Compositional Distributional Semantics: Present Participle and Relative Clause Structures in Dutch

Bachelor Kunstmatige Intelligentie UU

Céline de Jong

Thesis supervisor: Michael Moortgat
Second reader: Ben Rin

7,5 ECTS

2017

Abstract

In this essay a Compositional Distributional Semantic approach is researched for studying two separate phenomena in the Dutch language. Firstly a theoretical outline for the Compositional Distributional Semantics is drawn, starting with the beginning of Distributional Semantics and ending with the current research in the expanded Compositional Distributional Semantics. Then a case study focussing on the two phenomena is outlined, performed and discussed, all while taking into account the previously described theory. The case study results show that more research into the field is necessary.

Contents

1 Introduction

In this paper I will discuss the use of Compositional Distributional semantics on Dutch language. Compositional Distributional semantics is an increasingly popular subject in the field of Computer Science and Artificial Intelligence. Within the bachelor of KI it is mainly connected to the linguistic field and in particular the Logical Grammars course. This extension of Distributional semantics crucially increases the capability of the approach when it comes to understanding natural language, which makes intelligent uses of natural language in computer systems more successful. The field is still very much open for more research, not only because it has mainly been used on the English language, but also because so far there has not been one approach that has given all the desired results, which is often caused by a lack of sufficient data. (Marelli et al., 2014) However there are already some competent tool-kits available for practical research in the field, and in this paper I will use these on a Compositional Distributional approach towards Dutch language, focussing on two often occurring phenomena within the Dutch language, the adjectively used present participle, for example 'slapende student' (sleeping student), and the relative clause, for example 'student who sleeps' (student die slaapt). I will do this by firstly discussing the theory of Distributional Semantics, since this is the basis behind the entire essay. Then I will elaborate on the compositional extension of this theory, making it the Compositional Distributional Semantics. After the theoretical background this provides I will discuss my own case study pointed towards the two phenomena. I will do this by first sketching the outline of the case study, after which I will discuss the results of my research. I will end this essay with a conclusion and some suggestions for further research on the subject.

2 The Theory of Compositional Distributional Semantics

2.1 Distributional Semantics

Distributional Semantics is an area within linguistics that categorizes semantic similarities between linguistic items (words or morphemes) using their distributional properties taken from large databases. The main idea of distributional semantics can be found in the distributional hypothesis: "linguistic items with similar distributions have similar meanings." (Firth, 1957)

This hypothesis has certain implications, as stated by Lenci:

> If this is true, by inspecting a significant number of linguistic contexts representative of the distributional and combinatorial behavior of a given word, we may find evidence about (some of) its semantic properties. (Lenci, 2008)

Lenci explains that for this hypothesis, there are weak and strong versions, depending on whether we see this evidence for semantic properties as merely

correlations or as truly causal relations: a quantitative method for semantic analysis and lexical resource induction (weak) or a cognitive hypothesis about the form and origin of semantic representations (strong).

The idea distributional hypothesis suggests is that we can induce (aspects of the) meaning of words from texts. Harris first explained this idea in 1954, using the example of 'oculist' and 'eye-doctor'. he pointed out that "oculist and eye-doctor (...) occur in almost the same environment" and more generally that "If A and B have almost identical environments we say that they are synonyms." (Harris, 1954)

Over the years distributional semantics has gained popularity within the area of computational linguistics. Here, using the distributional approach, the meaning of words are computed from the distribution of words around it. These distributions are generally represented as vectors with values related to the counts in some way, often these vectors are based on a co-occurrence matrix.

Distributional semantics' popularity can be ascribed in some part to it's success within the research fields it has been used in. The theory in itself is rather intuitive which makes it easy to implement and use, yet there have been more negative noises around as well. Distributional semantics computes the meaning of a word based on the distribution of words around it, without looking at grammatical or contextual aspects of the surroundings of the word. Through this process we lose important information about the larger linguistic structure the word is located in. This is the main critique on distributional semantics: because it is a 'bag of words'-model, it is often unsuccessful when it comes to characterizing the semantics of entire sentences or phrases. To solve this, there has been a recent rise in the interest towards compositional distributional semantics.

2.2 Compositional Distributional Semantics

To explain the theory of compositional distributional semantics I will discuss the overview article by Stephen Clark et al. (Clark et al., 2016) This article discusses important research and findings in the area of compositional distributional semantics from several researchers.

