Type-driven distributional semantics for prepositional phrase attachment

Antonin Delpeuch
Trinity College

UNIVERSITY OF CAMBRIDGE

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of Master of Philosophy in Advanced Computer Science

University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom

Email: ald50@cl.cam.ac.uk

June 11, 2015

Declaration

I Antonin Delpeuch of Trinity College, being a candidate for the M.Phil in Advanced Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 14570 [1]

Signed:

Date:

[1] `texcount -brief -inc -sum dissertation.tex`

Acknowledgments

I am extremely grateful to Stephen Clark and Tamara Polajnar who have been fantastic advisors throughout the project. The code I have written for this thesis is based on an original script by Tamara.

I thank the Computer Laboratory and Trinity College for their support to present my work at the Advances in Distributional Semantics workshop. I thank (in no particular order) the reviewers of the workshop, the ESSLLI Student Session reviewers, Daniel Fried, Kathrin Stark, Laura Rimell, Anne Preller, Mehrnoosh Sadrzadeh, Dimitri Kartsaklis, and Peter Hines for their useful comments and discussions.

This year was made possible by the exchange program between École Normale Supérieure (Paris) and Trinity College.

Abstract

Combining the strengths of distributional and logical semantics of natural language is a problem that has gained a lot of attention recently. We focus here on the distributional compositional framework of Coecke et al. (2011), which brings syntax-driven compositionality to word vectors. Using type driven grammars, they propose a method to translate the syntactic structure of any sentence to a series of algebraic operations combining the individual word meanings into a sentence representation.

My contribution to these semantics is twofold. First, I propose a new approach to tackle the dimensionality issues this model yields. One of the major hurdles to apply this composition technique to arbitrary sentences is indeed the large number of parameters to be stored and manipulated. This is due to the use of tensors, whose dimensions grow exponentially with the number of types involved in the syntax. Going back to the category-theoretical roots of the model, I show how the use of diagrams can help reduce the number of parameters, and adapt the composition operations to new sources of distributional information.

Second, I apply this framework to a concrete problem: prepositional phrase attachment. As this form of syntactic ambiguity requires semantic information to be resolved, distributional methods are a natural choice to improve disambiguation algoritms which usually consider words as discrete units. The attachment decision involves at least four different words, so it is interesting to see if the categorical composition method can be used to combine their representation into useful information to predict the correct attachment. A byproduct of this work is a new dataset with enriched annotations, allowing for a more fine-grained decision problem than the traditional PP attachment problem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Two orthogonal views on natural language semantics have evolved in parallel for about a century. The first one was born with the advent of mathematical logic, and reuses much of its methodology to formalize what humans mean by a given sentence. These logical tools are very helpful to represent the truth of a sentence, entailment relations between statements, scope of quantification and many other phenomena. However, these theories rely on symbolic representations that remain isolated from the world they are supposed to describe, making them of little practical use. On the other end of the spectrum, distributional semantics represent words by their concrete use cases, aggregated into numerical representations. These representations can carry various forms of information that can be used to discover relations between words. In turn, they lack most of the reasoning capabilities of logical semantics, for instance because they lack a notion of quantification.

These two approaches have seemingly complementary features and many models have been proposed to combine them. Among them, the distributional compositional framework of Coecke et al. (2011) brings to distributional semantics one of the most praised features of logical models: compositionality. This means that the representation of a sentence is recursively built from the representation of its words, using the syntactic structure of the sentence to guide composition operations. Their proposal originally used pre-

group grammars for the syntax, but it can be adapted to other forms of type-driven grammars such as the Combinatory Categorial Grammar of Steedman (2000). This framework is presented in the first chapter of this thesis, together with brief introductions to CCG and distributional semantics.

One downside of this framework is that, unlike other distributional compositional models (Socher et al., 2013; Cho et al., 2014), it has not been applied to arbitrary sentences yet. The main reason behind this gap is the prohibitive dimensionality of the tensors that represent each word. The second chapter explains the issue and proposes a novel way to design parameter-efficient models of meaning, while preserving the theoretical guarantees of the framework. By showing how to construct the *free compact closed category* generated by a monoidal category, we weaken the requirements on the operation used to pair the meaning of words. This allows us to avoid uses of the tensor product and opens up the framework to non-linearities in the composition process. As this proposal requires some symbolic rewriting procedures to be applied in practice, we show how an existing diagrammatic theorem prover can be reused to perform this rewriting, and present a prototype of a tool to design models of meaning based on it.

In the second part of this thesis, we apply the distributional compositional model to the semantics of prepositions in English. This closed class of words had not been covered in the distributional compositional framework before, and it is interesting to evaluate to what extent they can be represented using techniques originally designed for open-class content words such as nouns and verbs. We show how the diagrams introduced in the first part help us to design parameter-efficient tensors for prepositions, a crucial step to make their estimation tractable.

We apply our model of meaning to the prepositional phrase attachment problem, introduced in Chapter 4. This problem provides a way to evaluate the quality of the tensors learnt, and to what extent they capture useful semantic information. The fine-grained syntactic analysis CCG provides in this case leads us to model not only the attachment location of prepositional phrases, but also its type, i.e. whether it is an adjunct or an argument for the phrase

it attaches to. This information was neglected by previous approaches to the problem, but we show that it plays a significant role and can be used to tell apart the lexical and semantic interactions involved in this problem. We extract an enriched version of the standard dataset used for the evaluation of PP attachment techniques, and analyze the performance of our system on it.

Chapter 2

Background

In this chapter, we give an introduction to the field, covering the tools upon which we build our approach to dimensionality reduction and to prepositional phrase attachment. We start with a quick tutorial on Combinatory Categorial Grammar, the grammatical framework we use to describe the syntax of natural languages. Then, we introduce distributional semantics, with pointers to the most recent developments in the field. The third section combines the first two by drawing a link between type-driven syntax and distributional semantics, the core idea of the distributional compositional theory.

2.1 Combinatory Categorial Grammar

We introduce and motivate the grammatical formalism we use, Combinatory Categorial Grammar (Steedman, 2000), and the associated corpus of annotated sentences, CCGbank (Hockenmaier and Steedman, 2007).

2.1.1 Combinatory rules

The traditional model of formal languages and linguistic structures relies on syntax trees and grammars, for instance context-free ones (Chomsky, 1956).

Figure 2.1: Comparison of rule-based and type-based derivations for a simple sentence

In this Chomskyan framework, words are given atomic categories such as `DT` (determiner) or `NN` (noun) and grammars are described by rewriting rules such as `S` $\rightarrow$ `NP VP` (a noun phrase followed by a verb phrase makes a sentence) or `NP` $\rightarrow$ `DT NN` (a determiner followed by a noun makes a noun phrase). The set of rules and the lexicon describe the language modelled by the grammar.

In a type-driven framework, the categories assigned to words are more complex and encode in themselves how words can be combined together. For instance, an intransitive verb such as *walks* will be given the type $S\backslash NP$, which means that once combined with an NP on its left, this word will produce a phrase of type S, as in Figure 2.1. Instead of defining specific rules for each language, CCG relies on general type combination principles to build syntax trees. Before we present the most important ones, let us define the structure of types.

Types are generated by a few basic types (such as NP or S) and are closed under two binary operators, the forward and backward slashes. These slashes create *function* types, in the sense that they consume an input type (written on the right-hand side of the slash) and produce an output type (on the left-hand side). A phrase of type A/B can be combined with an argument B on its right hand side to produce a phrase of type A, using the forward application rule, and similarly for backward application:

$$\frac{A/B \qquad B}{A}> \qquad\qquad < \frac{B \qquad A\backslash B}{A}$$

In some cases it is also useful to have function composition for types, espe-

cially when the final argument is not immediately available. There are again two versions of such a composition rule, for both forward and backward functions:

$$\frac{A/B \qquad B/C}{A/C}\,\text{B}> \qquad\qquad <\text{B}\,\frac{B\backslash C \qquad A\backslash B}{A\backslash C}$$

The composition rules are especially useful when combined with the type raising rules, that turn an argument into a function:

$$\frac{A}{B/(B\backslash A)}\,\text{T}> \qquad\qquad <\text{T}\,\frac{A}{B\backslash(B/A)}$$

One interesting feature of type-driven grammars is that richer types provide a stronger link between the words and the role they eventually play in the whole sentence: the grammar is naturally lexicalized. From a more cognitive perspective, the theory is also relevant for language acquisition. If we assume that the typing rules we have just described are universal, they could therefore not require any learning effort and be part of the innate symbolic capabilities of the human brain. Acquiring a language would just boil down to learning a lexicon, i.e. a mapping from words to their types. Finally, as will see in Section 2.3, using a more algebraic type system than context free grammars makes it easier to translate the syntactic structure to a composition method for semantic representations.

2.1.2 CCGbank

To empirically test the theory of CCG on real-world sentences, train parsers and taggers, it is crucial to have access to a gold standard of annotated sentences.

The corpus CCGbank (Hockenmaier and Steedman, 2007) has been obtained from the Penn Treebank by converting syntax trees to CCG type derivations.

This process is not straightforward as the analysis provided in the Penn Treebank is often much less informative than the corresponding CCG derivations, for instance for the analysis of noun phrases, which is mostly left flat in the Penn Treebank. To obtain the missing information, various heuristics have been employed, introducing some noise. The treebank has later been improved by Honnibal et al. (2010) who leveraged other annotation efforts on the treebank. Their new version, called the *rebanked* CCGbank, provides a more accurate analysis of adjoints and arguments for verb phrases, and an extension of this distinction to complements of noun phrases, among others.

2.2 Distributional semantics

Distributional semantics provide one partial answer to the long-standing problem of representation of meaning in natural language. We first present the theoretical motivation behind these semantics, and then introduce the concrete techniques that are commonly used to implement it in practice.

2.2.1 Motivation

The semantics of natural language were at first dominated by logical approaches, as research in mathematical logic and philosophy of language were originally quite close. Wittgenstein (1922) was greatly influenced by this faith in formal semantics and proposed a logical theory of meaning, that he radically dismissed a decade later (Wittgenstein, 1953). He proposed instead a view on meaning that is now considered as the precursor of distributional semantics.

The goal of these alternative descriptions of meaning is to link word senses to their uses. Wittgenstein describes the meaning of a word by means of *games*[1], in terms of the effect on others incurred by the use of a word in

[1]His notion of game is actually very broad, it describes the interaction of two persons and the mutual influence they have by communicating.

a given context. He also rejects the traditional use of logical predicates to model nouns and verbs, arguing that there is no precise boundary between what is or is not a game, for instance. A more applicable version of these ideas is proposed by Firth (1951), who claims that meaning can be related to the set of contexts where a given word occurs.

Such a hypothesis makes it easy to learn semantics out of long corpora, by simply collecting all the contexts in which words occur. Of course, this theory does not specify entirely what contexts are or how they are aggregated into one *meaning*, and many different approaches have been proposed for concrete experiments. Most of them use finite-dimensional real-valued vectors to represent word meanings, and use various algebraic metrics (distances, dot products) to compare them (Deerwester et al., 1990; Mitchell and Lapata, 2008; Polajnar and Clark, 2014; Kiela and Clark, 2014).

2.2.2 Counting word occurrences in contexts

One popular way to implement this theory is to consider that the context of a word occurrence is the enclosing sentence, and associate a vector with that context.

Concretely, we choose a set of N context words and associate each of them with a basis vector in an N-dimensional vector space. These context words are usually frequent content words, so that their presence in a context gives good semantic clues for the target word they surround. The vector associated with a context is the sum of the basis vectors associated with the context words. Word vectors are then defined as the sum of all the context vectors for the sentences where they appear (Manning and Schütze, 1999, chap. 15).

The original dimension N is usually rather large and many dimensions are redundant or uninformative. In order to keep only the most significant variation in the vectors, dimensionality reduction techniques are used. The vectors learnt for each word define a matrix M, where M_{ij} is the number of times word i has occurred in the same context as the context word j. We can

apply Singular Value Decomposition to this matrix and keep only the top n singular values. This gives us n-dimensional vectors that are also empirically more useful for various tasks (Deerwester et al., 1990).

The notion of context can also be changed: overall, larger contexts lead to more topical representations, whereas smaller contexts reflect the syntactic roles of words (Zhao and Lin, 2005). Enforcing a greater sparsity in the original matrix by only keeping the K most frequent context vectors for a given word as also been shown to enhance the resulting vectors (Polajnar and Clark, 2014).

2.2.3 Distributed models

Another way to relate contexts to word vectors is to train word representations in a supervised way, using contexts as instances of a prediction problem (Mikolov et al., 2013). This approach is often referred to as *distributed* (whereas the previous one is *distributional*). By supervised we mean that a prediction accuracy is optimized, although no extra information is needed to train the model as the contexts themselves define the training samples and their target value. An interesting aspect of these models is that the word representations and the composition structure suited for them are trained simultaneously. A form of compositionality is hence built in the model.

Whether this approach is superior to the count-based method is still investigated. Baroni et al. (2014b) find that distributed models yield more informative vectors for various tasks, but Levy and Goldberg (2014) show that the popular neural model called skip-gram can be rephrased as an implicit matrix factorization comparable to the traditional dimensionality reduction techniques. They show that the improvements it provides can be transferred to count-based vectors using this analogy (Levy et al., 2015).

Predictive techniques can also be used in conjunction with syntactic structures, for instance by building recursively a neural network based on a tree generated by a context-free grammar Socher et al. (2013). We will see in

Chapter 3 that distributed methods are compatible with type-driven semantics.

2.3 Semantics of CCG

We have presented our frameworks for both syntax and semantics: in this section, we relate the two. This relationship is the principle of compositionality which states (in a simple form) that the meaning of a compound is a function of the meaning of the individual parts and the way they are composed. This principle has been applied very early to logical semantics, for instance by Montague (Partee et al., 1990). The search for an equivalent of this composition in distributional frameworks is more recent and is still an active area of research.

We present here the framework proposed in Coecke et al. (2011), which uses type-logical grammars to relate syntactic structure and algebraic composition operations. This approach was originally developed with a different grammatical formalism, Pregroup Grammar, which we introduce in section 2.3.1. We explain in the next section how this grammar relates to compact closed categories, and how these categories can be used as models. The application of this model to various kinds of sentences is presented in Section 2.3.2, and the last section explains how the same tools can also help address dimensionality problems.

2.3.1 Compact closed categories

The problem to be solved can be stated very simply: how should we take into account the type derivation associated with a given sentence to combine the individual distributional word meanings into the meaning of a sentence?

The proposal of Coecke et al. (2011) is to use a category-theoretic methodology to link syntax to semantics. This framework was originally developped

for pregroup grammars (Lambek, 2008; Preller and Sadrzadeh, 2011; Preller, 2005) but we will explain it here directly in the context of CCG.

We briefly recall the notion of monoidal and compact closed categories, the mathematical structures we use to define our semantics. These notions define abstract structures, which state nothing more than the basic laws we require on our semantic objects.

