QUEEN MARY UNIVERSITY OF LONDON

PHD THESIS

A Compositional Vector Space Model of Ellipsis and Anaphora

Author:
Gijs Jasper WIJNHOLDS

Supervisors:
Dr. Merhnoosh SADRZADEH
Prof. Dr. Edmund ROBINSON

Independent Assessor:
Dr. Paulo Oliva

Co-supervisor:
Prof. Dr. Michael MOORTGAT

Submitted in partial fulfillment of the requirements of the Degree of Doctor of Philosophy

Theory Group
School of Electronic Engineering and Computer Science

July 28, 2020

Statement of Originality

I, Gijs Jasper WIJNHOLDS, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party?s copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Signed:

Date: July 28, 2020

Details of collaboration and publications: all original work described in this thesis was done in collaboration with Mehrnoosh Sadrzadeh. Section 3.3 arose from a collaboration with Michael Moortgat and Section 3.4 from a collaboration with Michael Moortgat and Mehrnoosh Sadrzadeh. The work described in Chapter 5 arose from a collaboration with Mehrnoosh Sadrzadeh and Stephen Clark.

Abstract

Gijs Jasper WIJNHOLDS

A Compositional Vector Space Model of Ellipsis and Anaphora

This thesis discusses research in compositional distributional semantics: if words are defined by their use in language and represented as high-dimensional vectors reflecting their co-occurrence behaviour in textual corpora, how should words be composed to produce a similar numerical representation for sentences, paragraphs and documents? Neural methods learn a task-dependent composition by generalising over large datasets, whereas type-driven approaches stipulate that composition is given by a functional view on words, leaving open the question of what those functions should do, concretely.

We take on the type-driven approach to compositional distributional semantics and focus on the categorical framework of Coecke, Grefenstette, and Sadrzadeh [CGS13], which models composition as an interpretation of syntactic structures as linear maps on vector spaces using the language of category theory, as well as the two-step approach of Muskens and Sadrzadeh [MS16], where syntactic structures map to lambda logical forms that are instantiated by a concrete composition model. We develop the theory behind these approaches to cover phenomena not dealt with in previous work, evaluate the models in sentence-level tasks, and implement a tensor learning method that generalises to arbitrary sentences.

This thesis reports three main contributions. The first, theoretical in nature, discusses the ability of categorical and lambda-based models of compositional distributional semantics to model ellipsis, anaphora, and parasitic gaps; phenomena that challenge the linearity of previous compositional models. Secondly, we perform an evaluation study on verb phrase ellipsis where we introduce three novel sentence evaluation datasets and compare algebraic, neural, and tensor-based composition models to show that models that resolve ellipsis achieve higher correlation with humans. Finally, we generalise the skipgram model [Mik+13] to a tensor-based setting and implement it for transitive verbs, showing that neural methods to learn tensor representations for words can outperform previous tensor-based methods on compositional tasks.

Acknowledgements

First of all, I with to express my gratitude towards my first and foremost supervisor, Mehrnoosh Sadrzadeh. For all your help, discussions, support, and so much more, *thanks a lot*. Next, I wish to thank Edmund Robinson for all his support in his alternating role as second, first, second, and first, supervisor for my PhD. Then, thanks to Paulo Oliva for serving as independent assessor. Furthermore, thanks to Michael Moortgat for all the fruitful interactions, from informal chats to collaborations. A special thanks goes out in advance to Mark Steedman and Massimo Poesio for examining this manuscript and for (undoubtedly!) sharpening my pen.

From the academic community, I am grateful for interacting with (in no particular order): Glyn Morrill, Matt Purver, Julian Hough, Shalom Lappin, Ruth Kempson, Martha Lewis, Arash Eshghi, Bob Coecke, Stergios Chatzikyriakidis, Christine Howes, Giuseppe Greco, Aleksandre Maskharashvili, Konstantinos Kogkalidis, Adriana Correia, Vladislav Maraev, Vidya Somashekar, Bill Noble, Ross Duncan, Jules Hedges, Richard Moot, Alexander Kurz, Erlinde Meertens, and many others!