Previous attempts at the integration of formal and distributional semantics start with the logic of formal semantics and fill the gaps with distributional semantics. However Clark et al. themselves start with the framework of distributional semantics (which is vector-based, as we saw in the previous section) and uses operations that come natural to vectors to take care of the compositional processes combining words into larger phrases and sentences. These processes are for example compositionality, quantification, negation, conjunction and inference. (Clark et al., 2016, p.1-2)

There are some works using simple operators on vectors, such as addition and element-wise multiplication. These operations do not take in account word order, because they are commutative. Yet there are many works about how these operations can still be used for phrasal composition.(Clark et al., 2016, p.3) One of these is the work of Mitchell and Lapata from 2008 and 2010 (Mitchell and Lapata, 2010) (Mitchell and Lapata, 2008). Their addition function is somewhat more elaborate and has the form $p = \alpha u + \beta v$, where α and β are weights prescribed to the word vectors u and v. This allows for some syntax to be taken account of: "for example, if u is the vector representation for an adjective, and v the vector representation for a noun, v may be given a higher weight since it provides the linguistic head for the resulting noun phrase; whereas if u is an intransitive verb and v a noun, u may be weighted more highly." (Clark et al., 2016, p.4)

Mitchell and Lapata's multiplicative model uses element-wise multiplication on vectors. These multiplicative models have previously performed well on various phrasal composition tasks, despite the fact that these models are intersective, so that any vector p only has a non-zero component at index i, p_i, under multiplication when the values that were multiplied to create p_i are both non-zero. Addition does not have this aspect, which means that "vector addition emphasizes components common to u and v" (Clark et al., 2016, p.4) (where u and v are the u and v from their addition function).

Despite the fact their simplicity, these methods have worked surprisingly well in many occasions. Despite the "theoretical reasons to believe that the simple nature of vector addition [...] could not hope to capture the subtleties of natural language semantics" (Clark et al., 2016, p.5), it is difficult to create datasets and tasks "in which vector addition does not provide an extremely competitive baseline." (Clark et al., 2016, p.5)

Apart from the works using the simpler operators, there have also been works using "neural networks architectures which introduce the notion of matrices acting as operators on pairs of vectors given as input." (Clark et al., 2016, p.3) For example, Socher et al. (Socher et al., 2010) use recursive neural networks for parsing. To make this recursion possible in their neural networks, "the output of the composition operation is a vector in the same space as the input vectors". (Clark et al., 2016, p.6) This makes it easier to compare words and phrases with different syntactic types, however, the neural network loses some of it's flexibility when it comes to the composition operator, which means that all word pairs, whatever their syntactic combination, are combined using the same matrix.

This model has been extended by parametrising the combination matrix by the type of combination to solve this flexibility problem (Hermann and Blunsom, 2013), which has thereafter again been extended by introducing separate matrix operators for every word and phrase. (Socher et al., 2012) Because every word and phrase have both a matrix and a vector representation, they introduce 'matrix-vector spaces': "The matrix operators can be thought of as capturing

how words or phrases affect the meanings of other words or phrases, whereas the vectors are the meanings of the words or phrases themselves." (Clark et al., 2016, p.7) The task handled by Socher et al. is "not syntactic parsing but predicting the sentiment of adverb-adjective pairs" (Clark et al., 2016, p.8), so that their findings can be used on new data and give a prediction of their meanings.

The research area of compositional distributional semantics gained popularity after "the perceived failure of distributed models" (Clark et al., 2016, p.8), which was also discussed in the previous section on distributional semantics. Clark et al. present an extension to the discussed approaches, which should be a better performing model on compositional semantics. This extension introduces the use of multi-linear algebra to represent words with more complex syntactic types. This extension makes it possible for semantic word functions to have more than one argument. For this extension, they use the compositional framework from Coecke at al. (Coecke et al., 2010). In this framework, different interpretations of distributionality are possible, not only the classical distributional where vectors are readily interpretable because their values depict the word counts found in certain data immediately. The models also do not have to be contextual and the distributional vector spaces may be modified.

Clark et al. set up a tensor-based use of the Compositional framework from Coecke et al. discussed in the previous section, where they use CCG, Combinatory Categorial Grammar, to represent the grammatical structures they will discuss. (Steedman, 2000) The set-up of this framework mainly includes discussing some of the operations on grammatical structures that are possible within this structure. These operations are exactly the kind of operations that make this approach to semantics more complete than a merely distributional approach, because with these operations we can take account of the contextual and grammatical environments words are in.
Next to these operations, the tensor-based semantics ask for a translation of the CCG types. The syntactic types provided by the CCG are translated into semantic types using a tensor operator, to represent the fact that these semantic types are, just like the syntactic types, functions. This translation means "replace all slash operators in the syntactic types with tensor product operators". So for example, translating the syntactic type for a transitive verb in CCG to a tensor-based semantic type gives us the tensor $S \otimes N \otimes N$. To relate this to the matrices discussed in the previous section, a transitive verb would be a particular 3rd order tensor, in the tensor product space $S \otimes N \otimes N$. Here, the function $S \otimes N \otimes N$, can be thought of as taking a vector in N to the right, followed by another vector in N, returning a vector in S.