Definition 1. *A **(strict) monoidal category** $\mathcal{C}$ is:*

- *a collection of objects* $\mathrm{Ob}\,\mathcal{C} = \{A, B, \dots\}$
- *for each pair of objects* A, B *a collection of morphisms* $\mathcal{C}(A, B) = \{f, g, \dots\}$
- *for each object* A *an identity morphism* $1_A \in \mathcal{C}(A, A)$
- *a composition operation* $\circ : \mathcal{C}(A, B) \times \mathcal{C}(B, C) \to \mathcal{C}(A, C)$*, associative and with identities as neutral elements*[2]
- *a monoid operation* $\otimes$ *on objects, with the object* I *as neutral element*[3]
- *for each objects* A, B, C, D *an operation* $\otimes : \mathcal{C}(A, B) \times \mathcal{C}(C, D) \to \mathcal{C}(A \otimes C, B \otimes D)$

such that the following equation is satisfied when the compositions are defined:

$$(f_1 \otimes g_1) \circ (f_2 \otimes g_2) = (f_1 \circ f_2) \otimes (g_1 \circ g_2) \tag{2.1}$$

Intuitively, monoidal categories allow us to model processes applied to compound systems. The monoidal operation on objects allows pairing of two systems to form a larger system, the vertical composition $\circ$ allows application of sequential transformations to a system, and the horizontal composition $\otimes$ applies in parallel two processes to two systems.

Let us give an illustrated example of these abstract notions. Following Coecke and Paquette (2011), we can define a monoidal category about cooking

[2] This means that for $f \in \mathcal{C}(A, B)$, $1_B \circ f = f = f \circ 1_A$.
[3] This means that for all object A, $I \otimes A = A = A \otimes I$.

recipes: objects are ingredients and pairing two objects justs consists in placing them next to each other. Morphisms transform ingredients into other substances. For instance, we could have the morphisms $\mathbf{mix} : \mathit{milk} \otimes \mathit{oats} \to \mathit{raw\ porridge}$ and $\mathbf{cook} : \mathit{raw\ porridge} \to \mathit{porridge}$. Composing these two morphisms gives:

$$\mathbf{cook} \circ \mathbf{mix} : \mathit{milk} \otimes \mathit{oats} \to \mathit{porridge}$$

Suppose that, in addition, we want to have a toast with our porridge. Monoidal categories allow to toast the bread while cooking the porridge: given a morphism $\mathbf{toast} : \mathit{bread} \to \mathit{toast}$, we can take its product with $\mathbf{cook} \circ \mathbf{mix}$:

$$(\mathbf{cook} \circ \mathbf{mix}) \otimes \mathbf{toast} : \mathit{milk} \otimes \mathit{oats} \otimes \mathit{bread} \to \mathit{porridge} \otimes \mathit{toast}$$

The equation (2.1) is best explained using the graphical language introduced in Joyal and Street (1991). An arrow $f : A \to B$ is represented by the leftmost diagram in Figure 2.5. When the domain (respectively the codomain) is the monoidal unit I, we depict f as a box without input (respectively without output) and give it a triangular shape.

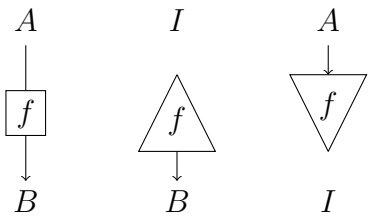


Figure 2.5: Representation of arrows as diagrams

The diagrams for composite arrows are defined as follows:

Figure 2.6: Representation of vertical and horizontal compositions with diagrams

With these conventions, the equation (2.1) takes the very simple form below:

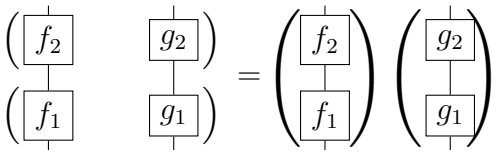


Figure 2.7: The bifunctoriality equation for diagrams

Yet, monoidal categories are not sufficient to give semantics to CCG as we have no way to represent quotients, i.e. systems where a sub-system is missing. We introduce the notion of *adjoint* for that purpose.

Definition 2. *In a monoidal category, an object A is a **left adjoint** of B (and B is a **right adjoint** of A) when there are two morphisms $\epsilon : A \otimes B \to I$ (the counit) and $\eta : I \to B \otimes A$ (the unit) such that*

$$(1_B \otimes \epsilon) \circ (\eta \otimes 1_B) = 1_B \qquad \textit{and} \qquad (\epsilon \otimes 1_A) \circ (1_A \otimes \eta) = 1_A \tag{2.2}$$

Again, we can use diagrams to make these equations clearer, provided we use appropriate representations for ϵ and η:

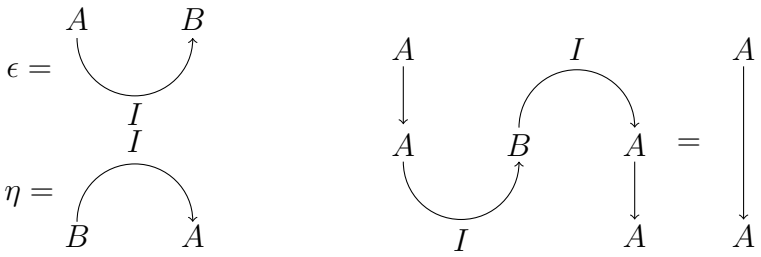


Figure 2.8: The first of the so-called *yanking equations* (2.2) in diagrams

Concretely, we will use this notion of adjoint only in *symmetric* monoidal categories. In this case, right adjoints and left adjoints are isomorphic, so we can simply ignore the direction.

Definition 3. *A symmetric monoidal category is* ***compact closed*** *when every object A has an adjoint, denoted by A^*.*

With this notion, we can represent the quotients of CCG on the objects. Let us define the semantics of CCG in any compact closed category $\mathcal{C}$. Concretely, this consists of a function $[\![-]\!] : \mathrm{CCG}_{\mathrm{types}} \to \mathrm{Ob}\,\mathcal{C}$ as well as a mapping from any type derivation F to a morphism f in $\mathcal{C}$. The domain of f is $[\![A_1]\!] \otimes \cdots \otimes [\![A_n]\!]$ where $A_1, \ldots, A_n$ are the premises of F, and $[\![B]\!]$ is its codomain, where B is the conclusion of F.

The interpretation of objects $[\![-]\!]$ is defined inductively. We fix interpretations $[\![A]\!]$ for all basic types A (generators of the set of CCG types).

$$[\![A/B]\!] = [\![A]\!] \otimes [\![B]\!]^*$$
$$[\![A\backslash B]\!] = [\![B]\!]^* \otimes [\![A]\!]$$

Note that the semantics of the two quotients actually coincide as the product $\otimes$ is symmetric. We choose to write the argument on the side where it is expected as this notation will simplify the translation of derivations.

The semantics of derivations is also defined inductively: we only need to define the semantics of individual rules and the representations of composite derivations will be uniquely determined. Table 2.1 presents these semantics, defined as string diagrams. We have included the semantics of more advanced

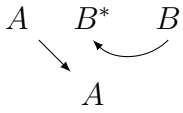
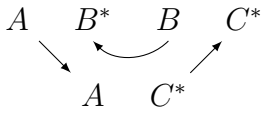
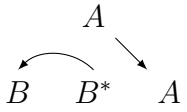
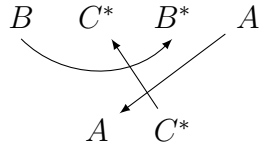
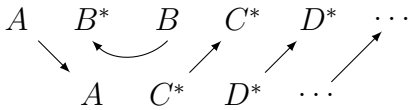
Rule	Syntax	Semantics
Application	$\frac{A/B \quad B}{A} >$	
Composition	$\frac{A/B \quad B/C}{A/C} >$	
Type raising	$\frac{A}{B/(B\backslash A)}$ T>	
Crossed	$\frac{B/C \quad A\backslash B}{A/C} < B_x$	
Generalized	$\frac{A/B \quad (\ldots(B/C)/D\ldots}{(\ldots(A/C)/D\ldots} < B^n$	

Table 2.1: Semantics of CCG rules in a compact closed category

rules and in particular the backward crossed composition rule to show why symmetry is needed. The crossings of the wires in the associated diagram correspond to the use of the symmetry isomorphism $A \otimes B \simeq B \otimes A$. This is a slight difference with the semantics of pregroup grammars where no symmetry is assumed, and hence no crossing is allowed in the diagrams. We refer the reader to Selinger (2011) for a very comprehensive review of the various graphical languages associated with these enriched categories.

Let us give a concrete example of how a derivation translates to a string diagram. Figure 2.9 gives the gold standard CCG derivation for the noun phrase *the bike that we saw*. Applying the translation defined in Table 2.1 gives the string diagram of Figure 2.10. This diagram can be simplified by applying the yanking equality 2.2 to get the simpler form of Figure 2.11.

In particular, it is interesting to observe that the semantics of CCG we have just defined preserve what is called the *spurious ambiguity* of CCG. This ambiguity is not a syntactic ambiguity, i.e. two different ways to understand the syntax of a sentence, but two different derivations that correspond to the same phrase structure.

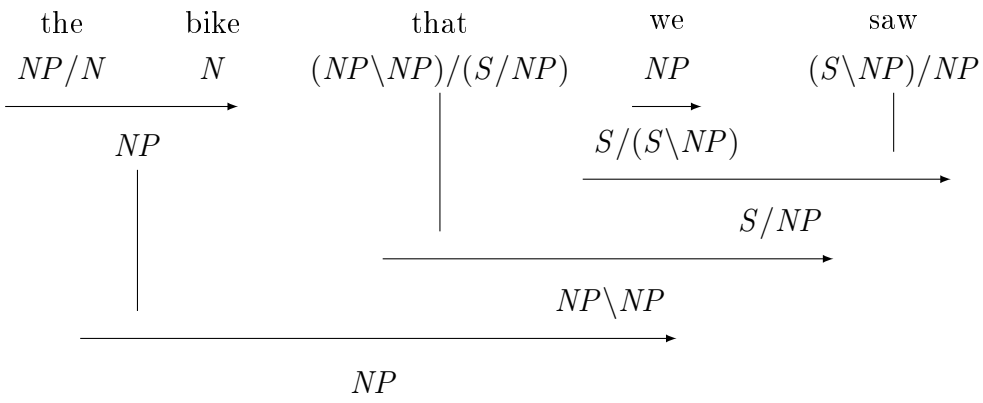


Figure 2.9: CCG derivation for the phrase *the bike that we saw*

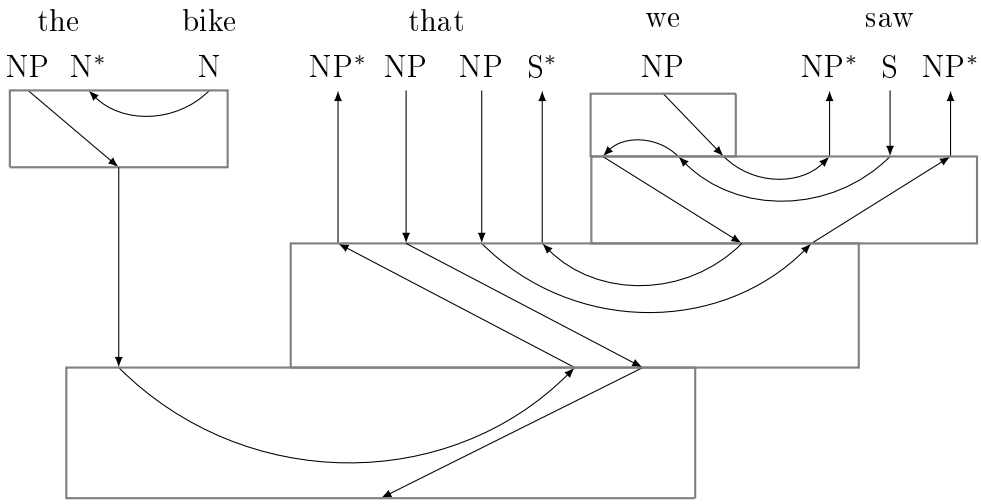


Figure 2.10: Translation of the derivation of Figure 2.9 to a compact closed category

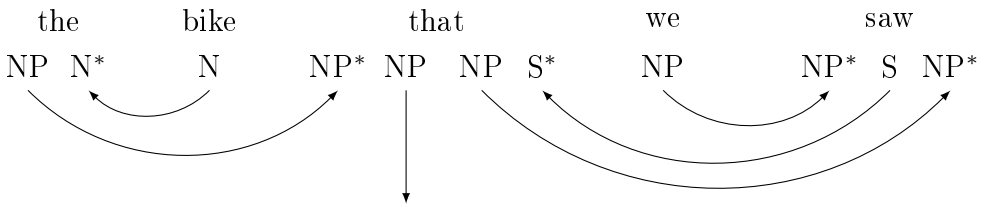


Figure 2.11: Reduced version of the diagram of Figure 2.10

A simple example of such an ambiguity is the fact that an application rule can be replaced by a type-raising rule followed by a composition. As we can see in Figure 2.14, the translations of the two different derivations are equal. This very useful feature of the semantics has been observed in the special case of the category of vector spaces with the tensor product (Maillard et al., 2014); we have just shown it categorically.

Figure 2.12: Two CCG derivations for the sequent $A/B \cdot B \vdash A$

Figure 2.13: Translations of the derivations of Figure 2.12

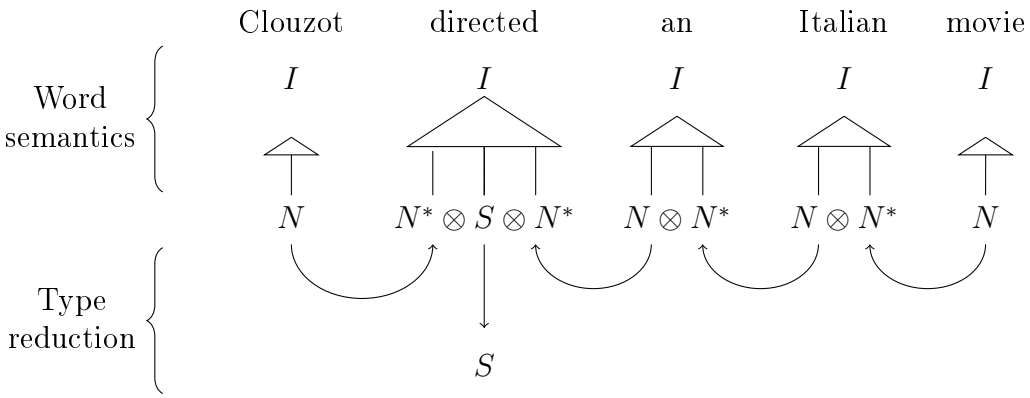
Figure 2.14: Reduced form of the derivations of Figure 2.13

2.3.2 Semantics of sentences

We have described in the previous section how to translate a CCG derivation to a morphism in a compact closed category, conveniently represented by a digram. We now show how this can be used to compose word meanings into the meaning of a sentence.

The core assumption of this categorical framework is that the semantic representation of a word is an object whose type is $[\![T]\!]$, where T is the CCG type of the word. This is not a straightforward assumption, as the widespread methods to learn word representations produce vectors of a fixed dimension, regardless of their types.

Let us assume for now that these word vectors are known. Categorically, we have morphisms $v_i : I \to [\![T_i]\!]$ for each word v_i of type T_i. The meaning of the sentence is defined as the composition of the word meanings with the type reduction: $f \circ (v_1 \otimes \cdots \otimes v_n)$, where v_i is the vector for the i-th word in the sentence, and f is the type reduction.



