim. 2008a. A comparison of bag of words and syntax-based approaches for word categorization. In Marco Baroni, Stefan Evert, and Allesandro Lenci, editors, *Proceedings of the Lexical Semantics Workshop: Bridging the gap between semantic theory and computational simulations*, pages 47–54, ESSLLI 2008, Hamburg, Germany.

[Van de Cruys2008b] Van de Cruys, Tim. 2008b. A simple extension of non-negative matrix factorization for three-way data. In *Workshop on Mining Multidimensional Data (MMD '08) at ECML PKDD 2008*, Antwerp, Belgium.

[Van de Cruys2008c] Van de Cruys, Tim. 2008c. Using three way data for word sense discrimination. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 929–936, Manchester, United Kingdom.

[Van de Cruys2009] Van de Cruys, Tim. 2009. A non-negative tensor factorization model for selectional preference induction. In *Proceedings of the workshop on Geometric Models for Natural Language Semantics (GEMS)*, pages 83–90, Athens, Greece.

[Van de CruysTo appear] Van de Cruys, Tim. To appear. A non-negative tensor factorization model for selectional preference induction. *Journal of Natural Language Engineering. Special Issue on Geometrical Models of Natural Language Semantics.*

[Van de Cruys and Villada Moirón2007a] Van de Cruys, Tim and Begoña Villada Moirón. 2007a. Lexico-semantic multiword expression extraction. In Peter Dirix et al., editor, *Proceedings of the 17th Meeting of Computational Linguistics in the Netherlands (CLIN)*, pages 175–190, University of Leuven, Leuven, Belgium.

[Van de Cruys and Villada Moirón2007b] Van de Cruys, Tim and Begoña Villada Moirón. 2007b. Semantics-based multiword expression extraction. In *Proceedings of the Workshop on A Broader Perspective on Multiword Expressions*, pages 25–32, ACL, Prague, Czech Republic.

[Vasilescu and Terzopoulos2002] Vasilescu, M. Alex O. and Demetri Terzopoulos. 2002. Multilinear analysis of image ensembles: Tensorfaces. In *ECCV*, pages 447–460.

[Venkatapathy and Joshi2005] Venkatapathy, S. and A. Joshi. 2005. Measuring the relative compositionality of verb-noun collocations by integrating features. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 899–906, Vancouver.

[Villada Moirón and Tiedemann2006] Villada Moirón, Begoña and Jörg Tiedemann. 2006. Identifying idiomatic expressions using automatic word-alignment. In *Proceedings of the EACL 2006 Workshop on Multi-word-expressions in a multilingual context"*, pages 33–40, Trento, Italy.

[Vossen1998] Vossen, P., editor. 1998. *EuroWordNet: a multilingual database with lexical semantic networks for European Languages*. Kluwer, Dordrecht.

[Wall, Rechtsteiner, and Rocha2003] Wall, Michael E., Andreas Rechtsteiner, and Luis M. Rocha, 2003. *Singular Value Decomposition and Principal Component Analysis*, chapter 5, pages 91–109. Kluwel, Norwell, MA, Mar.

[Weeds2003] Weeds, Julie. 2003. *Measures and Applications of Lexical Distributional Similarity*. PhD Thesis, University of Sussex.

[Welling and Weber2001] Welling, M. and M. Weber. 2001. Positive tensor factorization. *Pattern Recognition Letters*, 22:1255–1261.

[Wittgenstein1953] Wittgenstein, Ludwig. 1953. *Philosophical Investigations. Philosophische Untersuchungen*. Basil Blackwell, Oxford. Translated by G.E.M. Anscombe.

[Wu and Palmer1994] Wu, Zhibiao and Martha Palmer. 1994. Verb semantics and lexical selection. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 133–138, New Mexico State University, Las Cruces, New Mexico.

[Zhao and Karypis2001] Zhao, Ying and George Karypis. 2001. Criterion functions for document clustering: Experiments and analysis. Technical Report #01-40.