In table 1 the translations for the types in CCG grammar to semantic types, so vector-spaces in the tensor-based framework are given. The translations for the CCG rules to tensor-based rules are given below.

Syntax	Semantic
h(NP)	N
h(S)	S
h(A/B)	h(A) $\otimes$ h(B)
h(B\A)	h(B) $\otimes$ h(A)

Table 1: Syntax to semantic interpretation

One of the operations on grammatical structures in their framework is composition. Here, composition is the operation of putting words together in such a way that together they become a meaningful whole. In their article they use the example of 'red car', where the composition of 'red' and 'car', leads to the coming together meaningfully of 'red car'. The way their proposed framework takes care of the composition operation is through matrix multiplication. The usefulness of this way of composition is explained through the 'red car' example. In CCG the syntactic type of an adjective in English is N/N; "an adjective requires a noun as an argument to the right, and once it has found such a noun, will return another noun as the result." (Clark et al., 2016, p.10) This syntactic type is a function, and here Clark et al. make a link towards functions in linear algebra, in which linear maps are represented as matrices. This link was previously made by Baroni and Zamparelli (Baroni and Zamparelli, 2010), proposing to represent meanings of adjectives as matrices. Using this idea, their composition operation multiplies the context vector for the noun with the matrix for the adjective, creating the vector for the result noun. These matrices for adjectives should be learned from large enough data through linear regression techniques, which makes it possible for the generalisation of adjective-noun combinations to appear, so that even combinations not previously seen in training data can be meaningfully interpreted.

Next to composition, another important CCG-rule for the tensor-based framework is application. In CCG, there is forward and backward application.

Forward application:
$$A/B \otimes B \rightarrow A$$
Backward application:
$$B \otimes B\backslash A \rightarrow A$$

These two rules are translated into tensor-contraction, which is the operation used in the tensor-based semantics. In tensor-contraction, the Einstein summation notational convention is used. This convention simplifies equations because it 'implicitly assumes summations over the relevant range on every component index that occurs twice'. (Clark et al., 2016, p.13) This convention does not change anything in the actual operations, only in the notation of the operations. Through summation according to this convention the tensor-contraction operation then reduces the tensor-rank by 2. The translations for application into tensor-contraction work the same for both forward an backward application,

which results in, for example for an intransitive verb, the following transformation:

$$\frac{\text{NP} \qquad \text{NP}\backslash\text{S}}{\text{S}}$$

becomes:

$$\frac{\text{N} \qquad \text{N}\otimes\text{S}}{\text{S}}$$

If this phrase containing an intransitive verb would for example be 'student sleeps' (student slaapt), the Einstein summation convention's workings can be made visual as such:

$$\sum_i \text{Student}_i \ \text{sleeps}_{ij} = \text{Student}_i \ \text{sleeps}_{ij} = \text{Student sleeps}_j$$

> The framework we have described derives a sentence vector for any syntactic derivation resulting in S, including those which use the additional combinatory rules of CCG, providing a complete recipe for the meaning composition process for any sentence. (Clark et al., 2016, p.19)

2.2.1 Composition operations

To show that this is not the only possible approach to calculating the vectors, the four possible ways of computing are discussed below. The first is a simple additive model, where the vector of a phrase is found by summing up all the vectors of the words it contains. (Paperno et al., 2014) The second is the also simple multiplicative model, where the vector for phrases are produced by component-wise multiplication of the vectors that belong to the words that the phrase contains, like in figure 5. In Paperno et al., two more complicated composition-models are described, the first being lexical function:

> For the lf (lexical function) model, we construct functional matrix representations of adjectives, determiners and intransitive verbs. These are trained using Ridge regression with generalized cross validation from corpus-extracted vectors of nouns, as input, and phrases including those nouns as output. (Paperno et al., 2014, p.6)

The second of these more complicated models is introduced in this paper by Paperno et al. and is called the practical lexical function. This model differs in practice from the lexical function model by also building proposition matrices and preparing separate subject and object matrices. They introduce this model because the lexical function model has some known issues where, if we do not want to lose any useful information embedded in the data, tensor dimensions can get impractically high. They resolve this issue by designing their model so

that 'a functional word is not represented by a single tensor of arity-dependent order, but by a vector plus an ordered set of matrices, with one matrix for each argument the function takes.' (Paperno et al., 2014, p.7) These last two models use regression learning to find vectors and matrices, which means they need to be trained and, afterwards, tested, in order to be used. They are also based on the previously discussed tensor-contraction operation by Clark et al.