This is the main assumption of the categorical approach to compositional semantics. The concrete nature of the semantic representation depends on the category chosen. Our goal is to be able to compose distributional semantics, and we will hence use the category proposed in Coecke et al. (2011): objects are vector spaces, the monoid operation is the tensor product, and functions are linear maps.

Concretely, using this category means using the following recipe to compute meanings:

- Take the tensor product of word meanings $V = \bigotimes_i v_i$
- Translate the CCG type reduction to a string diagram as defined in Table 2.1.
- For each *cup* (or counit) in the string diagram, apply tensor contraction at the source and target of the cup.

This is a naive recipe because the computation of V requires a space exponential in the number of types. We will show later how we can make this

algorithm more efficient.

This theory of semantics is elegant, but it remains to be shown why this semantic composition is a sensible way to derive the meaning of a sentence. This problem is far from solved and still being actively investigated. We give here some insights about the pros and the cons of the theory.

One first argument for this categorical framework is that it coincides to some extent with another theory developed independently. Baroni and Zamparelli (2010); Baroni et al. (2014a) argue that words should have different representations depending on their type. They propose to model nouns as vectors and adjective as matrices, the representation of an *adjective-noun* compound being the product of the matrix and the vector associated to the words. It turns out that this is exactly what the categorical framework proposes for this particular case.

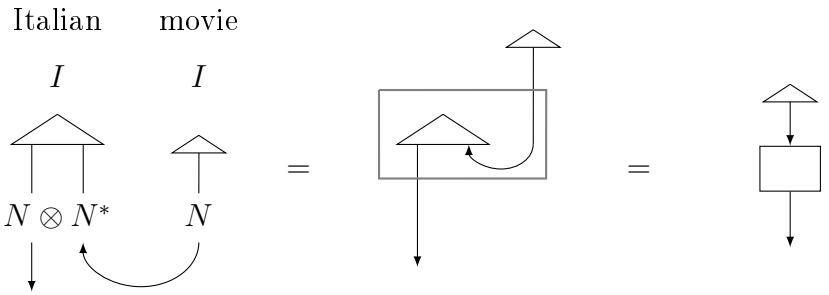


Figure 2.15: Correspondance between tensor contraction and function application

Figure 2.15 shows how the tensor contraction proposed by the categorical framework can be seen as a function application in the case of a second-order tensor. The leftmost diagram represents the tensor contraction based on the type reduction. Let us define a morphism, whose string diagram is the box in the center of the figure. We can reinterpret the same computation as the composition of an element and this morphism, i.e. the multiplication of a vector by a matrix, as shown in the right-hand side. Conversely, any matrix can be seen as a second-order tensor.

Another argument why this composition method is sensible are the concrete results obtained on practical tasks, such as the experiments of Grefenstette and Sadrzadeh (2011), showing that the model achieves a good performance on some sentence similarity tasks.

A third reason is the connection with logical models of meaning. It was noticed by Coecke et al. (2011) that if the coefficients of the vectors and the tensors are restricted to $\{0, 1\}$, we recover a sort of logical semantics. This connection between two very different paradigms has been explored (Preller, 2013; Preller and Sadrzadeh, 2011) and has led to interesting analogies. Let us explain one about adjectives. To recast a logical model of meaning into vector spaces, it is common to assume that the space for nouns N has one basis vector for each object in the *domain of discourse*.

For the sake of example, let us say that we have 4 such vectors: $e_{\text{Diana}}, e_{\text{Mike}}, e_{\text{Amanda}}, e_{\text{Joshua}}$. With this convention, a common noun is represented by a vector: the sum of the basis vectors whose denotation as an instance of this common noun. In our example, the noun *man* would get the vector $\overrightarrow{man} = \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}$. Then, a noun adjunct such as *married* can be modelled as a projection[4], whose image is the vector space spanned by the basis vectors where the predicate holds. Assuming that in our case only Diana and Mike are married, we get $\overrightarrow{married} = \text{Diag}(\begin{smallmatrix} 1 & 1 & 0 & 0 \end{smallmatrix})$. Applying the matrix $\overrightarrow{married}$ to the vector $\overrightarrow{man}$, we get the vector $\begin{pmatrix} 0 & 1 & 0 & 0 \end{pmatrix}$, which represents indeed the set of married men.

Although distributional models of meaning are very different, the insight that adjectives should be projections can be kept to some extent (Grefenstette et al., 2014). For instance, it has proved useful to estimate adjectives as $\sum_i u_i \otimes u_i$, where the u_i are the n example vectors the adjective is applied to in a corpus. This is a quite similar expression: if $(u_i)_i$ were an orthonormal basis, the estimated matrix would be a projection. Similarly, other approaches (Fried et al., 2015) minimize the rank of the estimated tensors: such a minimization can be seen as constraining the dimension of the

[4] A projection is a linear map $f : N \to N$ such that $f \circ f = f$. Such a map is the identity on $\text{Im}\, f \subseteq N$ and the null morphism on $\text{Ker}\, f \subseteq N$, with $\text{Im}\, f \oplus \text{Ker}\, f = N$.

image while trying to preserve the outputs of the predicate, which is similar to the estimation of a projection.

Chapter 3

Dimensionality reduction with diagrams

In this chapter, we address one of the major challenges of the distributional compositional framework, namely the dimensions of the tensors that we would need to estimate in order to provide a wide coverage semantics of English.

We first review the various attempts to reduce the number of parameters of tensors. Then, we propose a novel use of diagrams to design parameter-efficient models, using the notion of free compact closed category generated by a monoidal category. This work has been presented at the Advances in Distributional Semantics workshop of the International Conference on Computational Semantics (IWCS 2015). Finally, we show how our proposal could be concretely implemented, and present the tool we have developed.

3.1 Dimensionality issues

While the CCG types considered in this thesis are relatively short, CCGbank contains many types that are very long. For instance, here are a few types with their number of occurrences:

3176	$((S\backslash NP)\backslash(S\backslash NP))/((S\backslash NP)\backslash(S\backslash NP))$
232	$(((S\backslash NP)\backslash(S\backslash NP))\backslash((S\backslash NP)\backslash(S\backslash NP)))/NP$
84	$(((N/N)/(N/N))/((N/N)/(N/N)))/(((N/N)/(N/N))/((N/N)/(N/N)))$

If we want to use the categorical framework to derive the meaning of sentences including these types, we need to estimate tensors for the words occurring with these types. As $\dim A \otimes B = \dim A \times \dim B$, the number of parameters to estimate for these tensors is enormous: assuming that NP and S are mapped to the same vector space of dimension d, the dimension of the tensor spaces induced by the types above range from d^6 to d^{16}. Although these long types are relatively infrequent, there are a lot of them: the rebanked CCGbank contains 440 types of length more than 5, and they make up 4% of the type occurrences.

In fact, dimensionality issues are encountered even for shorter types. For instance, a lot of effort has been put into reducing the number of parameters for transitive verbs, which have type $(S\backslash NP)/NP$, of order 3 (Kartsaklis et al., 2012; Polajnar et al., 2014; Fried et al., 2015).

There are various ways to do so. The simplest way is perhaps to use a *plausibility space* (Polajnar et al., 2014) for the vector space associated with the sentence. This consists in representing sentence values by single real numbers, representing how plausible the sentence is, by analogy with the truth values of logical semantics. As the dimension of the sentence space is hence 1, this automatically reduces the dimension of the verb space to $(\dim \mathrm{NP})^2$: verbs become matrices instead of third-order tensors. In some experiments, the dimension of the sentence space is actually two, so that negation can be implemented as swapping the two dimensions. Such a representation is not suited to all applications however, and in particular not the one we are interested in, as we will see in Section 4.3.2.

When we need proper vector representations for sentences, one way to extend the plausibility space approach is to make use of a Frobenius algebra to expand the sentence space. Concretely, this consists in composing the verb matrix with the linear map $e_i \mapsto e_i \otimes e_i$, where $(e_i)_i$ is an orthonormal basis of

the noun space. This copies the subject or object vectors back to the sentence space, weighted by the plausibility given by the matrix. Suppose we have a plausibility matrix $\sum_{ij} P_{ij} \cdot e_i \otimes e_j$. Applying the Frobenius copy operator on the subject, we get a third order tensor $\sum_{ij} P_{ij} e_i \otimes e_i \otimes e_j$. If the sentence space is equal to the noun space, this is indeed a suitable tensor for a verb, in terms of dimensions at least. The meaning of a *subject verb object* sentence will then become $\overrightarrow{subject} \odot (P(\overrightarrow{object}))$, where $\odot$ is the component-wise product of vectors. This method, as well as the symmetric one for the object, has been proposed by Grefenstette and Sadrzadeh (2011) and Kartsaklis et al. (2014). They also use the Frobenius operator on both subject and object, leading to a larger sentence space $S = N \otimes N$. This way to expand the sentence space might look slightly ad-hoc at first, but is in fact motivated by analogies with logical models of meaning (Sadrzadeh et al., 2014), and is supported by experimental results on sentence similarity (Grefenstette and Sadrzadeh, 2011).

Another line of work consists in restricting the interaction between the subject and object of a verb. For instance, Polajnar et al. (2014) propose to apply two different matrices to the subject and object, and then concatenate the vectors obtained to get the sentence vector. In other words, the sentence vector is obtained by

$$\overrightarrow{sentence} = \left(\begin{array}{c|c} A & 0 \\ \hline 0 & B \end{array}\right) \left(\begin{array}{c} \overrightarrow{sub} \\ \hline \overrightarrow{obj} \end{array}\right) \tag{3.1}$$

The verb is hence a function $f : N \oplus N \to S$, where $\oplus$ is the direct sum (or cartesian product) of vector spaces. If n is the dimension of the noun space and s is that of the sentence space, this method (called **2Mat**) reduces the number of parameters to estimate from sn^2 to $2sn$, and it happens to improve the performance of the representation for their task.

Due to the concatenation operation however, **2Mat** cannot be directly recast within the original framework. The theory states that verbs have to be linear maps of the form $N \otimes N \to S$, taking the subject-object pair as a tensor

and returning the sentence representation. What **2Mat** provides here is a linear map $N \oplus N \to S$ taking the subject-object pair as a concatenation of the two vectors, and returning the sentence representation. The two are incompatible, because there is no function $f : N \otimes N \to N \oplus N$ such that for each $u, v \in N$, $f(u \otimes v) = (u, v)$, where (u, v) denotes the concatenation of u and v. Such a function does not exist for instance because for all $u \in N$, $f(u \otimes 0) = f(0)$). This is a limitation of the categorical framework that we will overcome in the next section.

A third approach is the rank minimization technique of Fried et al. (2015) mentioned earlier. They constrain verb tensors to have at most rank r, which means that they can be written as $\sum_{i=1}^{k} u_i \otimes v_i \otimes w_i$ for some vectors $(u_i, v_i, w_i)_{1 \leq i \leq r}$. For small values of r such as 20, this dramatically reduces the number of parameters while preserving an equivalent performance on sentence similarity tasks.

3.2 Semantics with monoidal categories

The categorical framework is mathematically elegant, and has impressive connections with both foundations of quantum physics (Coecke et al., 2011) and cognitive science (Clark et al., 2008). However, this does not make it automatically linguistically relevant. For instance, we have seen in the previous section that it leaves out efficient models of meaning such as the **2Mat** approach of Polajnar et al. (2014). Hence, it is interesting to assess what we really assume about distributional semantics when we state our recipe to derive the representation of a sentence.

One of the striking features of this framework is that it describes how to compose word representations together, without knowing what the vectors actually represent. The estimation of the model parameters is left unspecified, and although the vectors and matrices have been mainly estimated using distributional techniques, nothing prevents us from using other models, such as topics learnt with Latent Dirichlet Allocation (Blei et al., 2003)

or neural networks. In fact, this has been done for neural networks by Milajevs et al. (2014) and comparable performance is reported. This generality is appealing, but also suspicious: how could it be possible to devise a sensible way to compose vectors if we do not know what the vectors are? We would intuitively expect that the meaningful operations to combine neural embeddings or LDA topic distributions would be quite different. In this section, we introduce a way to make the categorical framework more flexible, adapting the composition operations to the properties of the vectors we manipulate.

First, let us summarize the core assumptions of the categorical approach. Following the simple recipe to derive the meaning of a sentence given in Section 2.3.2, we could give the following summary, putting aside computational considerations:

Composition in the category of vector spaces with the tensor product

The representation of a sentence can be obtained by taking the tensor product of the word meanings and applying tensor contraction as indicated by the type reduction.

As we have seen in the last section, this leaves out parameter-efficient approaches such as **2Mat**. More generally, it forbids additive compositions for other grammatical roles than verbs, even if they are sometimes legitimate given the structure of the vectors used.

For instance, Mikolov et al. (2013) uncover a relation between their word vectors: they report that the nearest neighbour[1] of $\overrightarrow{queen} - \overrightarrow{king} + \overrightarrow{man}$ is $\overrightarrow{woman}$, and that similar patterns can be observed for semantically relevant analogies, such as $\overrightarrow{biggest} - \overrightarrow{big} + \overrightarrow{small} \simeq \overrightarrow{smallest}$ or $\overrightarrow{Rome} - \overrightarrow{Italy} + \overrightarrow{France} \simeq \overrightarrow{Paris}$. If such relations are widespread enough, we want to model the meaning of an adjective such as *female* by a map adding the vector

[1] The nearest neighbour is here the closest noun vector for the $\mathcal{L}^2$ norm.

$\overrightarrow{woman} - \overrightarrow{man}$ to its argument:

$$\overrightarrow{female} : v \mapsto v + (\overrightarrow{woman} - \overrightarrow{man})$$

With this definition, we get indeed that $\overrightarrow{female}(\overrightarrow{king}) \simeq \overrightarrow{queen}$ and $\overrightarrow{female}(\overrightarrow{man}) = \overrightarrow{woman}$. More generally, we might want some words to combine the vectors of their various arguments by summing them.

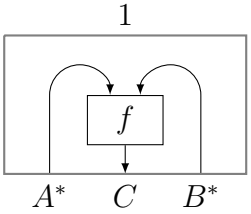
Hence, our first formulation of the categorical assumption is too restrictive. Luckily enough, category theory allows us to formulate it in a more general way, so that we can use any compact closed category for our semantics, not just the category of finite-dimensional vector spaces with the tensor product as monoidal operation:

Composition in any compact closed category

The representation of a sentence can be obtained by taking the ~~tensor~~ product of the word meanings and applying ~~tensor contraction~~ counits as indicated by the type reduction.

This is more general, but it is not clear yet what flexibility it brings. In general, constructing an ad-hoc compact closed category is complicated as this is a rather rich structure with many identities to satisfy. In the remainder of this section, we show how monoidal categories can be turned into autonomous categories, and how this can solve the problem raised in the example above. We give an informal explanation of the construction, a mathematical formalisation can be found in Delpeuch (2014). The main idea is to define a category whose morphisms are diagrams themselves, where the nodes are labelled by morphisms from the original category. In these diagrams, we allow ourselves to use units and counits even when there is no corresponding morphism in the original category.