Thanks to all from the QMUL Computational Linguistics Lab, especially: Sophie Chesney and Alexandra Ume for jointly experiencing the trauma of assisting too many students with all their NLP assignments, and Carlos Ruiz Mendariz for teaching me about ELMo and BERT while enduring a vegan burger. Thanks also to all my fellow PhD students/postdocs in the Theory Group: Andrew Lewis-Smith, Thomas Cuvillier, Yu-Yang Lin, Emily Donovan, Jack Lik Hon Crawford, Arthur Passos de Rezende. A special thanks goes to Melissa Yeo for making sure that my PhD went as smoothly as possible, but also for taking me to the Buddhist Center. *Be water, my friend*. Moreover, I'd like to thank June Coster for the innumerous cups of coffee, and to Edward Hoskins for his cheerfulness and excellent mime.

To the following people, thank you for the music: Delia Fano Yela, Alessia Milo, Keunwoo Choi, Giulio Moro, Chris Harte and Peter Harrison, Shalom Lappin, Stergios Chatzikyriakidis, Rasmus Blanck, Simon Dobnik, Staffan Larsson, Andrew Lewis-Smith, and Julian Hough.

In addition, I'd like to thank a number of people in London that uplifted my time in London in one way or another. Thanks to Katherine Geier for being my first friend in London and taking me out of my university setting all those times. Thanks to Abed Qaddoumi for being my first new friend in London and being so much fun. Thanks to An Liang and Haiyang Zhong for all the photos, the hot pot, the plants.

Thanks to all others in London for all the pub nights, lunches, dinners, music shows, and birthday parties: Jonathan Young, Eddie Wade, Emmanouil Chourdakis, Beici Liang, Hazar Emre Tez, Marco Martinez Ramirez, Adán Benito Temprano, Gala de Vallejo Pérez, Adib Mehrabi, Will Wilkinson, Alo Allik, Florian Thalmann, Michael McLoughlin, Dan Stowell, Yvonne Blokland, Dave Ronan, Dave Moffat, Sophie Skach, Giulio Moro, Juliana Jaramillo.

Thanks to my friends in the Netherlands: Jeroen de Vos and Lily Knox, Ruben Dieleman, Sander Hetebrij, Rob Klabbers, Bram de Rijk, Annemiek Schellenbach and Iryn Bijker, Sanne Brinkhorst.

Thanks to Natalia Fano Yela, for welcoming and hosting us in Stockholm, and welcoming us back after an outing to the airport. Thanks to Fernando and Isabel for welcoming me in their home.

Finally, I wish to thank my mother, Margreet, and my father, Ger, for all their support. A thanks to my brother, Mart, for his affection, and for grinding my ears in London on several occasions.

And last, a big thanks to Delia: for all the avenues and adventures, for all the music and theatre, for all the food and the films, for all the laughter and for all the love.

This research was mainly supported by a QMUL Principal Studentship and the School of Electronic Engineering and Computer Science at Queen Mary University of London.

List of Publications

Below is the list of publications that were submitted and/or published during the course of this PhD.

[MW17] Michael Moortgat and Gijs Wijnholds. "Lexical and Derivational Meaning in Vector-Based Models of Relativisation". In: *Proceedings of the 21st Amsterdam Colloquium*. 2017.

[SMW19] Mehrnoosh Sadrzadeh, Michael Moortgat, and Gijs Wijnholds. "A Frobenius Algebraic Analysis for Parasitic Gaps". In: *Workshop on Semantic Spaces at the Intersection of NLP, Physics, and Cognitive Science*. 2019.

[Wij17] Gijs Jasper Wijnholds. "Coherent diagrammatic reasoning in compositional distributional semantics". In: *International Workshop on Logic, Language, Information, and Computation*. Springer. 2017, pp. 371–386.

[Wij19] Gijs Wijnholds. "A proof-theoretic approach to scope ambiguity in compositional vector space models". In: *Journal of Language Modelling* 6.2 (2019), pp. 261–286.

[WS18] Gijs Wijnholds and Mehrnoosh Sadrzadeh. "Classical Copying versus Quantum Entanglement in Natural Language: The Case of VP-ellipsis". In: *EPTCS 283, 2018, pp. 103-119* (2018), pp. 103–119. DOI: 10.4204/EPTCS.283.8.

[WS19a] Gijs Wijnholds and Mehrnoosh Sadrzadeh. "A Type-Driven Vector Semantics for Ellipsis with Anaphora using Lambek Calculus with Limited Contraction". In: *Journal of Logic, Language and Information* (2019).

[WS19b] Gijs Wijnholds and Mehrnoosh Sadrzadeh. "Evaluating Composition Models for Verb Phrase Elliptical Sentence Embeddings". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2019.

[WSC19] Gijs Wijnholds, Mehrnoosh Sadrzadeh, and Stephen Clark. "Representation Learning for Type-Driven Composition". In: *Preprint*. 2019.