These more difficult models are more elaborate than the simpler models, however they have the obvious downside that training and testing takes generally more time than simple addition or multiplication, which can be a good reason to choose for the simpler models if the data allows it. On top of this downside, there is also the fact that the models that use regression learning do not automatically always perform better than the simple models. Therefore, it is not always the natural choice to use the more elaborate models for composition.

2.3 Lambek Calculus

In the Clark et al. article, the CCG grammar is used. For my essay, I will be using their theory and framework, however, I will do this using the Lambek Calculus. The Lambek Calculus is a logical framework, therefore in stead of rules, like in CCG, it is based on theorems. The theorems for composition and combination work similarly to the rules of the CCG grammar. (Moot and Retoré, 2012) The types for words are also the same, however the notation of the composition through the theorems is different. The inference rules for Lambek calculus are shown below:

Axiom

$$\frac{}{A \vdash A}\ (\mathrm{Id})$$

Theorems

$$\frac{A \vdash B \qquad C \vdash D}{A \otimes C \vdash B \otimes D}\ (\mathrm{Mon}\otimes)$$

$$\frac{A \vdash B \qquad C \vdash D}{B \backslash C \vdash A \backslash D}\ (\mathrm{Mon}\backslash)$$

$$\frac{A \vdash B \qquad C \vdash D}{C / B \vdash D / A}\ (\mathrm{Mon}/)$$

$$\frac{B \vdash A \backslash C}{A \otimes B \vdash C}\ (\mathrm{Res}_{\backslash\otimes})$$

$$\frac{A \otimes B \vdash C}{B \vdash A \backslash C}\ (\mathrm{Res}_{\otimes\backslash})$$

$$\frac{\mathrm{A} \vdash \mathrm{C/B}}{\mathrm{A} \otimes \mathrm{B} \vdash \mathrm{C}} \, (\mathrm{Res}_{/\otimes})$$

$$\frac{\mathrm{A} \otimes \mathrm{B} \vdash \mathrm{C}}{\mathrm{A} \vdash \mathrm{C/B}} \, (\mathrm{Res}_{\otimes/})$$

(Moortgat and Oehrle, 1999)

The two phenomena I will be looking at in the case study are those of the adjectively used present participle and the relative clause. Examples for the respective phenomena are 'sleeping student' (slapende student) and 'student who sleeps' (student die slaapt). Because the rest of the essay will focus on these two types of phrases, I will give their derivation in Lambek Calculus as examples for how the theorems in Lambek Calculus work below.

The deduction tree for the phrase 'sleeping student', where 'sleeping' has the type np/np and 'student' has the type np:

$$\dfrac{\dfrac{\overline{np_0 \longrightarrow np_3} \quad \overline{np_2 \longrightarrow np_1}}{np_0/np_1 \longrightarrow np_3/np_2} \, (\mathrm{Mon}/)}{(np_0/np_1) \otimes np_2 \longrightarrow np_3} \, (\mathrm{Res}_{/\otimes})$$

The deduction tree for the phrase 'student that sleeps', 'student' has the type np, 'that' has the type $(np/np)/(np\backslash s)$ and 'sleeps' has the type $np\backslash s$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{np_0 \longrightarrow np_1} \quad \overline{np_2 \longrightarrow np_7}}{np_1\backslash np_2 \longrightarrow np_0\backslash np_7} \, (\mathrm{Mon}\backslash) \quad \dfrac{\overline{np_3 \longrightarrow np_5} \quad \overline{s_6 \longrightarrow s_4}}{np_5\backslash s_6 \longrightarrow np_3\backslash s_4} \, (\mathrm{Mon}\backslash)}{(np_1\backslash np_2)/(np_3\backslash s_4) \longrightarrow (np_0\backslash np_7)/(np_5\backslash s_6)} \, (\mathrm{Mon}/)}{((np_1\backslash np_2)/(np_3\backslash s_4)) \otimes (np_5\backslash s_6) \longrightarrow np_0\backslash np_7} \, (\mathrm{Res}_{/\otimes})}{np_0 \otimes (((np_1\backslash np_2)/(np_3\backslash s_4)) \otimes (np_5\backslash s_6)) \longrightarrow np_7} \, (\mathrm{Res}_{\backslash\otimes}))$$

The Lambek calculus indeed allows us to make a sound deduction for phrases that should indeed reduce to a noun-phrase, a NP-type, as the above examples should convince us to believe. The above examples also contain indices for the types. These indices are ascribed to each unique type and can be followed through the derivation to later be used in the tensor-contraction, since these indices are used in the Einstein summation convention.