For instance, given a morphism $f : A \times B \to C$ in a monoidal category $(\mathcal{C}, \times, I)$, we can define the following morphism $\tilde{f} : 1 \to A^* \cdot C \cdot B^*$ in the free compact closed category $(\tilde{\mathcal{C}}, \cdot, 1)$ generated by $\mathcal{C}$:

$\tilde{f} :=$ 

Intuitively, this morphism represents the original function f, with the inputs converted to outputs. This is very similar to the correspondence between tensor contraction and function application of Figure 2.15 except that in this case, the *tensor* is only a symbolic object. In this state, it is not very useful as we cannot use it in numerical computations. However, we can use it formally to define semantics that will eventually become numerical, thanks to the following theorem:

Theorem 1 (Blute et al. (1996))**.** *Let f be a morphism in the free compact closed category generated by $\mathcal{C}$. If its domain and codomains are products of objects of $\mathcal{C}$, then it is equal*[2] *to a morphism in $\mathcal{C}$.*

Let us show how this can be used in practice. In the example above, our goal was to be able to sum distributional representations, so let us define the category Aff, whose objects are finite-dimensional vector spaces and morphisms are affine[3] maps between them. We will equip this category with the direct sum $\oplus$ (or cartesian product) as monoidal operation. Concretely, this means that pairs of vectors are constructed by concatenating the vectors instead of taking their outer product. Let N and S be the vector spaces associated with noun phrases and sentences respectively. The sum operation of vectors in N is a morphism in Aff: $+ : N \oplus N \to N := (u, v) \mapsto u + v$. We can therefore analyze the sentence *Pat plants female seeds* as shown in Figure 3.1.

[2]Technically speaking, this theorem is more accurately stated by saying that the left adjoint (free functor) in the free-forgetful adjunction is full.

[3]An affine map is a map $f : x \mapsto \overrightarrow{f}(x) + b$ where $\overrightarrow{f}$ is a linear map and b is a constant vector.

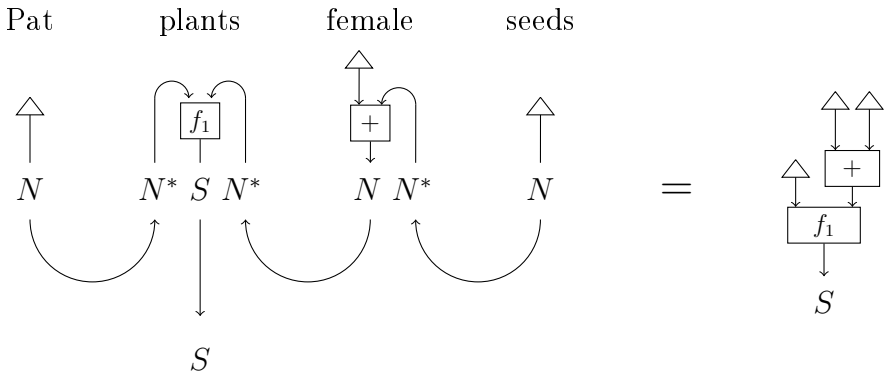
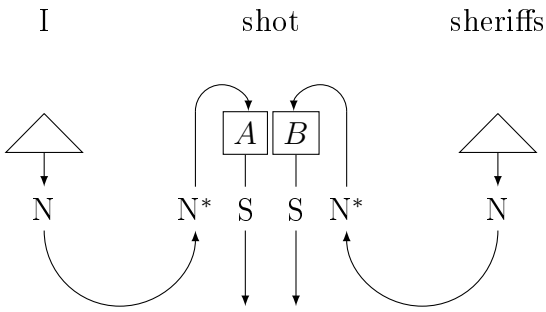


Figure 3.1: Semantics of adjectives with a free compact closed category

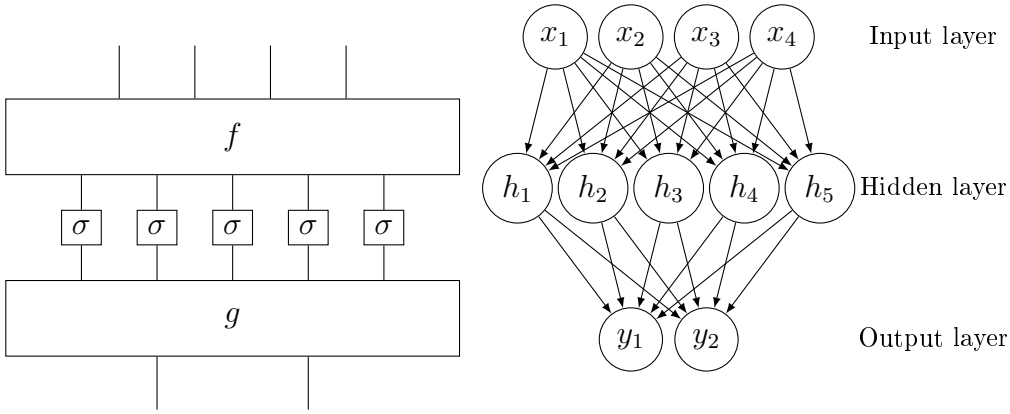
In the left-hand shape, the diagram cannot be used to compute the representation of the sentence, because the units and counits do not correspond to actual morphisms. Theorem 1 guarantees that we can eliminate them using the yanking equalities (2.2), and get the representation on the right-hand side, which is a valid diagram for a monoidal category. We get the expected representation for *female*: it adds a vector to its argument.

Similarly, we can now represent the **2Mat** approach:



We have applied this construction to allow for affine maps, but in fact it can be used with any monoidal category. In particular, we can even add non-linearities, leading us to type-driven neural models of meaning. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a nonlinearity. We define the category Net_σ as the monoidal category generated by σ and affine maps. Any neural network using σ as nonlinearity can be written as a series of compositions and pairing of affine

maps and σ and are hence arrows in Net_σ.



(a) An arrow in Net_σ. f and g are affine maps.

(b) Its traditional representation as a neural network

Figure 3.2: A categorical neural model of meaning in pictures

There are various reasons why using this model of meaning could be interesting. First, because it gives us some freedom in the functions we can use to combine meanings. The class of neural networks with one hidden layer can approximate any continuous function, with respect to the uniform norm on a compact set. This result has been obtained first by Cybenko (1989) for the case where σ is the sigmoid, a popular choice of activation function ($\sigma(x) = \frac{1}{1+e^{-x}}$) and has been generalized to any non-constant, continuous and bounded function by Hornik (1991). Of course, we could also directly use the monoidal category of such functions and freely generate the autonomous structure on it, but for concrete applications we need a finite parametrization of the objects we consider, and neural networks provide one.

Second, this result provides a way to define the representation of a sentence as a neural network whose shape depends on the syntactic structure. This is analogous to the syntax-driven convolutional neural network of Socher et al. (2013), where the sentence vector is recursively computed by neural networks at each node of the parse tree (using a Context Free Grammar).

3.3 Diagrammatic rewriting

In this section, we explain the importance of diagrammatic rewriting and give an overview of the tools we need for it.

Diagram rewriting consists in using equalities of functions written as equalities of diagrams to simplify the representation of a sentence. As we have seen in Section 2.3.1, the sequence of units and counits[4] induced by translating the CCG derivation to a compact closed category can be simplified by using the yanking equalities. This is witnessed in particular by the difference between Figures 2.10 and 2.11.

If we were to implement the categorical framework for arbitrary sentences, this diagrammatic rewriting would be crucial as it eliminates tensor expansion and contractions. For instance, if we translate naively the right hand side of Figure 2.13 in terms of tensor operations, we need first to expand the B tensor, and then perform a double tensor contraction. If we simplify the diagram first, we get the diagram on the left hand side, which requires only one tensor contraction. In particular, it is worth noting that the intermediate result before tensor contractions in the naive recipe has dimension $(\dim A)^3(\dim B)^2$, whereas the original tensor had dimension $(\dim A)(\dim B)^2$. This is our first example of how diagrammatic rewriting can serve distributional semantics: simply as a way to optimize computations. As most experiments carried out currently within the framework involve a fixed sentence structure, this rewriting is implicit, but we belive it is necessary for any attempt to deal with unrestricted parses.

In the case of the monoidal semantics defined in the previous section, diagrammatic rewriting is even not an optimization of the computation, it is mandatory for the computation itself. As the units and counits introduced by the construction have no interpretation in terms of concrete operations carried out on the vectors we manipulate, it is necessary to eliminate them first using the yanking equations.

[4] Units and counits are equivalently the tensor expansion and respectively contraction, if we use the traditional tensor-based model.

A concrete implementation of this proposal would hence require symbolic manipulation tools. This is where the use of category theory becomes useful: as the mathematical language we use is shared with other fields, we can reuse tools built for other purposes. In particular, the foundations of quantum physics also use diagrams in monoidal categories to model operations on quantum systems (Coecke, 2009). In this field, diagrammatic rewriting is used to give proofs of correctness of quantum algorithms and can be automated, using a software package called Quantomatic (Dixon et al., 2010). This software is generic enough to be used to perform the diagrammatic rewriting we need, as shown in Figure 3.3. In this graphical interface, green nodes represent morphisms from the original category, i.e. those we know how to compute and compose. The red nodes represent units and counits introduced by the construction. By applying successively the yanking equation, Quantomatic eliminates these formal units, until only green nodes remain.

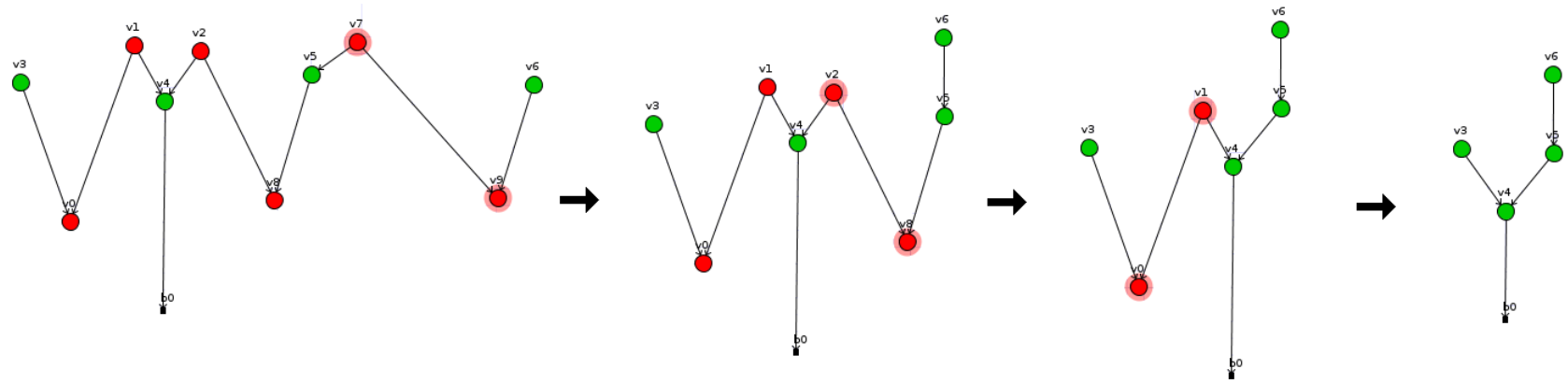


Figure 3.3: Automatic rewriting of Figure 3.1 with Quantomatic (screen shots from the graphical interface).

Chapter 4

The prepositional phrase attachment problem

In this chapter, we apply the distributional compositional model to study a particular form of syntactic ambiguity, the attachment of prepositional phrases. A prepositional phrase (PP) is a constituent whose head is a noun or verb phrase and is introduced by a preposition. By definition, prepositions occur before the constituent they introduce, such as in ***to*** *the river* or ***in*** *the news*. Words with the same role can occur after the constituent they dominate, in which case they are called postpositions, as in *an inner roadway a half-block* ***away***. As prepositions are much more common in English, we still call phrases with postpositions *prepositional*.

These phrases can occur in various constructions, for instance to modify nouns phrases, verb phrases or sentences. They can be coordinated, nested and extracted. They can hence introduce some ambiguity in the sentence structure, and the prepositional phrase attachment problem focuses on a particular kind of ambiguity, described in Section 4.1. Many learning strategies have been tried to predict how this ambiguity should be resolved, and Section 4.2 presents the main lines of research. In the last section of this chapter, we take a closer look at the treatment of prepositions in CCGbank, and analyse what it implies for the semantics.

4.1 Analysis of the problem

The PP attachment problem occurs when a verb is followed by a noun phrase, a preposition, and a noun phrase again. Two interpretations are usually possible:

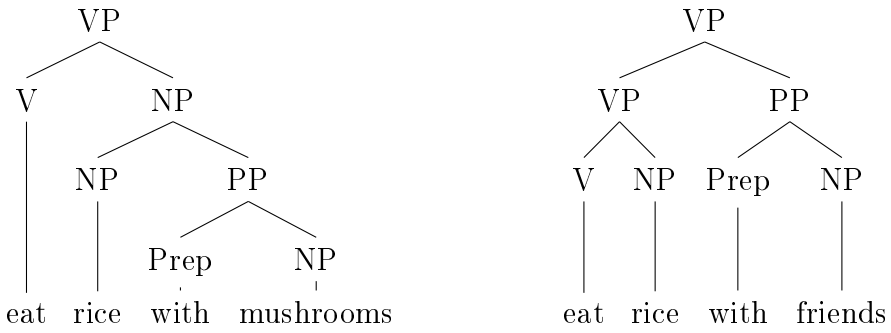


Figure 4.1: Two possible readings for the target phrase

The two interpretations often carry a quite different meaning and are hence not an artifact of a particular theory of syntax. This ambiguity is resolved seamlessly by humans but is a challenge for parsing technology as this decision is often made based on the plausibility of the meanings associated with the two parses.

As a side note, the ambiguity is also present for postpositions, as they can modify both verbs and nouns. For instance, the following sentence is formally ambiguous, as *two years ago* could be applied to *place*.

The interviews took place two years ago.

4.1.1 Ambiguity beyond the attachment location

Due to the importance of this disambiguation for accurate parsing, the problem has received a lot of attention and has been standardized as we have presented it. From a machine learning perspective, the problem is to create

the best binary classifier that predicts whether the preposition attaches to the verb or to the noun.

But in fact, the attachment location is not the only ambiguous property of parses, as Merlo and Ferrer (2006) points out. A prepositional phrase attached to a verb can either modify the verb phrase, giving extra information, or be a mandatory argument to the verb. For instance, in the sentence *She runs in a park*, the phrase *in a park* is an adjunct (or modifier), whereas *to a park* is an argument in *She goes to a park*. Although the linguistic tests used to determine the role of a prepositional phrase do not always provide a clear distinction between these two cases, the notion is widely accepted.

As we will see in Section 4.3, it is possible in CCG to make a difference between these two kinds of attachment, and to give them different semantics.

The distinction is also important from a cognitive perspective. There has been a long running debate about the purely syntactic biases that could influence attachment decisions. For instance, it has been suggested that readers are biased towards noun or verb attachments, or that they prefer structures where the attachment creates the fewest additional syntactic nodes (Frazier, 1978), for instance. These criteria do not correlate very well with experimental evidence. A more plausible criterion relies on the distinction between argument and adjuncts. Abney (1989) and Schütze (1995) argue that structures where a given prepositional phrase is an argument are preferred over those where it is an adjunct. They give the following sentence as example:

He thought about his interest in the Volvo.