Contents

Introduction

In order to make computers understand human language, they will need *efficient algorithms* that can process vast and diverse quantities of related linguistic data. Moreover, they will need good *computational representations* to make sense of that data. At first sight, it may seem that a well prepared, large, and rich dictionary may be enough for a computer to understand language. This would solve one major problem in the understanding of language, as such a dictionary would contain all the *words* of a language that the machine is trying to learn. But word meaning is not a static thing, and it is not intrinsic: new words appear and existing words change in meaning. When humans encounter new words they are able to make an educated guess about their meaning based on context. The question is how computers can learn to do the same thing.

Automatic word sense understanding and acquisition in context is one of the underlying principles of *distributional semantics*: words with similar meaning will occur in similar contexts. A major asset of this idea is that it allows for efficient implementation; numerous techniques exist for representing words *in context*, effectively encoding words as vectors in such a way that the distributional principle is preserved under some similarity metric [Mik+13; LG14; PSM14; Boj+17]. This subsequently allows for computer systems to use these *word embeddings* to address a natural language processing task such as question answering, paraphrase detection, or natural language inference.

But language is also *compositional*: although any dictionary of any language spoken on Earth will necessarily be finite, humans are capable of understanding phrases or sentences that they have never seen or heard before. Research in formal linguistics has delivered mathematically elegant models of compositional semantics, where the meaning of a sentence is computed as a transformation of syntactic structure, applied to the individual, denotational semantics of constituent words [Mon70a]. The presence of such mathematically rigorous models of compositional semantics has led to the main question underlying *compositional distributional semantics*: if we have efficient numerical encodings of word meaning, how can we compose these into meaningful numerical representations of larger units of text?

Over the last ten years, a number of different approaches arose that tried to answer this question. Formal approaches to distributional semantics aim to transfer

the compositionality principle from formal semantics to the world of word embeddings. Specifically, given a sequence of words $w_1...w_n$, an interpretation σ of the individual words as embeddings, and an associated grammatical structure G, a *formal compositional distributional semantic* model defines a transformation F that derives the meaning of the word sequence as a mapping from syntax to a compositional semantics:

$$F(G)\big(\sigma(w_1) \ ... \ \sigma(w_n)\big)$$

An example of such a model comes from the work of Coecke, Sadrzadeh, and Clark [CSC10] and Coecke, Grefenstette, and Sadrzadeh [CGS13], who instantiate a model using a *type-driven* approach. Words are assigned *types* and a grammar logic assigns syntactic structure to sequences of words, after which a homomorphic passage maps the syntactic structure of a sentence to (multi-)linear maps that act on the constituent word representations. Not all type-driven grammar are logics; for example, Combinatory Categorial Grammar [Ste00] systems are typically presented as calculi containing rules for changing or combining types. However, in the framework of Coecke, Sadrzadeh, and Clark [CSC10] and Coecke, Grefenstette, and Sadrzadeh [CGS13] category theory is used as the underlying tool to unify mathematical structures, where it is paramount to be able to show that the grammar is both sound and complete with respect to a particular category.

The formal approach is in contrast with neural compositional approaches. These approaches define a sentence embedding as the result of training a neural network on a specific task that computes a vector representation from the vector representation of the words in the sentence. In some cases this representation is a weighted sum of the word vectors of a sentence [ALM19; Cer+18]:

$$N(w_1...w_n) = \alpha_1\overrightarrow{w_1} + ... + \alpha_n\overrightarrow{w_n}$$

where the weights α_i are the weights of a neural network trained using machine learning techniques.

The formal and the neural viewpoint come with their own merits and drawbacks: neural approaches easily generalise to arbitrary sentences, paragraphs, and text, at the drawback of limited *interpretability* of the models and limited *explanatory power*. On the other hand, the formal approach explicitly starts from linguistic theory — thereby providing an explanation about the compositional process — but it imposes syntactic structure on the embedding of sentences, which comes at the cost of extra processing requirements and limited scalability.