In the previous section, the tensor-based semantics from Clark et al. was discussed. The translation for this tensor-based semantics works similarly for the Lambek Calculus types as the CCG types. For example, a verb, $np\backslash s$, results in the translated function $np \otimes s$, a function that takes a vector in np to the left and returns a vector in s. This works the same for the other types that consist of multiple atomic types and are therefore translatable into these functions.

Two of the combinatory functions for the tensor-based semantics are application and composition. These are similar to the CCG rules of function application and composition, however here I will not show them in relation to the CCG-grammar. In the tensor-based semantics, both of these functions reduce to tensor-contraction, an example for which is given below, with the natural language words combined with their tensor-based types:

$$\frac{\text{Student}_{np} \qquad \text{sleeps}_{(np \otimes s)}}{\text{Student sleeps}_{s}} \text{ (application)}$$

As shown in the example, tensor contraction reduces the amount of tensors in the resulting type. This can be done as well for larger sentences than this very simple example, following the same rules. As well as being used for larger sentences, the tensor-contraction operation can also be used as the translated operation for other operations in CCG and Lambek calculus.

3 Case Study: Present Participle and Relative Clauses in Dutch

3.1 Outline

The above described theory of the combination of distributional and compositional semantics has been researched and put into practical use by an increasing amount of researchers the past years, however most of these instances have been studying the effectiveness of the theory on the English language. The point of this essay is to see how effective the theory is on Dutch language, and to study this, I will be looking at one specific aspect of Dutch language; the similarity of meaning between the following two phenomena, the adjectively used present participle (bijvoeglijk gebruikt tegenwoordig deelwoord), and relative clauses containing intransitive verbs (bijvoeglijke bijzin). An example:

Slapende student (adjectively used present participle)
Student die slaapt (relative clause)

So far, the theory of compositional distributional semantics has been discussed with examples of short phrases consisting of a few words. The semantic value of these phrases can be found by combining the semantic values of the words it consists of in some manner. These semantic values will come in the form of vectors or matrices, which are based on co-occurrence counts. The combination of these semantic values can be done using several different composition models, as discussed in section 2.2.1, however in this case study I will be using the tensor-based approach by Clark et al and compare it to the simple multiplicative model.

3.1.1 Data

The data for this case study will consist of phrases containing the phenomena mentioned above. For each of the phenomena, an approach to finding the vector interpretations of the entire word combination will be needed, for this will be the end-product of the study.

For the adjectively used present participle type, this can be done rather simply by finding the vector of the words it consists of, for example 'slapende' (an adjectively used present participle, which I will treat as an adjective) and 'student' (noun), and combining these vectors through multiplication, producing the combined vector of the phrases. For the adjectives in this type, I will be using the theory that nouns are a simpler type, they will be vectors, and that adjectives will be represented by matrices, which was also briefly discussed in the section on the Clark et al. article. (Baroni and Zamparelli, 2010) This is based on the idea that words are not simple loose elements, but rather functions that take arguments to give a certain result.

For the relative clause type, the combination of the vectors might be slightly more complicated, due to the occurrence of a relative pronoun; 'die' in for example 'student die slaapt' (student that sleeps). Intuitively, the vector of 'student that sleeps' should describe the intersection of 'students' and 'things that sleep'. To find this intersection we would ideally want to use multiplication, but the relative pronoun will have a type too complex to use this simple function on due to the amount of dimensions it's matrix will have. Therefore, for this phrase type, I will be looking at the relative pronoun as a logical constant, describing a combinatory function. This will make it possible to use the tensor-contraction composition model and end up with a vector in the right space.

For this to work I need to find a hand-crafted value for the relative pronoun that will work with all the data. In these type of phrases, there is a verb, which in Lambek Calculus will be of the type $np\backslash s$ (this counts for intransitive verbs, transitive verbs are of the type $(np\backslash s)/np$, however for the sake of simplicity in this essay I will be looking only at intransitive verbs). So the verb takes an np (noun) to produce an s (sentence). We have this noun (np) on the other side of the relative pronoun. However, we still have the relative pronoun in the way, preventing the easy combination of the two.