Although *in the Volvo* could apply to the thinking, *interest* expects an argument introduced by *in*, hence noun-argument attachment is preferred. This can be intuitively understood considering that words accept a restricted set of prepositions to introduce an argument, and these arguments carry an important part of their meaning.

4.1.2 Influence of context

Another important caveat to bear in mind when considering the PP attachment problem is that we focus on one very particular kind of ambiguity. We do so because it is very widespread, but concretely it occurs in sentences where other forms of uncertainty might accumulate, or reduce the overall ambiguity of the sentence.

The first computational approaches to the problem (Hindle and Rooth, 1993; Ratnaparkhi et al., 1994) have progressively defined the PP attachment problem as a very particular machine learning problem. Samples are tuples *(verb, noun_1, preposition, noun_2)* where *noun_1* is the lexical head of the object of the verb, *noun_2* is the head of the object of the preposition. The task is to predict whether the preposition attaches to the noun or the verb.

Such a setting simplifies the problem in the sense that the information to be processed by the classifier is much simpler than if it had to rerank full parses. This helps to focus to the problem itself, and compare new approaches to previous ones easily as the task is independent from the grammatical framework used or the parser being used.

However, this problem is artificial. Atterer and Schütze (2007) argue that the final goal is to improve parsing accuracy and warn that the performance of such a simple classifier is quite unrelated to the improvement it can bring to a wide coverage parser. In fact, this reduced problem is actually harder than attachment decisions in context, because of the progress made in parsing technology since this period. Parsers are more lexicalized, and other syntactic constraints help them to make the right choice. For instance, in CCGbank, a large proportion of the cases where a verb has both an object and a prepositional adjunct happen inside relative clauses where there is actually no ambiguity at all:

The first of the three $200 million ships that Carnival has on order

This is a case where a prepopsitional phrase, *on order*, applies to a verb phrase with an object, but where there is no ambiguity as the object is

located elsewhere.

Although this criticism against a stand-alone classification task is indeed justified, we will still deal with the problem independently from parsing, for various reasons. The first one is that it allows for a simple comparison with many other approaches, and that it is simple to implement. The second one is that our goal is not to build a system that performs better than the state-of-the-art on this artificial task, but rather to evaluate the relevance of the distributional compositional framework for a particular syntactic structure. In the long term, we would like to know whether the categorical composition described in the previous chapter is suitable for wide coverage applications. The properties of some simple composition cases such as adjective-noun or subject-verb-object have been thoroughly studied, but little work has been carried out on closed-class function words such as prepositions. The only works we are aware of are models of negation (Hermann et al., 2013) and of relative pronouns (Clark et al., 2013). None of them estimate closed-class words from data but rather propose a distributional analogue of logical operators.

4.2 Related work

The problem of PP attachment has been tackled by many researchers using a variety of techniques. We survey here the main types of approaches, first starting with classical supervised methods. They apply a standard classification technique to predict the attachment based on features extracted from the data itself.

As the attachment decision often requires semantic information, these classifiers have been augmented with data extracted from ontologies. We review these extensions in the second section.

The use of these handcrafted information sources is not very satisfying, as they are expensive to create and have an inherently low coverage. Recently, the development of distributional techniques has enabled researchers to re-

place these features with unsupervised vector representations. We will cover these approaches in the last section.

4.2.1 Supervised learning

The syntactic criteria to predict attachment decisions mentioned in 4.1.1 yield poor classifiers: the particular words involved in the ambiguity are important to predict the attachment. Although it is hard to capture what aspect of these nouns drive the decision process, it is possible to collect statistics from a gold standard. The first attempts to guess automatically the correct attachment of a preposition, such as Hindle and Rooth (1993), relied on counting the number of times a preposition had been seen attaching to a given verb or noun. These counts were collected from parses generated by the Fidditch parser on an originally unannotated corpus. This parser produces incomplete parses, and in particular many prepositional phrases are left unattached when the syntax is ambiguous. Few errors are hence introduced in the training data by this preprocessing step.

The problem with this approach is that it only focuses on some particular frequency counts, whereas counting other combinations of words could be useful in some instances. This feature selection problem is solved in Ratnaparkhi et al. (1994) by using a maximum entropy classifier whose feature set is iteratively expanded. Their classifier has access to the tuple of *(verb, noun_1, preposition, noun_2)* and can count any combination of such words. Relying solely on words is not robust as two synonyms are treated as incomparable words. Their approach to word grounding uses the word clustering algorithm of Brown et al. (1992), whose byproduct is binary representation for words that encodes some semantic information about the words. Integrating features using these binary representations helps them to increase the accuracy of their maximum entropy classifier.

4.2.2 Ontology-based approaches

To integrate even more semantic information, one can use hand built resources such as ontologies. This information can be used in various ways.

Brill and Resnik (1994) use WordNet to learn attachment rules based on the semantic classes of the words involved. Knowing that *anchovies* and *cheese* are both edible makes it easier to generalize the training example *They eat a pizza with cheese* to predict a noun attachment for the sentence *They eat a pizza with anchovies*.

In Bailey et al. (2015), VerbNet provides information about selectional preferences for the verbs. By combining this with the hypernymy hierarchy provided by WordNet, they can leverage the bias towards argument attachment to increase the accuracy of their logistic regression classifier. For instance, the sentence *We base our analysis on this review* can be parsed using a verb attachment, knowing that all the frames in the VerbNet class `base-97.1` involve an argument introduced by *on*. In more complex cases where the type of the complement plays a role, WordNet helps them to relate the word to glosses in VerbNet frames.

These approaches are powerful because they have access to a very rich and structured source of information. But acquiring this knowledge is expensive and has not been done for most languages. Moreover, such a hard classification between word senses and frames could prove too rigid or sparse for sentences in different domains, even in the same language.

4.2.3 Unsupervised learning

Unsupervised approaches have been devised to replace ontologies by statistical models acquired from data. They often rely on the techniques introduced in Section 2.2 to ground words and learn relations between them.

The classifier of Zhao and Lin (2005) uses distributional word representations for all four words in the samples and uses Nearest Neighbours on these

vectors. The distance between two samples is defined as the sum of the pairwise distances of the vectors. To combat data sparsity, they gradually increase their beam as long as it does not contain enough samples to predict the attachment. The vectors themselves are trained using a short context window, so that they capture more syntactic similarities. They also train vectors using dependency relations instead of bare proximity-based contexts and observe an increase in accuracy. This approach yields state-of-the-art accuracy (86.5%) but it is not clear how the same ideas could be used for other forms of syntactic ambiguities.

The parse reranker of Belinkov et al. (2014) uses vector representation for words, learned with the distributed models of Section 2.2.3. These neural embeddings are used as word representations for all four words involved in the attachment decision. The vectors are then used as inputs for a neural network whose shape depends on the parse tree considered. The outputs of these two different neural networks are then scored using a linear function and compared, so that attachment is predicted for the structure that gets the highest score.

As the vectors they use have originally been trained to maximize another objective function, they tweak their vectors by gradient descent on their objective function. They also note that if the original vectors are trained using syntax-aware models, the performance on PP attachment increases. Their evaluation method relies on the integration of their classifier in a parser, using parsing accuracy as the score they want to maximize. To compete with state-of-the-art parsers, the word vectors are not sufficient: they need to integrate information from WordNet and VerbNet.

A third way to obtain topic-based representation for words is to use Latent Dirichlet Allocation, a Bayesian model where latent variables define topics. A modified version of this model has been used in Emerson and Copestake (2015) to learn topics for dependency relations. Each type of dependency relation is treated as a document in the original LDA formulation, and words are replaced by pairs of source-target words linked by the relation. The source and target words are emitted from topics, using dedicated probability

distributions over the vocabulary. Attachment decisions are then made by comparing the likelihood of the dependency relations for each attachment site. This method proves effective to improve the accuracy of PP attachment over a simple frequency-based baseline.

4.3 Prepositions in CCGbank

In this section, we review the treatment of prepositions in CCG, using the conventions of the rebanked CCGbank. We show that a fairly rich account of the various roles they can play is possible thanks to this type-driven approach.

In Section 4.3.2, we show how these syntactic choices influence the semantics. These consequences are not just technical constraints imposed by the framework but actually correspond to the semantic compositions we expect given the syntax, except for some cases of the use of the `PP` type.

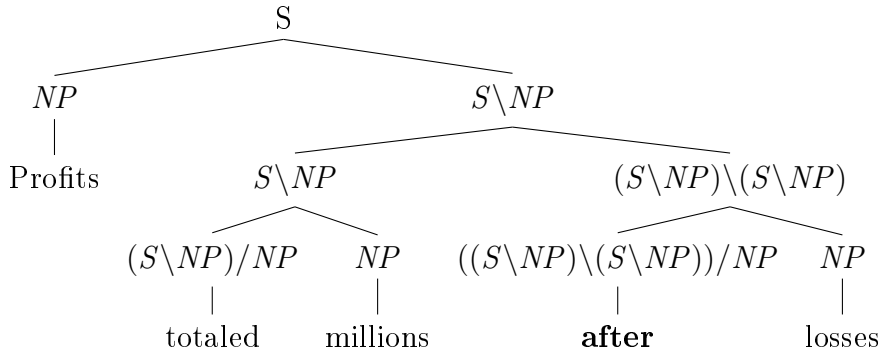
4.3.1 Types and their use cases

Prepositions are a good example of the fundamental difference between Context Free Grammars and Combinatory Categorial Grammar, because these few closed-class words can be observed with many types. The type given to a preposition (and the other words involved in the phrase) already determines much of the properties of the attachment discussed in Section 4.1. This is useful because in a statistical parser, the probability for a word to have a given type will depend on many lexicalized features. Yet, this is a tough decision for a tagger as some global view on the sentence is required to predict the correct tag. This requires a richer interaction with the parser than a simple pipelined approach (Clark and Curran, 2007).

We review in Table 4.1 the most frequent types with which prepositions can be found and analyze the properties of the attachment involved. The first column indicates the number of occurrences of each type, in the rebanked CCGbank. The properties of the attachment implied by the type are given

in the third and fourth column: *location* refers to the type of phrase the prepositional phrase attaches to, and *relation* indicates whether this type implies an argument or adjunct attachment. To understand better how these types work, we give some examples taken from the corpus in the last column.

Let us analyze the cases of verb attachment first. The type $((S\backslash NP)\backslash(S\backslash NP))/NP$ can look a bit cumbersome at first, but becomes much clearer when we set VP:=$S\backslash NP$: it becomes $(VP\backslash VP)/NP$, i.e. a word that becomes a VP right adjunct once combined with a NP on its right. In the context of an attachment problem, tagging the preposition with this type will hence imply a verb attachment. It also implies that the prepositional phrase[1] is an adjunct for the VP, as it modifies the full verb phrase.

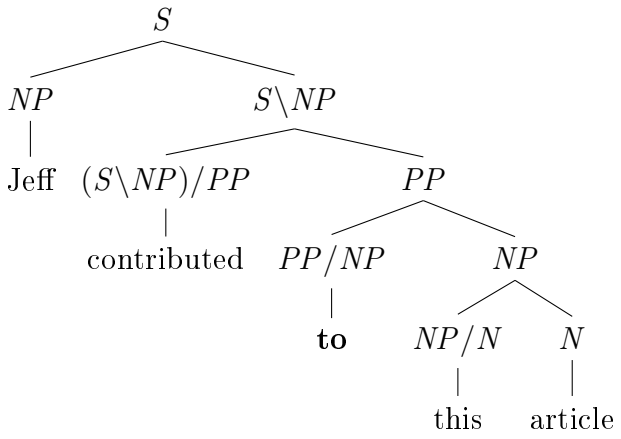


[1] In this section, we stop using the PP abbreviation for *prepositional phrase*, because it could be confused with the CCG type `PP`, which has a different meaning.

Freq.	Type	Location	Relation	Example
63591	PP/NP	Any	Argument	*Verb*: Jeff Rowe contributed$_{(S\backslash NP)/PP}$ **to** this article$_{NP}$ *Adjective*: the Richmond warehouse north$_{(NP\backslash NP)/PP}$ **of** San Francisco$_{NP}$ *Noun*: Part$_{NP/PP}$ **of** the problem$_{NP}$
21628	$((S\backslash NP)\backslash(S\backslash NP))/NP$	Verb	Adjunct	totaled $ 58 million$_{S\backslash NP}$ **after** the property sale loss$_{NP}$.
14256	$(N\backslash N)/NP$	Noun	Adjunct	the profits$_{N}$ **from** U.S. exploration$_{NP}$
6048	$(S/S)/NP$	Sentence	Adjunct	**For** instance$_{NP}$, franchisers no longer must ...$_{S}$
4033	$((S\backslash NP)/(S\backslash NP))/NP$	Verb	Adjunct	WCRS group, **for** its part$_{NP}$, will be able to ...$_{S\backslash NP}$
3266	$(NP\backslash NP)/NP$	Noun	Adjunct	a brief rescue$_{NP}$, **with** political undertones$_{NP}$
2378	$PP/(S\backslash NP)$	Any	Argument	high temparatures used$_{(S\backslash NP)/PP}$ **in** canning vegetables$_{S[ng]\backslash NP}$. Big companies are growing weary$_{(S[adj]\backslash NP)/PP}$ **of** fighting environmental movements.$_{S[ng]\backslash NP}$
2292	$(S\backslash S)\backslash NP$	Sentence	Adjunct	It was the case$_{S}$ two years$_{NP}$ **ago**
2168	$(S\backslash S)/NP$	Sentence	Adjunct	... increased by 62.3%,$_{S}$ **for** the 12-month period$_{NP}$
933	PP/PP	Any	Argument	wriggling$_{(S\backslash NP)/PP}$ **out** of horrible positions$_{PP}$
282	$(N\backslash N)/N$	Noun	Adjunct	an expected premium of 75 %$_{N}$ **to** 85 %$_{N}$
185	$(NP\backslash NP)\backslash NP$	Noun	Adjunct	..., compared with 82.2% the previous week and 86.2%$_{NP}$ a year$_{NP}$ **ago**
56	$(PP\backslash NP)/NP$	Noun	Both	spewing sulfurous material 190 miles$_{NP}$ **into** its atmosphere$_{NP}$
43	$(NP/NP)/NP$	Noun	Adjunct	some subgroups – **for** example$_{NP}$, married women with children at home$_{NP}$ – would be larger.

Table 4.1: Most common preposition (and postposition) types in CCGbank

Verb arguments are represented differently in CCGbank. When a verb expects an argument introduced by a preposition, it is witnessed by its type: instead of consuming an object of type NP, it will expect a special type PP.



Note that this convention still allows for optional arguments because verbs can be typed differently depending on the context. To parse the sentence *Jeff contributed very often*, it is enough to give *contributed* the type $S\backslash NP$. This is a difficulty for taggers as it increases the number of types verbs can have and choosing one sometimes requires a global knowledge of the sentence due to long range dependencies. On the other hand, this is valuable because it enables us to model the probability of verb subcategorization frames in the tagger itself. The rebanked CCGbank also encodes particles in verb types with a special `PR` type, as in "leave$_{((S\backslash NP)/NP)/PR}$ one out$_{PR}$", so verbs can occur with mostly any combination of NP, PP, and PR types as arguments, such as $((S\backslash NP)/NP)/PP$ or $(((S\backslash NP)/PP)/NP)/PR$.