In this thesis, we are interested in studying the relevance of linguistic knowledge for sentence embeddings, and therefore we focus on the theoretical and experimental aspects of a type-driven framework in the style of Coecke, Grefenstette, and Sadrzadeh [CGS13]. In general we are interested in discourse, which is a large domain, therefore the particular phenomenon we focus on is *ellipsis* (with anaphora). Informally,

we speak of ellipsis when part of the meaning of a phrase is missing from its syntactic realisation. In many cases, such as verb phrase ellipsis, this missing semantic element can actually be recovered from syntactic context, making such cases suitable for a formal modelling in a compositional distributional setting. Consider for example the phrase "Hannah sings and Ben does too"; it is clear that what Ben is doing involves his vocal chords, though this is not explicitly given. Things get even worse when we introduce *anaphora*, as in "Hannah sings her song and Ben does too": now, besides from the verb phrase being implicit, we are not even sure what song Ben is singing.

Ellipsis forms an interesting test case for compositional distributional semantics for two main reasons. First, it violates the assumption that a one on one relation exists between the meaning of a sentence and its physical realisation, an assumption that typically underlies both neural models as well as the categorical framework of Coecke, Grefenstette, and Sadrzadeh [CGS13]. Secondly, from an experimental perspective ellipsis is interesting because it allows us to contrast models that perform the necessary analysis to resolve ellipsis (i.e. recover the implicit semantic information) with models that take a 'what you see is what you get' approach. For example, rather than simply adding the vectors for the words in "Hannah sings and Ben does too", we may instead do this for the vectors of the resolved sentence "Hannah sings and Ben does too". The same holds for neural approaches to composition, which can moreover be compared with type-driven models.

The use of Frobenius Algebras in the categorical framework has been shown in modelling of pronoun relativisation and coordination [SCC13; SCC14; Kar16; MW17], where syntactic material needs to be *moved* into a different position in a sentence in order to make sense of its meaning. The role of Frobenius Algebras is to model function words like relative pronouns and coordinators, allowing their interpretations to be able to merge information in a way akin to the use of set intersection in formal semantics. The same approach approach has also been applied to the case of quantifiers [HS19; Sad16; Wij19]. However, this theoretical work is still limited to cases where all semantic information is explicitly given. Thus, the theoretical contribution of this thesis is to extend the categorical framework in a principled and general way to deal with ellipsis, anaphora, and to other phenomena related to wh-movement, such as parasitic gapping. We do this by defining a novel compositional distributional model, based on a limited form of contraction as an addition to the Lambek Calculus of Lambek [Lam58] to allow for the controlled recovery of implicit semantic information.

Experimental work evaluating compositional distributional models of meaning has considered sentence-level tasks that test the disambiguation of verbs in context and sentence similarity [GS15; KS13; KS14; Mil+14]. Here, the issue of scalability really shines through, as the *datasets* involved in this evaluation are limited to transitive sentences of the form *subject verb object* and a variant in which subjects and objects are modified by adjectival phrases. Our contribution is the introduction of

three new datasets that serve both as test beds for distributional models of verb phrase ellipsis, as well as being larger than the beforementioned datasets, both in the number of datapoints and the length of the sentences that are modelled. For example, one of the introduced datasets contains phrases of the form *subject verb object and subject* does too*, and has twice as many sentence pairs as the transitive sentence dataset it is based on.

Finally, one main assumption behind the tensor-based modelling is that words be represented as tensors of various orders, depending on their grammatical function. For example, where nouns are represented as vectors, adjectives are considered to be matrices. Such an adjective matrix is the physical representation of a linear map transforming a noun vector into a vector representing the compound adjective-noun combination. This idea extends to the case of transitive verbs, that get a cube representation, which function now as bilinear maps that transform both a subject and an object vector to a vector giving their compound representation. The assumption of a type-driven tensorial word representation is one of the technical distinctions between the formal and the neural approaches. Although Milajevs et al. [Mil+14] experiments with prescribed formulas for creating verb matrices, and previous work has been done to learn the content of adjective and verb representations [BZ10; Gre+13; PRC14; MC15], these approaches do not all form a general model for learning *any word with any grammatical role*, and most approaches suffer from data sparsity issues.

Thus, our third and final contribution in this thesis is to reformulate the well known skipgram model of Mikolov et al. [Mik+13] to a general tensor-based setting. Although this was partly investigated by Maillard and Clark [MC15] for the case of adjectives, we discovered that a different notion of context is needed to treat arbitrary words, and so we generalise the model accordingly. We then give an implementation of several different representations for the case of verbs. Our results show that correlation with human judgments improves over previous approaches when the proposed method is used.