We want the entire phrase to live in the same vector space as the noun, so in NP-space, because the type of the entire combination (noun, relative pronoun and verb) is in fact a noun phrase, an NP. However, the verb-type introduces the phrase to the S vector-space, taking the phrase out of the NP vector-space we want it to be in. Therefore, a logical constant value for the relative pronoun should take care of this problem, making the S-space disappear within the relative clause. The logical constant that will resolve this, is of the type $(np\backslash np)/(np\backslash s)$, which will leave the phrase in the NP-space, where we want it to be, so this will be the type for the word 'die'.

The semantic types in Lambek calculus for all the other word-types are shown below. The adjectively used present participle word combination consists of the following two types:
- The adjective: np/np
- The noun it modifies: np
The relative clause word combination consists of the following types:
- The verb in the subordinate clause: $np\backslash s$
- The noun it modifies: np
- The relative pronoun: $(np\backslash np)/(np\backslash s)$

The composition models give a way to combine the meaning of the words into phrase meanings. However, for this to be possible, we will first need to find the meaning of the words themselves. These meanings are represented in the vectors and matrices that result after manipulating the co-occurrence counts of the words in the phrases through learning techniques. This technique will be used to learn values for all the words in the data. (However I will not be using the result for the relative pronoun, because of the logical constant value that will be used for this.)

I will do this with the TensorFlow tool-kit, using their models to find word vectors and then using these in regression learning. This is based on the idea that the non-atomic types are in fact functions that can be learned. (Grefenstette et al., 2013) This learning means that the matrix for 'sleeping' (slapende) will be learned from the vectors that represent 'student' and 'slapende student'. The vectors representing 'student' and 'slapende student' can be extracted from the corpus, using TensorFlow's word2vec model. (Abadi et al., 2016) This is the distributional part of the case study, based on the intuition stated in section 2.1; "linguistic items with similar distributions have similar meanings." (Firth, 1957) The learning of the matrix values is compositional.

To be able to find any reliable outcome, a large database is necessary. Therefore, I will be using the tokenised, POS-tagged and lemmatized corpus of Dutch sentences called Lassy Groot. (van Noord et al., 2013) This database will be filtered for the sentences containing the mentioned phenomena using XPath queries, which will result in a new database for each of the phenomena. The used XPath queries are shown below.

Adjectively used present participle:
//node[@cat='np' and node[@wvorm='od' and @positie='prenom'] and node[@rel='hd' and @pt='n']]
Relative clause:
//node[@cat='np' and node[@rel='hd' and @pt='n'] and node[@rel='mod' and @cat='rel' and node[@rel='rhd' and @pt='vnw'] and node[@rel='body' and @cat='ssub' and node[@rel='hd' and @pt='ww']]]]

These sentences will then be the input for TensorFlow. In table 2 the sizes of the new databases are shown.

Adjectively used present participle	Relative Clause
261402	384771

Table 2: Sizes of the databases in amount of phrases

After finding these learned vectors and matrices for the data, I will be combining them using the tensor-contraction compositional models. The scripts I will be using for this finds the tensor interpretations for the words based on their deduction chain in Lambek calculus, so this takes the syntactical derivation of the phrases in account when computing. I will be using code made by M.J. Moortgat and G.J. Wijnholds (Moortgat and Wijnholds, 2017). The code makes a deduction in Lambek calculus of a phrase, keeping track of the axioms in the deduction. These axioms hold information about what types belong together. Keeping track of these axioms makes it possible to link the axioms, therefore link the types. After linking these axioms, the linked axioms can be reduced using the Einstein notation to get a much smaller result term, whereas without this axiom linking the resulting map would be much larger and less easily usable.

The code as it is written by Moortgat and Wijnholds takes an arbitrary matrix for each word to begin with, however I will use the numeric values for these matrices that I learned through regression learning. The dimensions I will be using are based on the dimensions used in another study on Compositional Distributional Semantics for Dutch (Tulkens et al., 2016). In their research they use a dimension of 160 or 320 for all types, however due to the time for and size of this research I will be using dimensions of 160 for my types. The numeric values are the results gained from the TensorFlow scripts. It also takes a lexicon containing the words and their semantic types as they are in the Lambek Calculus, which are the ones named above for each word type.

When I have found these tensor interpretations I will look at the similarity of the results between the phenomena to produce an answer to the question whether the method has given us any results worth looking further into. Intuitively, the expected outcome is that for example the 'slapende student' and 'student die slaapt' phrases both produce similar result vectors, since their meaning in natural language is similar as well.