The PP type is mostly used to introduce argument prepositional phrases, but sometimes its argument attachment is only spurious, for instance in the case of multi-word prepositions:

Solvents are in prospect because$_{((S\backslash NP)/(S\backslash NP))/PP}$ ***of*** *the Montreal Protocol*$_{NP}$

In this case, *of* is indeed the argument of *because*, but the full prepositional phrase is actually an adjunct of the verb phrase. Hence, most of the types

marked as arguments in Table 4.1 can actually be involved in adjunct attachments, although this is quite rare.

The most frequent type for prepositions introducing an adjunct to a noun is $(N\backslash N)/NP$, used in the case where the preposition introduces a noun phrase. In the original CCGbank, the type $(NP\backslash NP)/NP$ was used instead, both for argument and adjunct attachments. This was due to the uninformative treatment of noun phrases in the Penn Treebank. The rebanked version introduces the argument-adjunct difference for noun phrases, also using the type PP to represent argument prepositional phrases.

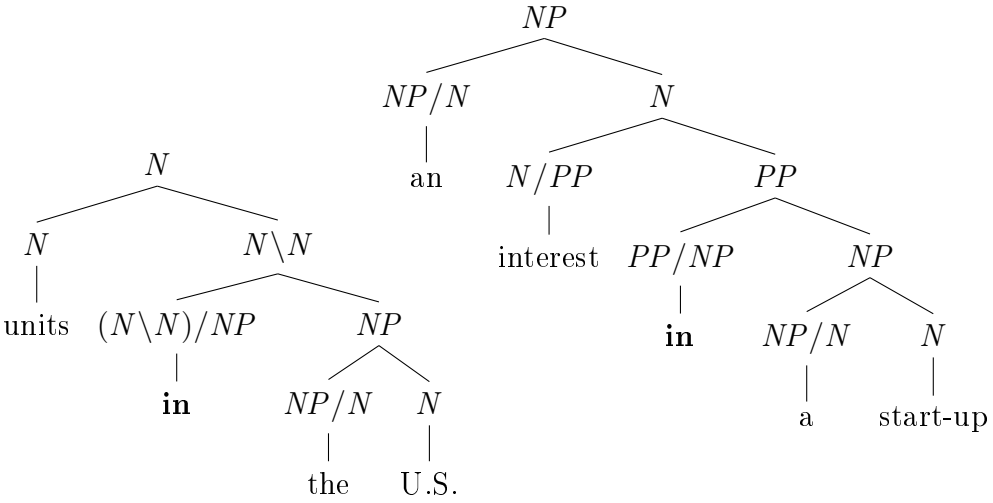


Figure 4.2: Adjunct and argument attachment to noun phrases

The framework also allows us to distinguish the prepositional phrases modifying verb phrases and those modifying whole sentences. This distinction is somewhat spurious, because these constructions are mostly equivalent in terms of meaning.[2] Concretely, it occurs when the prepositional phrase appears before the sentence, or is separated from the verb phrase with a comma. Sentence attachments correspond to the types $(S/S)/NP$, $(S\backslash S)/NP$ for prepositions and $(S/S)\backslash NP$, $(S\backslash S)\backslash NP$ for postpositions.

[2]From the perspective of pragmatics, this is not quite true as there can be a difference of emphasis or focus.

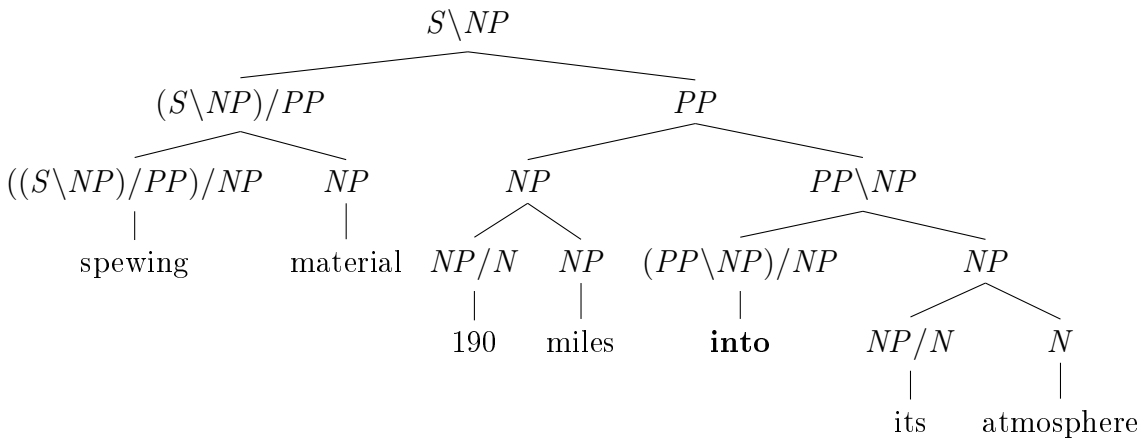


Figure 4.3: An example with both noun and verb attachments for the same preposition

Note that the distinction between these different attachment locations and types can become more complicated in some cases, for instance with prepositions bearing the type $(PP\backslash NP)/NP$. This type is used to combine two noun phrases to create a prepositional argument, for instance for a verb. It is hence used for prepositions introducing an adjunct for a noun phrase, and where the compound is in turn an argument for a verb. The derivation tree in such a case in shown in Figure 4.3. This shows again that the PP-attachment problem is quite artificial when considered as a binary classification problem.

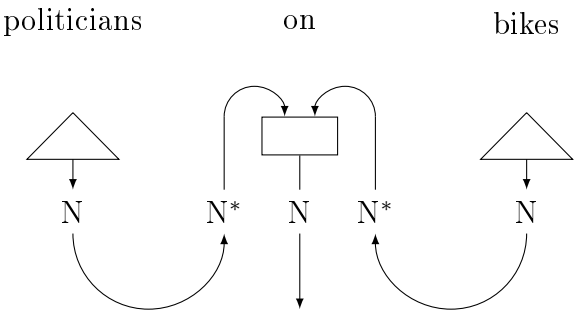
4.3.2 Semantics

Modelling sentences with prepositions in the distributional compositional framework of Section 2.3 requires us to define appropriate meanings for each *(preposition, type)* pair. In their most general form, these meanings are tensors whose dimensions are determined by the type. Luckily, the number of cases to consider is much lower than in Table 4.1, because the translation of the type to vector spaces forgets the direction of the quotients for instance. Moreover, we assume that we use the same vector space for the semantics of N and NP, written N, following common practice. The distributional compositional framework does not determine entirely the semantics, as we

still need to devise a method to estimate these tensors from data.

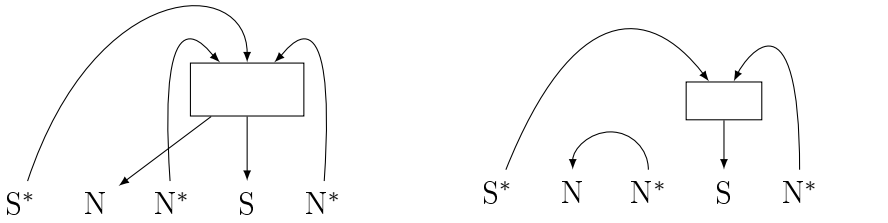
Adjunct attachments

We begin with the semantics for prepositions attaching as adjuncts, and focus first on noun adjunct attachments. As we can see in Table 4.1, the large majority of them use the types $(N\backslash N)/NP$ or $(NP\backslash NP)/NP$. The semantic representation for these types is an object in a third-order tensor space $N^* \otimes N \otimes N^*$, or equivalently a map from $N \otimes N$ to N, using the equivalence explained in Figure 2.15.



In the traditional framework using the tensor product, this means that the representation for the noun phrase *politicians on bikes* is $\overrightarrow{on}(\overrightarrow{politicians} \otimes \overrightarrow{bikes})$. The Kronecker operation $\otimes$ takes the two noun vectors of size n and returns a vector of size n^2, a flattened version of the outer product of the two vectors. The matrix $\overrightarrow{on}$, of size (n, n^2), is then applied to this vector.

Adjuncts for verb phrases are somewhat more complicated. As shown in Table 4.1, these prepositions use the type $((S\backslash NP)\backslash(S\backslash NP))/NP$ or a variant of it. Once translated to the semantics, this type is the product of five semantic spaces, $S^* \otimes N \otimes N^* \otimes S \otimes N^*$. This is rather long: suppose for instance the semantic spaces S and N both have 100 dimensions, which is a reasonable experimental setup. The dimension of the space for such prepositions would then have dimension $\dim \mathrm{S}^2 \times \dim \mathrm{N}^3 = 10^{10}$. This is of course a lot more than what we can afford on a computer, and having such a large number of parameters for a single preposition makes any form



(a) General form, of order 5

(b) Reduced form, of order 3

Figure 4.4: General and reduced form of semantics for verb adjunct prepositions

of training intractable. This is a good example of why the definition of the semantics of Section 2.3.2 is not a practical recipe for the meaning of a sentence, but rather an elegant abstract definition of compositionality.

To solve this dimensionality issue, one can leverage the fact that adjuncts for sentences and adjuncts for verb phrases carry the same meaning. In the following examples, the preposition *on* gets different types for each sentence, but the meaning remains the same:

The board enforces the regulations on my behalf.	$((S\backslash NP)\backslash(S\backslash NP))/NP$
The board enforces the regulations, on my behalf.	$(S\backslash S)/NP$
On my behalf, the board enforces the regulations.	$(S/S)/NP$
The board, on my behalf, enforces the regulations.	$((S\backslash NP)/(S\backslash NP))/NP$

This suggests that the general form for the VP adjunct preposition is unnecessarily general. The semantic representation of *on :* $((S\backslash NP)\backslash(S\backslash NP))/NP$ should essentially be the same as *on:* $(S\backslash S)/NP$. In other words, the noun phrase introduced by the preposition only modifies the result of the verb phrase once it has been applied to a subject, and does not need any direct interaction with the subject itself. But concretely, it is not straightforward to understand how to constrain the high dimensional representation of the larger type to match the smaller one. Again, diagrams come to the rescue: we simply need to draw a link between the two N types we want to cancel out, as shown in Figure 4.4.

By making the influence of the noun phrase on the right independent from the

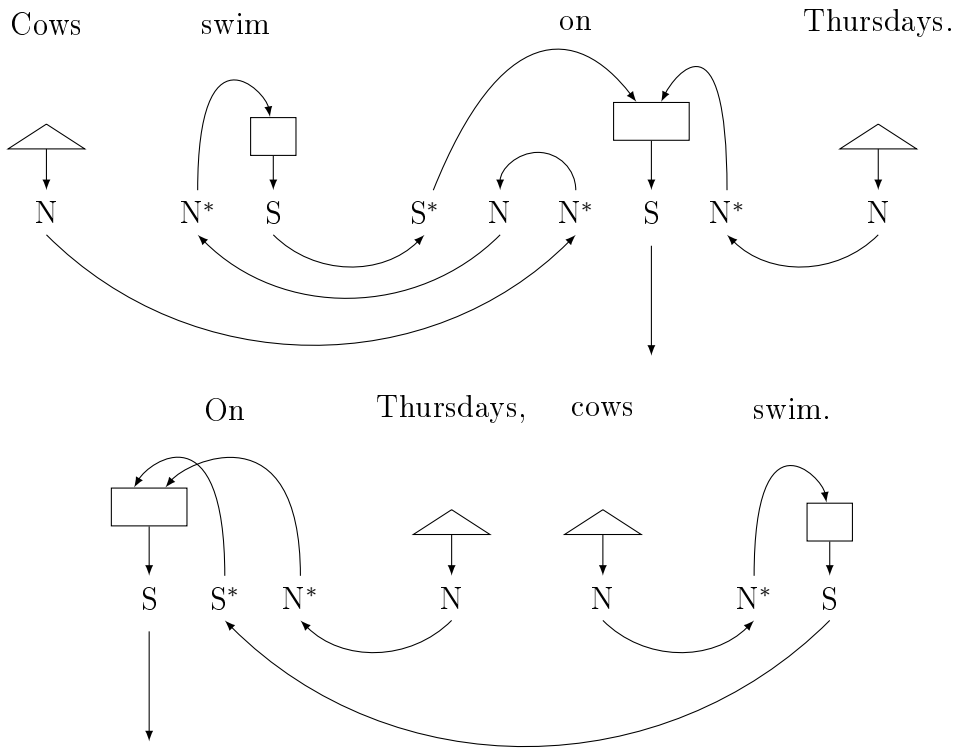


Figure 4.5: Reduced form of preposition ensures semantic equality of spurious syntactic differences

subject of the verb phrase, we reduce the order of the tensor from 5 to 3. The semantics is now fully determined by a tensor of dimension $\dim \mathrm{S}^2 \times \dim \mathrm{N}$, which is still large but manageable in practice. In fact, these estimations are based on the assumption that we use the traditional tensor-based model, but the simplification we have just described applies to any categorical model of meaning, in particular the ones we have sketched in Section 3.2. Moreover, we use the same tensor for all four types shown above, and this implies that the meanings of the corresponding sentences are identical.

Argument attachments

The semantics of argument prepositions is very different. Let us analyze first the case of the most common type, PP/NP. According to the framework,

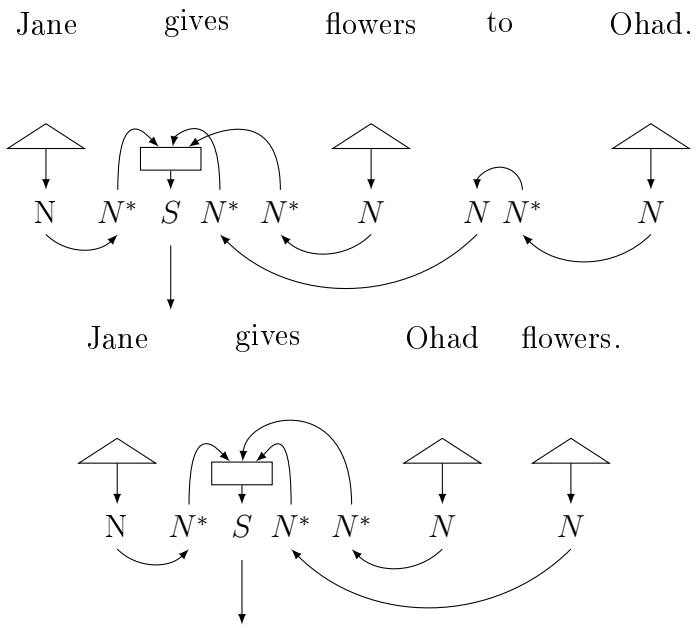


Figure 4.6: Representation of argument prepositions as identities

there should be a function $\overrightarrow{to} : N \rightarrow \text{PP}$ used to define in the meaning of the following sentences:

Outbreaks were traced $to_{PP/NP}$ *staphylococcus aureus.*

Net income rose $to_{PP/NP}$ *$213 million.*

The administration banned mushrooms in response $to_{PP/NP}$ *the outbreak.*

However, the meaning of *to* here is tightly linked to the meaning of the verb or the noun it attaches to. Hence, learning one generic function $\overrightarrow{to} : N \rightarrow PP$ does not make sense. Prepositions introducing arguments have no meaning in themselves but only indicate which one of the possible arguments they announce, allowing for omissions and reorderings. For instance, the verb *to give* can have the type $((S\backslash NP)/NP)/NP$ (*Jane gives Ohad a book*) or $((S\backslash NP)/PP)/NP$ (*Jane gives a book to Ohad*). Therefore, we propose to use the semantic space N for PP types as well. This allows us to use the identity for the semantics of prepositions with type PP/NP.