This thesis is structured as follows: in Chapter 1 we start off with a discussion of the general background of compositional distributional semantics, from the viewpoint of state of the art deep neural network techniques as well as linguistics oriented formal approaches to composition of distributional word representations. Then, in Chapter 2 we zoom in on the categorical framework of Coecke, Grefenstette, and Sadrzadeh [CGS13], and define a functorial passage from the syntax of the Lambek Calculus to the semantics of vector spaces and linear maps. We also briefly discuss the concept of diagrammatic reasoning, as this simplifies many of the mathematics underlying the framework.

Then come the main contributions of the thesis. First, we discuss the theoretical motivations of a compositional vector space model for ellipsis and anaphora in

Chapter 3, where we investigate the extent to which we can encapsulate the non-linearity of ellipsis in the lexical semantics of the models, to move on to a general setting in which the reuse of information is allowed in a structured way. Chapter 4 then introduces our second contribution, which is the introduction of three new datasets that allow us to test a variety of different concrete composition models for ellipsis, and a comprehensive evaluation of such models. Then we move to the separate albeit related issue of learning tensor representations for words in Chapter 5. We define a general machine learning architecture for learning dependency-based tensor representations for words, that then can be used in a type-driven composition model. We implement this architecture for the case of transitive verbs and show how the proposed representation achieves new state of the art results on the datasets that have been introduced to test formal compositional distributional models.

Part I

Background

Chapter 1

Distributional Semantics: From Word to Sentence

Chapter Abstract

In this chapter, we discuss compositional distributional models of meaning. We start with an overview of distributional semantics on the word-level and its commonly used implementations, then we continue to define compositional models, and their amply different implementations. We conclude by situating this thesis in the context of this highly active field of research.

Distributional semantics is a field of research within computational linguistics that provides an easily implementable algorithm with an empirically verifiable output for representing word meanings and degrees of semantic similarity thereof. This semantics is rooted in the distributional hypothesis, investigated by Harris [Har54], and popularised by Firth [Fir57] using the phrase “a word is characterized by the company it keeps". More precisely, according to the distributional hypothesis, “words that occur in the same contexts tend to have similar meanings” [Pan05].

This idea has been made concrete by gathering the co-occurrence statistics of context and target words in large text corpora and using that as a basis for developing vector representations for word meanings. A notion of similarity based on the cosine of the angle between vectors allows one to compare degrees of word similarity in the vector space models where these word vectors embed. Such models have been shown to perform well in a variety of natural language processing (NLP) tasks, such as semantic priming [LB96] and word sense disambiguation [Sch98]. The underlying philosophy has gained attention in cognitive science as well [Len08].

A lot of work has been done to improve the basic co-occurrence based word embeddings, ranging from using machine learning to predict a word’s context rather than directly computing it [Mik+13], optimising for ratios of co-occurrences rather than the co-occurrences themselves [PSM14], and encoding morphological information below the word level [Boj+17].

1.1 Word Embeddings: Implementing the Distributional Hypothesis

1.1.1 Count-Based Word Embeddings

One way of approximation a word's context is in a count-based model. Here, the context is taken to be a linear window and the corpus is traversed to collect raw co-occurrence counts as depicted below for the focus word w_i, context window of 3:

$$w_1 \ldots \underbrace{w_{i-3} w_{i-2} w_{i-1} w_i w_{i+1} w_{i+2} w_{i+3}}_{\text{context window}} \ldots w_n$$

Then, a weighting scheme is applied to smooth the raw frequencies in the meaning representation. Examples include point-wise mutual information (PMI) and its positive variant (PPMI), and local mutual information (LMI). More discussion on count-based vector space models can be found in Turney and Pantel [TP10], and a systematic study of the parameters of count-based word embeddings is given by Kiela and Clark [KC14].

1.1.2 Neural Word Embeddings

Skipgram The skipgram model with negative sampling [Mik+13] generates word embeddings by optimising a logistic regression objective in which target vectors should have high inner product with context vectors for positive contexts (as observed from a linear context window around a target word), and low inner product with negatively sampled contexts. Given a target word n and a set of positive contexts C, a set of negative contexts $\overline{C}$ is sampled from a unigram distribution raised to some power (often $3/4$, after [LGD15]). The number k of negative contexts per positive one is a parameter of the model, and typically ranges between 5 and 20. Context words are subsampled to decrease the difference between very frequent and very infrequent words. Initially, both target vectors $\mathbf{n}$ and context vectors $\mathbf{c}$ are randomly intialised, and during training the model updates both target and context vectors to maximise the objective function

$$\sum_{c \in C} \log \sigma(\mathbf{n} \cdot \mathbf{c}) + \sum_{\overline{c} \in \overline{C}} \log \sigma(-\mathbf{n} \cdot \overline{\mathbf{c}}) \tag{1.1}$$

which formalises the desired properties of the resulting vectors. These vectors are sometimes also referred to as `word2vec`, after the software package that implements the skipgram model[1].