3.2 Results

Firstly I have calculated the prediction-based vectors for each noun with TensorFlow from my data, as well as the vectors for the noun-verb and noun-adjective combinations. I calculated the vectors for these combinations by taking the words it consists of and adding them together as one word, so that the TensorFlow algorithm would view them as a whole. For example, the combination 'volgende commissie' and 'commissie die volgt' would be adapted to be respec-

tively 'volgende_commissie' and 'commissie_die_volgt'. Then I learned the matrices for the verbs and adjectives based on the co-occurrence vectors. Every atomic type has a dimension of 160, so every non-atomic type has a dimension of (160,160), because both the verb and adjective non-atomic types consist of two atomic types.

The value for the relative pronoun has to be hand-crafted as I stated in the previous section, and it cannot simply be an identity matrix. This is due to the fact that the verb lives in a $np\backslash s$ space, so an identity matrix would not be able to take care of taking away the s space from the whole. Previously, this problem has been solved by discarding the relative pronoun all together and deciding that the s space may very well be comparable to the np space. However, there have been some developments recently about how to handle this problem. These developments centre around the idea of using a Frobenius operation to let the information flow freely from the noun to the relative clause. (Sadrzadeh et al., 2013)

In a practical sense for my case study, this means that the value for the relative clause will be dependent on the verb that it contains, manipulating it's vector through the *uncopying* operation. This operation encodes vectors of higher dimensions into vectors of lower dimensions into by copying the values of the diagonal of the higher dimension vector into the lower dimension vector.

I used the Moortgat and Wijnholds code to compute the vectors for word combinations for several examples of both phenomena, based on random test sampling, to study their similarities.

The result vectors will be of a size 160, because the end-type will always be np for each of the phrase-types. The test-examples I used for my sample are the following phrases:

1. Pastoor die loopt
2. Lopende pastoor
3. Kinderen die schrijven
4. Schrijvende kinderen
5. Commissie die volgt
6. Volgende commissie

The combinations of 1 and 2, 3 and 4, 5 and 6, should have similar outcomes, since their meaning is the same in natural spoken Dutch. The vector for the noun is the same every time it is used, so for the outcome to be somewhat similar, the vectors for the adjective and the verb should already be somewhat similar. These are the vectors that have been learned from the data, and they will make the most difference in how close the results will be to each other. However, since they are learned from the existing data, their outcomes rely on

the size of the example-database they are learned from. In table 3 these sizes are shown and their differences should be taken into account when reviewing the results.

Type of combination	Amount of combinations
'..._die_loopt'	143
'lopende_...'	1329
'..._die_schrijven'	3
'schrijvende_...'	45
'..._die_volgt'	48
'volgende_...'	40448

Table 3: Sizes of example-databases

To be able to compare their similarities, I calculated the cosine spatial difference between all vectors. This calculates the angle between the vectors,so the lower this value, the more the result vectors of the phrases are alike. These results are based on the more elaborate regression learning compositional approach, however, to see whether this more elaborate approach indeed gives us more promising outcomes, I will also compare them to the results gained from the simple multiplicative model. These results I gained by simply multiplying the vectors for their parts, so for example for 'student sleeps', the multiplicative result is gained by multiplying the vectors for 'student' and 'sleeps'. The results are in table 4 below:

Sentence combination	Regression Learning	Simple multiplication
1 and 2	1.01503573096	0.815702607733
3 and 4	0.915613827026	0.766193557366
5 and 6	0.860043399952	0.761181701691

Table 4: Cosine spatial differences between the sentences, using different compositional methods

The outcomes for the sentence combinations are better when they are closer together (when their cosine spatial difference is lowest), because in natural language, the phrases mean the same. Since their numerical results are an abstract portrayal of their meaning in natural language, these should be also the same, or at least similar. So according to these results, the more elaborate model based on Clark et al.'s framework does not necessarily give us the best outcome.

This is not necessarily surprising, since it has previously been found in other research as well that the results from simple additive and multiplicative models are often hard to beat. However in this case, the better results for the multiplicative model can perhaps be somewhat explained by the relatively small example datasets used for the regression learning. As stated above, the sizes of these datasets should be taken into account when looked at the results, so keeping in mind that the training sets for the learning of the vectors were not

only sometimes small, but also in-consequent in size, the lower-scoring regression learning results can be at least partly explained.

4 Conclusion

After finding the results to my own case study I firmly believe in the need for larger case studies in the field of Compositional Distributional semantics. The tools that were within my reach were promising, comprehensible and usable with my data, but despite the theoretic outline of my case study being sound, the results show that more elaborate models do not always give better results. However, the results were also not entirely negative towards these models, considering the fact they continue to raise interesting questions that have not yet been answered.