Although it works well for PP/NP, the use of N for PP types is not entirely

satisfactory. For instance, some prepositions introduce verb phrases, with type $PP/(S\backslash NP)$, so such a choice would require a generic mechanism to transform a verb phrase into a noun. Even worse, the analysis of multi-word prepositions involves a PP regardless of the attachment of the prepositional phrase. For instance, the preposition *instead of* can be used to combine two verb phrases and should hence have the type $((S\backslash NP)\backslash(S\backslash NP))/(S\backslash NP)$. But as each word should have a type, the following analysis is used:

remain independent $\underline{\textit{instead}}_{((S\backslash NP)\backslash(S\backslash NP))/PP}$ $\underline{\textit{of}}_{PP/(S\backslash NP)}$ *pursuing a buy-out*

Using N for PP here would create a bottleneck between *instead* and *of* which is not desirable in the semantics.

4.4 The PP attachment problem in pictures

We now have all the ingredients to study the PP attachment problem in the categorical framework, using the grammar of English defined by CCGbank.

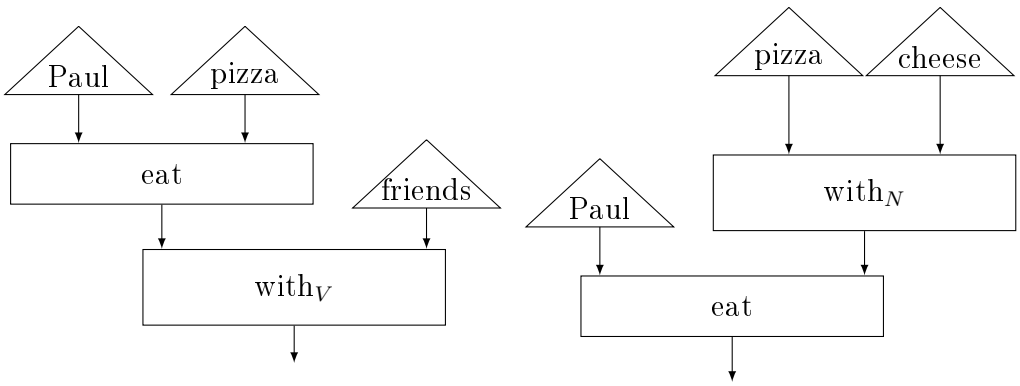


Figure 4.7: Composition structure for verb and noun adjunct attachments

Chapter 5

Experiments

In Natural Language Processing, semantics are bound to be imperfect approximations of what is really meant and understood in our sentences. To evaluate the quality of these representations, we use them to solve a task where we believe that some understanding of the words themselves is needed.

This is the reason why, despite all the criticism of the PP-attachment problem, it makes sense for us to work on this task. The goal is not to improve the state of the art of CCG parsing, but rather to evaluate distributional semantics and the composition method on a particular problem. Trying to directly improve a parser could be possible, but would make evaluation more complicated as various components influence the final parser accuracy.

In the first section, we explain why the Ratnaparkhi dataset (Ratnaparkhi et al., 1994) is too limited for our purpose, and how we were able to extract a more informative dataset from the rebanked CCGbank. We present a simple but effective baseline on this dataset, inspired by the standard baseline on the Ratnaparkhi dataset. Then, we describe our experimental setup to estimate distributional vectors and tensors. Finally, we evaluate our results and analyse the tensors learnt.

5.1 Extracting an enhanced dataset

The Ratnaparkhi dataset (Ratnaparkhi et al., 1994) is the standard resource to evaluate systems solving the PP attachment problem. Each sample is a tuple `(verb, noun_1, prep, noun_2, loc)`, where `noun_1` is the head of the verb object, `noun_2` is the head of the prepositional phrase introduced by `prep`, and `loc` indicates where the preposition attaches to: either the verb (`V`) or the noun (`N`). These tuples were automatically extracted from the Penn Treebank.

```
join board as director V
is chairman of N.V. N
named director of conglomerate N
caused percentage of deaths N
using crocidolite in filters V
bring attention to problem V
is asbestos in products N
led team of researchers N
making paper for filters N
including three with cancer N
```

Figure 5.1: The first 10 samples of the training set of Ratnaparkhi et al. (1994)

The standard task on this dataset is to predict the last field, the attachment location. As we have seen in Section 4.3.1, the ambiguity that CCG supertaggers have to deal with is much more diverse. The type of a given preposition usually encodes much more than the attachment location, and adds in particular the attachment type. In addition, some types – such as the most frequent one, PP/NP – do not even specify an attachment location but only that the prepositional phrase is an argument.

Moreover, we have explained in Section 4.1.1 that the cognitive biases for the PP attachment problem make more use of the argument/adjunct distinction than the noun/verb. This implies that the distinction is probably also interesting to study in a classification problem.

For these two reasons, we have extracted a new version of the Ratnaparkhi corpus, with argument (R) or adjoint (D) labels.

```
join board as director V D
is chairman of group N R
is chairman of N.V. N R
named director of conglomerate N R
led team of researchers N D
making paper for filters N D
have diseases including three V D
is finding among those V D
be highest for workers N D
regulate class including crocidolite N D
```

Figure 5.2: The ten first samples of our extended dataset

This dataset was extracted from the rebanked CCGbank (Honnibal et al., 2010), by filtering the predicate-argument dependences for each sentence. We have manually selected a set of possible types for each possible role in the tuple. Each combination of matching types with the corresponding predicate-argument links is extracted as a training sample, and the labels are deduced from the types, as explained in Section 4.3.1.

Although both datasets have been extracted from the same sentences, there are some differences between the two corpora. It would have been difficult to extract the same samples as in the original dataset, because the exact extraction procedure they have used is not explained in Ratnaparkhi et al. (1994). Moreover, many postprocessing steps have taken place betweeen the original Penn Treebank and the rebanked CCGbank, so even if we had access to the original heuristics, it would probably have been very hard to mimic it on our corpus.

Generally, prepositions cannot be detected simply by looking at their type: for instance, the word *said* gets the type $(S\backslash S)/NP$ in the following sentence:

We have no information on whether users are at risk said J. A. Talcott

However, looking at the part of speech is not enough either, as some prepo-

	of	in	to	for	on	from	with	at	as	by	into
NR	88.0	22.5	16.0	35.7	35.6	21.7	20.4	11.4	8.6	29.6	8.9
ND	9.3	22.0	3.3	15.2	9.2	5.5	16.7	10.9	28.5	2.4	0.2
VR	2.3	7.9	75.0	28.0	27.9	64.5	36.7	28.5	41.3	27.0	80.8
VD	0.3	47.4	5.4	20.9	27.1	8.2	26.0	49.1	21.5	40.8	9.8

	a	about	between	over	including	against	after	than	during
NR	0.0	66.5	68.8	24.7	1.9	41.3	0.6	0.7	0.0
ND	100.0	10.0	6.2	15.5	74.2	12.5	11.4	88.4	12.6
VR	0.0	20.5	4.8	13.7	0.4	28.3	1.3	0.0	0.0
VD	0.0	2.8	20.0	45.8	23.3	17.7	86.4	10.7	87.3

Table 5.1: Attachment statistics for the 20 most frequent prepositions
Key: N: noun, V: verb, R: argument, D: adjunct.

sitions such as *notwithstanding* or *including* are tagged VBG (gerund). We have hence used a closed list of 110 single word prepositions obtained from Wikipedia.

5.2 Baseline

The addition of the argument/adjoint information turns the original binary classification problem into a 4-class version, for which we have not found any previous work. We propose here a simple but already very effective baseline, against which our distributional semantics will be evaluated.

A simple approach to the attachment location prediction problem is proposed by Hindle and Rooth (1993). They compare the affinity between the preposition and either the verb or the object, and attach accordingly. The affinity between the verb v and the preposition p is defined as the ratio of training samples where p was attached to v, denoted by $f_V(v,p)$, over the number of training samples involving v as a verb, $c(v)$. The noun affinity is defined similarly. To account for the sparsity of such counts, they smooth the frequencies using the overall proportion of verb attachments for a given

preposition, $a(p)$.

$$\text{score}_V(v,p) = \frac{f_V(v,p) + a(p)}{c(v)+1} \qquad \text{score}_N(n,p) = \frac{f_N(n,p) + (1-a(p))}{c(n)+1}$$

The attachments are then predicted by choosing the greatest score. This simple method is very effective and achieves 80.2% accuracy on our test set. This accuracy is pulled upwards by prepositions that are strongly biased towards one attachment site or the other, such as *of* or *to*. Following Emerson and Copestake (2015), we also use a restricted set of more balanced prepositions[1], where it reaches 73.9% accuracy. A more interesting restriction is to evaluate the baseline on argument and adjunct attachments (as indicated by the gold standard of the test set). This approach gives 87.9% accuracy on argument attachments and 68.4% on adjuncts. The difference is not surprising as the lexical association between prepositions and the lexical head they attach to is stronger in the case of arguments, whereas the restriction a phrase imposes on its adjuncts is more semantic, and hence harder to capture with an approach considering words as atomic units.

This simple baseline can be extended to our new classification problem, where the relation between the head and the prepositional phrase has also to be predicted. For $x \in \{R, D\}$ (argument or adjunct), we define

$$\text{score}_{V,x}(v,p) = \frac{f_{V,x}(v,p) + a_x(p)}{c(v)+1}$$

and similarly for $\text{score}_{N,x}(n,p)$. The counts $f_{V,x}(v,p)$ denote the number of times the verb v has been attached with p under the relation x. The classifier chooses the best score among the four $\{(V,R),(V,D),(N,R),(N,D)\}$. Such a classifier achieves 71.1% accuracy on the test set. Table 5.2 shows its confusion matrix and confirms that in this case as well, adjuncts are harder to predict than arguments.

[1]These prepositions are: *on, at, by, for, from, in, as, to, with*

	ND	NR	VD	VR
ND	157	165	109	61
NR	15	620	59	83
VD	100	253	518	152
VR	29	152	57	815

Table 5.2: Confusion matrix of the lexical baseline on the full classification problem. Rows: gold standard, columns: prediction.

5.3 Distributional approach

We propose another approach based on the distributional compositional framework. As the lexical baseline presented in the last section is already very effective for arguments, we propose to improve the prediction of adjuncts. Argument attachments are made based on essentially lexical associations, as they arise with the combination of a lemma and a preposition (*report-on, interest-in*). Adjunct attachments are more topical and involve a form of semantic compatibility between the complement and the head. These are the cases where semantic interactions drive the decision process, and where distributional semantics could help.

Given a sample, we compute two sentence representations, first assuming that the prepositional phrase applies to the noun, then to the verb, in both cases as adjuncts. This gives us two sentence vectors, that we can use as features of an SVM to predict the attachment. This is similar to Zhao and Lin (2005) where distributional representations are used as features to a classifier. The difference is that in our case, these representations are combined together beforehand, reducing the dimension of the feature space.

The composition methods we use are directly taken from Figure 5.4. As the subject of the verb is not present in our samples, we actually compute the representation for verb phrase only. Therefore, we choose to estimate verb representations that ignore their subject, i.e. functions $v : N \to S$ instead of functions $v : N \otimes N \to S$ for transitive verbs. Let us denote by $n_1, n_2 \in N$

the two noun vectors, $p_n : N \otimes N \to N$ the tensor for the noun preposition p, and $p_S : S \otimes N \to S$. This composition process boils down to the following recipes:

$s_v \coloneqq p_v(v(n_1) \otimes n_2)$

(a) Composition for verb attachments

$s_n \coloneqq v \circ p_n(n_1 \otimes n_2)$

(b) Composition for noun attachments

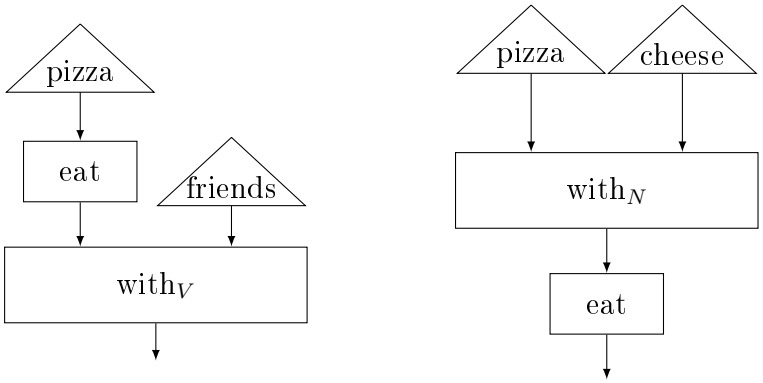


Figure 5.4: Composition structure for verb and noun adjunct attachments

5.4 Experimental setting

To turn these representations into actual procedures to predict attachments, we need to learn the parameters of the model from a corpus. Unfortunately, there is no off-the-shelf method to learn categorical models of meaning. In this section, we develop one, along the lines of the previous work on this topic, reviewed in Section 2.3.2. First, we estimate word vectors using standard distributional techniques, then we compute the tensors based on these words. Vectors and tensors are estimated using a copy of Wikipedia (October 2013). We used the tokeniser from the Stanford NLP tools, the Morpha lemmatiser (Minnen et al., 2000) and the C&C parser (Curran et al., 2007).

5.4.1 Word vectors

Many parameters influence the estimation of word vectors (Kiela and Clark, 2014), and the kind of semantic information they carry crucially depends on them. By its nature, the PP attachment problem is rather syntactic. Smaller contexts have been reported to represent better this form of information (Zhao and Lin, 2005). To make these contexts even more syntactic, we use the 100 most common prepositions and the 100 most common content words as context words. We use two different basis vectors for each preposition: one for the contexts where the target word appears before the context preposition, and one when it appears after. Target words are the union of the 10000 most frequent content words (adjectives, nouns, verbs, adverbs) and the words appearing in our PP attachment corpus. Cooccurrence counts are weighted with Positive Pointwise Mutual Information (PPMI):

$$\text{PPMI}(i,j) = \max(\text{PMI}(i,j), 0) \qquad \text{PMI}(i,j) = \log \frac{f_{i,j}}{f_i f_j}$$

where $f_{i,j}$ is the occrrence frequency of the context-target bigram, and f_i, f_j are their individual frequencies. The vectors are then reduced to a 100-dimensional space using Singular Value Decomposition as explained in Section 2.2.2. Words vectors are normalized to have unit $L2$ norm before and after SVD.

5.4.2 Verb matrices

As motivated in Section 5.3, our model of verbs ignores their subjects. Therefore, a verb is represented by a linear map $v : N \to S$, transforming the representation of its object to that of the sentence.

We estimate these matrices for the 10000 most common verbs in the corpus, by looking for instances where they are the source of a `dobj` dependency relation with a target word. Suppose that for a given verb v, there are N

such instances n^i in the corpus. The verb matrix is computed by:

$$V = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(n_i + v) \otimes n_i$$

This estimation method is motivated by the similar summation techniques of Grefenstette et al. (2014) presented in Section 2.3.2. This suggests to use sum $n_i \otimes n_i$, but we modify the output towards the verb's vector by changing it to $\frac{1}{2}(n_i + v)$ because the verb being the head of the verb phrase, its representation should have a significant influence on the representation of the verb phrase.