The theory behind the case study has been more elaborated and researched within the field than the practical aspect, especially for the Dutch language. However, some of the less expanded parts of the practical aspect of Compositional Distributional semantics can be traced back towards some unanswered questions within the theoretical background of the field. The main example of these unanswered questions is the discussion around the most successful composition models, which proved to be relevant again in my own case study. There is need for more research to give a definite answer to whether the simple multiplication and addition methods are significantly superior compared to the more elaborate learning methods, and if not, why they sometimes give better results anyway. Despite the fact that it is of course possible to try each one out for every task within the compositional distributional semantics, I suggest more research focussed on when each compositional method is most useful. If there is a more conclusive answer, we might be able to decide which method to use depending on the kind of data is being studied.

Within the practical part of the field, there is a need for larger studies on larger databases, for multiple languages apart from English and using multiple tools to see how their performances may differ. Currently there are different tools for Compositional Distributional Semantic studies, however there seems to not yet be a clear way to distinguish which tools are most successful for which studies.

Research in the field of Compositional Distributional Semantics has so far provided some promising results for how useful it could become for the natural language programming task within Artificial Intelligence. Through more elaborate and expansive research, the field should prove very successful not only for a better understanding of the computability of the English language, but for all intelligent uses of all languages in computer systems.

5 References

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I. J., Harp, A., Irving, G., Isard, M., Jia, Y., Józefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D. G., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P. A., Vanhoucke, V., Vasudevan, V., Viégas, F. B., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467.

Baroni, M. and Zamparelli, R. (2010). Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. *Proceedings of the EMNLP Conference*, pages 1183–1193.

Clark, S., Rimell, L., Polajnar, T., and Maillard, J. (2016). The categorial framework for compositional distributional semantics. *Technical Report, University of Cambridge Computer Laboratory.*

Coecke, B., Sadrzadeh, M., and Clark, S. (2010). Mathematical foundations for a compositional distributional model of meaning. *CoRR*, abs/1003.4394.

Firth, J. (1957). *A Synopsis of Linguistic Theory, 1930-1955.*

Grefenstette, E., Dinu, G., Zhang, Y. Z., Sadrzadeh, M., and Baroni, M. (2013). Multi-step regression learning for compositional distributional semantics. *arXiv preprint arXiv:1301.6939.*

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

Hermann, K. M. and Blunsom, P. (2013). The role of syntax in vector space models of compositional semantics. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria*, 1: Long Papers.

Lenci, A. (2008). Distributional semantics in linguistic and cognitive research. *Italian journal of linguistics*, 20(1):1–31.

Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S., and Zamparelli, R. (2014). Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment.

Mitchell, J. and Lapata, M. (2008). Vector-based models of semantic composition. *Proceedings of ACL*, pages 236–244.

Mitchell, J. and Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive Science*, 34:1388–1439.

Moortgat, M. and Oehrle, R. (1999). Proof nets for the grammatical base logic. *Abrusci, V.M., Casadio, C., Sandri, G. (eds)Dynamic Perspectives in Logic and Linguistics*.

Moortgat, M. and Wijnholds, G. (2017). Bewijzen en tensorrekenen. *Logische Grammatica's*, Week 9, Course material and code.

Moot, R. and Retoré, C. (2012). *The Logic of Categorial Grammars - A Deductive Account of Natural Language Syntax and Semantics*, volume 6850 of *Lecture Notes in Computer Science*. Springer.

Paperno, D., Pham, N. T., and Baroni, M. (2014). A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 90–99.

Sadrzadeh, M., Clark, S., and Coecke, B. (2013). The Frobenius anatomy of word meanings I: subject and object relative pronouns. *Journal of Logic and Computation*, 23(6):1293–1317.

Socher, R., Huval, B., Manning, C. D., and Ng, A. Y. (2012). Semantic compositionality through recursive matrix-vector spaces. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1201–1211.

Socher, R., Manning, C. D., and Ng, A. Y. (2010). Learning continuous phrase representations and syntactic parsing with recursive naural networks. *Proceedings of NIPS Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9.

Steedman, M. (2000). *The syntactic process*, volume 24. MIT Press.

Tulkens, S., Emmery, C., and Daelemans, W. (2016). Evaluating unsupervised dutch word embeddings as a linguistic resource. *CoRR*, abs/1607.00225.

van Noord, G., Bouma, G., Van Eynde, F., de Kok, D., van der Linde, J., Schuurman, I., Sang, E. T. K., and Vandeghinste, V. (2013). *Large Scale Syntactic Annotation of Written Dutch: Lassy*, pages 147–164. Springer Berlin Heidelberg, Berlin, Heidelberg.