5.4.3 Prepositions attaching to noun phrases

Noun phrases are estimated similarly. We look for triples (n^1, p, n^2) where n^1, n^2 are target words, p is a preposition, and two dependency relations hold: `dobj` p n^2 and `ncmod` n^1 p.

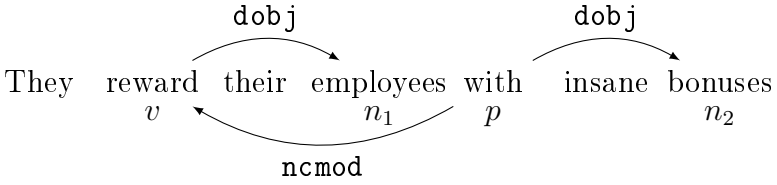
The tensor for the preposition is also obtained by summing outer products of vectors. Let N be again the number of training samples in the dataset.

$$P_N = \frac{1}{N}\sum_{i=1}^{N} n_i^1 \otimes n_i^1 \otimes n_i^2$$

Note that the output of the tensor (the first term) is again the lexical head of the noun phrase.

5.4.4 Prepositions attaching to verb phrases

For verb phrases, we look for more complex dependency patterns, as four words are involved.



We estimate again the preposition as the sum of the outer products of its arguments and desired output:

$$P_V = \frac{1}{N}\sum_{i=1}^{N} u_i \otimes u_i \otimes n_i^2 \qquad \text{where} \qquad u_i = \frac{v_i(n_i^1)}{\|v_i(n_i^1)\|}$$

The u_i vector is the vector associated with the inital verb phrase (v_i, n_i^1), and is computed using the verb matrix v_i as estimated previously, and the word vector n_i^1.

5.4.5 Classification method

This distributional model only covers adjunct attachments. To turn it into a full classification system, we pair it with the lexical scores produced by the baseline. Our goal is to mimic the argument preference explained in Section 4.1.1 using the following algorithm, parameterized by a threshold t:

1. Compute the lexical scores for noun and verb attachment
2. If the best lexical score exceeds t times the other lexical score, use the baseline to predict the attachment.
3. Otherwise, predict the attachment by using a Support Vector Machine classifier with the two sentence vectors as features.

Given a threshold t, the SVM is trained on the samples it would have had to classify on the training set, that is to say only the samples where the lexical scores are close.

5.5 Results

We split our new version of the Ratnaparkhi dataset into a training set, development set and test set, keeping the same proportions as the original dataset. The word vectors and tensors are learnt independently using the Wikipedia corpus. The SVM parameters and the threshold are optimized using a 5-fold cross-validated grid search on the training set. Setting the threshold to 1 gives the lexical baseline, setting it to a very large value gives a system that relies solely on the distributional vectors.

System	Arguments	Adjuncts	Both
Baseline ($t = 1$)	87.9	68.8	80.2
SVM, $t = 1.9$	**88.4**	68.9	**81.1**
SVM, $t = 6.0$	84.9	**71.8**	80.0

Table 5.3: Accuracy rates on the test set

We present our system with two different values for the threshold t. These have been tuned on the training set, first to optimize the overall accuracy ($t = 1.9$) and second for the adjunct attachment accuracy ($t = 6.0$). In the first case, our classifier achieves a better overall accuracy than the baseline. In the second, we observe a more significant increase on adjunct classification, which is what we expect as the distributional vectors have been trained for that purpose.

Chapter 6

Conclusion

Although the two parts of this project are very different in nature, they are two complementary ways to consider one general question: is the distributional compositional model suited to represent the meaning of arbitrary sentences, at a large scale and with wide coverage?

6.1 Alternate type-driven models of meaning

In the first part of this dissertation, we have shown that type-driven, compositional and categorical do not necessarily imply tensor-based, which opens up the framework to more economical and efficient representations. Some of them had already been considered as simplifications of the original model of meaning. Some others can be seen as adaptations of the popular neural models to type-driven syntax. We do not consider any of them as the panacea of distributional compositional models, but as they share many of the theoretical properties presented in the seminal work of Coecke et al. (2011), this is an incentive to reconsider why the community focuses on tensor-based models.

6.2 Distributional semantics of prepositions

Applying the distributional compositional model to arbitrary sentences is not only a technical challenge: we also have to ensure that this form of representation is suited for a wide range of syntactic roles. Although the distributional hypothesis is convincing for content words such as adjectives, verbs or nouns, its extension to closed class words with a more logical behaviour is still a challenge. Our contribution is to show that, to some extent, the model is also suited to represent prepositions, as the semantics it provides can be used to improve the performance of a strong baseline on the PP attachment problem, especially for adjunct attachments where purely lexical approaches perform poorly.

6.3 Future work

The development of symbolic rewriting tools to simplify the composition structure of sentences is a crucial prerequisite to our first proposal. Hence, a natural extension would be to continue their development and integration with existing tools.

However, it does not seem to match the current directions in the community. A very different line of research has recently been taken by Piedeleu et al. (2015), who propose to enrich the semantics by doubling the order of the tensors, using the Completely Positive Maps (CPM) construction. Inspired by analogies with quantum mechanics, this construction allows us to represent an additional level of superposition in the word tensors, that could be used for various purposes. Let us emphasize that such a construction effectively squares the dimensions of the vector spaces used. In the short term at least, this drives the distributional compositional model to a more theoretic agenda.

In the meantime, large coverage distributional models of meaning (Mikolov et al., 2013) tend to ignore syntax, considering it an inaccurate linguistic

vision rather than a useful guide for semantic compositions. This makes these models simpler, hence easier to design, train and tune, but does not necessarily imply that syntax has no role to play in the composition process. Overall, it seems that the theoretical and practical approaches to distributional semantics drift away from each other. We hope to have convinced the reader that the categorical model can still provide a conceptual bridge between them.

Bibliography

Steven P Abney. A computational model of human parsing. *Journal of psycholinguistic Research*, 18(1):129–144, 1989.

Michaela Atterer and Hinrich Schütze. Prepositional phrase attachment without oracles. *Computational Linguistics*, 33(4):469–476, 2007.

Daniel Bailey, Yuliya Lierler, and Benjamin Susman. Prepositional phrase attachment problem revisited: how Verbnet can help. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 12–22, London, UK, April 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W15/W15-0102.

Marco Baroni and Roberto Zamparelli. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics, 2010.

Marco Baroni, Raffaela Bernardi, and Roberto Zamparelli. Frege in space: A program of compositional distributional semantics. *Linguistic Issues in Language Technology*, 9, 2014a.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 238–247. Association for Computational Linguistics, 2014b. URL http://aclweb.org/anthology/P14-1023.

Yonatan Belinkov, Tao Lei, Regina Barzilay, and Amir Globerson. Exploring compositional architectures and word vector representations for prepositional phrase attachment. *Transactions of the Association for Computational Linguistics*, 2:561–572, 2014.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

Richard F Blute, J Robin B Cockett, Robert AG Seely, and Todd H Trimble. Natural deduction and coherence for weakly distributive categories. *Journal of Pure and Applied Algebra*, 113(3):229–296, 1996. doi: 10.1016/0022-4049(95)00159-X.

Eric Brill and Philip Resnik. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th conference on Computational linguistics-Volume 2*, pages 1198–1204. Association for Computational Linguistics, 1994.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1179.

Noam Chomsky. Three models for the description of language. *Information Theory, IRE Transactions on*, 2(3):113–124, 1956.

Stephen Clark and James R Curran. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4): 493–552, 2007. doi: 10.1162/coli.2007.33.4.493.

Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. A compositional distributional model of meaning. In *Proceedings of the Second Quantum Interaction Symposium (QI-2008)*, pages 133–140, 2008.

Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. The Frobenius anatomy of relative pronouns. In *The 13th Meeting on the Mathematics of Language*, page 41, 2013.

Bob Coecke. Quantum picturalism. *Contemporary Physics*, 51:59–83, 2009. doi: 10.1080/00107510903257624. arXiv:0908.1787.

Bob Coecke and Éric Oliver Paquette. Categories for the practising physicist. In *New Structures for Physics*, pages 173–286. Springer, 2011.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36(1-4):345–384, 2011.

James R Curran, Stephen Clark, and Johan Bos. Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 33–36. Association for Computational Linguistics, 2007.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989. doi: 10.1007/BF02551274.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *JAsIs*, 41(6):391–407, 1990.

Antonin Delpeuch. Autonomization of Monoidal Categories. *arXiv preprint arXiv:1411.3827*, November 2014.

Lucas Dixon, Ross Duncan, and Aleks Kissinger. Open graphs and computational reasoning. In S. Barry Cooper, Prakash Panangaden, and Elham Kashefi, editors, Proceedings Sixth Workshop on *Developments in Computational Models: Causality, Computation, and Physics*, Edinburgh, Scotland, 9-10th July 2010, volume 26 of *Electronic Proceedings in Theoretical Computer Science*, pages 169–180. Open Publishing Association, 2010. doi: 10.4204/EPTCS.26.16.

Guy Emerson and Ann Copestake. Leveraging a semantically annotated corpus to disambiguate prepositional phrase attachment. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 1–11, London, UK, April 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W15/W15-0101.

J Firth. *Papers in Linguistic [s J]*. Oxford University Press, 1951.

Lyn Frazier. *On comprehending sentences: syntactic parsing strategies*. PhD thesis, University of Connecticut, 1978.

Daniel Fried, Tamara Polajnar, and Stephen Clark. Learning low-rank tensors for transitive verbs. In *Advances in Distributional Semantics Workshop*, 2015.

Edward Grefenstette and Mehrnoosh Sadrzadeh. Experimental support for a categorical compositional distributional model of meaning. In *EMNLP*, pages 1394–1404, 2011.

Edward Grefenstette, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke, and Stephen Pulman. Concrete sentence spaces for compositional distributional models of meaning. In *Computing Meaning*, pages 71–86. Springer, 2014. doi: 10.1007/978-94-007-7284-7_5.

Karl Moritz Hermann, Edward Grefenstette, and Phil Blunsom. "not not bad" is not "bad": A distributional account of negation. *Proceedings of the ACL Workshop on Continuous Vector Space Models and their Compositionality*, August 2013. URL http://www.aclweb.org/anthology/W13-3209.

Donald Hindle and Mats Rooth. Structural ambiguity and lexical relations. *Computational linguistics*, 19(1):103–120, 1993.

Julia Hockenmaier and Mark Steedman. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396, 2007. doi: 10.1162/coli.2007.33.3.355.

Matthew Honnibal, James R Curran, and Johan Bos. Rebanking CCGbank for improved NP interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 207–215. Association for Computational Linguistics, 2010.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991. doi: 10.1016/0893-6080(91)90009-T.

André Joyal and Ross Street. The geometry of tensor calculus, I. *Advances in Mathematics*, 88(1):55–112, 1991. doi: 10.1016/0001-8708(91)90003-P.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. A unified sentence space for categorical distributional-compositional semantics: Theory and experiments. In *In Proceedings of COLING: Posters*. Citeseer, 2012.

Dimitri Kartsaklis, Nal Kalchbrenner, and Mehrnoosh Sadrzadeh. Resolving lexical ambiguitiy in tensor regression models of meaning. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, June 2014.

Douwe Kiela and Stephen Clark. A Systematic Study of Semantic Vector Space Model Parameters. In *Proceedings of EACL 2014, Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, 2014.

Joachim Lambek. Pregroup Grammars and Chomsky's Earliest Examples. *Journal of Logic, Language and Information*, 17(2):141–160, 2008. doi: 10.1007/s10849-007-9053-2.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

Jean Maillard, Stephen Clark, and Edward Grefenstette. A Type-Driven Tensor-Based Semantics for CCG. *EACL 2014 Type Theory and Natural Language Semantics Workshop*, 2014. URL http://goo.gl/JTEKLR.

Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

Paola Merlo and Eva Esteve Ferrer. The notion of argument in prepositional phrase attachment. *Computational Linguistics*, 32(3):341–378, 2006.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. Evaluating neural word representations in tensor-based compositional settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, October 2014. URL http://arxiv.org/pdf/1408.6179v1.pdf.

Guido Minnen, John Carroll, and Darren Pearce. Robust, applied morphological generation. In *Proceedings of the first international conference on Natural language generation-Volume 14*, pages 201–208. Association for Computational Linguistics, 2000.

Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *ACL*, pages 236–244, 2008.

Barbara Partee, Alice Ter Meulen, and Robert Wall. *Mathematical methods in linguistics*, volume 30. Springer Science & Business Media, 1990.

Robin Piedeleu, Dimitri Kartsaklis, Bob Coecke, and Mehrnoosh Sadrzadeh. Open system categorical quantum semantics in natural language processing. In *6th Conference on Algebra and Coalgebra in Computer Science (CALCO)*, 2015. URL http://arxiv.org/abs/1502.00831.

Tamara Polajnar and Stephen Clark. Improving distributional semantic vectors through context selection and normalisation. In *Proceedings of EACL*, pages 230–238, 2014.

Tamara Polajnar, Luana Fagarasan, and Stephen Clark. Reducing dimensions of tensors in type-driven distributional semantics. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1036–1046. Association for Computational Linguistics, 2014. URL http://aclweb.org/anthology/D14-1111.

Anne Preller. Category theoretical semantics for pregroup grammars. In *Logical aspects of computational linguistics*, pages 238–254. Springer, 2005. doi: 10.1007/11422532_16.

Anne Preller. From logical to distributional models. In *Proceedings of the 10th Workshop on Quantum Physics and Logic (QPL)*, Barcelona, Spain, 2013.

Anne Preller and Mehrnoosh Sadrzadeh. Semantic vector models and functional models for pregroup grammars. *Journal of Logic, Language and Information*, 20(4):419–443, 2011. doi: 10.1007/s10849-011-9132-2.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the workshop on Human Language Technology*, pages 250–255. Association for Computational Linguistics, 1994.

Mehrnoosh Sadrzadeh, Stephen Clark, and Bob Coecke. The Frobenius anatomy of word meanings II: possessive relative pronouns. *Journal of Logic and Computation*, 2014. doi: 10.1093/logcom/exu027. URL http://logcom.oxfordjournals.org/content/early/2014/06/02/logcom.exu027.abstract.

Carson T Schütze. PP attachment and argumenthood. *MIT working papers in linguistics*, 26(95):151, 1995.

Peter Selinger. A survey of graphical languages for monoidal categories. In *New Structures for Physics*, volume 813 of *Lecture Notes in Physics*, pages 289–233. Springer, 2011. doi: 10.1007/978-3-642-12821-9_4.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer, 2013.

Mark Steedman. *The syntactic process*. MIT Press, 2000.

Ludwig Wittgenstein. *Tractatus Logico-Philosophicus*. Keagan Paul, 1922.

Ludwig Wittgenstein. *Philosophical Investigations*. Blackwell, 1953. (2010).

Shaojun Zhao and Dekang Lin. A nearest-neighbor method for resolving pp-attachment ambiguity. In *Natural Language Processing–IJCNLP 2004*, pages 545–554. Springer, 2005.