Dependency Skipgram The count-based and skipgram models above rely on a *linear context window* to provide contextual information: setting aside the subsampling

[1] `github.com/tmikolov/word2vec`

done for skipgram, in both models the context is given by a certain number of context words around a focus word. Levy and Goldberg [LG14] propose a dependency-based context: given a target word w with modifiers $m_1 ... m_n$ and head h, the context of w is given as $(m_1, l_1), ...(m_n, l_n), (h, l_h^{-1})$ where l_i is a dependency label associated with a modifier, l_h^{-1} encodes the inverse dependency of the target w on the head h. These word-dependency pairs now form the context, in the exact same skipgram objective as above.

GLoVe The intuition behind GloVe [PSM14] is that co-occurrence information is needed to learn word embeddings, but that the *ratio* of co-occurrence probabilities is in fact more informative of word similarity. The GloVe model therefore learns word embeddings by minimising the least-squares objective between the dot product of two word embeddings and the log-probability of the words' co-occurrences. Writing $X_{i,j}$ for the co-occurrence probability of words i, j, and $\overrightarrow{w_i}$ for the associated word vector, the GloVe learning model minimises the function

$$\sum_{i,j=1}^{V} f(X_{i,j}) \left(\overrightarrow{w_i}^T \overrightarrow{w_j} + b_i + b_j - \log X_{i,j} \right)^2$$

where $f(X_{i,j})$ is a smoothing function[2].

FastText The FastText vectors [Boj+17] use again the skipgram model described above Word2Vec, but now considers *subword* information. Rather than training vectors for words, vectors are trained for n-grams of characters. The final word vector will then be the sum of its constituent n-gram vectors. The idea is that the incorporation of subword information may improve the quality of the embeddings, in particular for morphologically rich languages.

1.1.3 Evaluating Word Embeddings

As word embeddings provide concrete implementations of the distributional hypothesis, and with different implementations available, whether one type of embedding is 'better' than another depends on the type of task they are used for. To evaluate the quality of embeddings a notion of similarity needs to be defined, and human similarity judgments need to be available.

The most commonly used similarity measure is cosine similarity, which measures the angle between two vectors in a space:

$$\cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}||\mathbf{b}|}$$

[2] In the original paper, the authors discuss using an alternative vector $\tilde{w_j}$, which may reduce overfitting.

Values range between -1 and 1, so usually these are rescaled to give a value between 0 and 1, 0 meaning full dissimilarity and 1 meaning full similarity.

Given a vector space of word embeddings, we can now compute similarity results for any pair of words for which embeddings are available. In order to ground these similarities in human cognition, a way to evaluate the quality of these embeddings is to measure the correlation between model similarity and an aggregate of human similarity judgments. A standard correlation measure here is the Spearman correlation coefficient ρ, which expresses whether a monotonic relation exists between the computed similarities of the model and the (average) similarity judgments made by humans.

One of the first such datasets was created by Rubenstein and Goodenough [RG65], which contains 65 word pairs that were rated by 15 annotators for similarity, though the aim was to capture synonymity of words. The MC30 dataset [MC91] contains 30 noun pairs, that were supposed to capture semantic similarity. A much larger dataset is WordSim-353 [Fin+01], containing 353 noun pairs. A specialised dataset for verbs is VerbSim [YP06], that contains similarity judgments for 130 pairs of verbs. Thanks to crowdsourcing tools, even larger datasets could be constructed: a recent large similarity datasets is MEN [Bru+12], a 3000 word pair dataset containing nouns, adjectives, and verbs. In addition, SimLex-999 contains 999 pairs of nouns, adjectives and verbs. Finally, in 2016, the SimVerb dataset was developed, which contains 3500 verb pairs. Many other word similarity datasets exist, but we list here the ones that are relevant for the purposes of this thesis:

Name	Reference	# of Pairs	Categories
RG	[RG65]	65	nouns
MC30	[MC91]	30	nouns
WordSim353	[Fin+01]	353	nouns
VerbSim	[YP06]	130	verbs
MEN	[Bru+12]	3000	nouns, adjectives, verbs
SimLex	[HRK15]	999	nouns, adjectives, verbs
SimVerb	[Ger+16]	3500	verbs

1.2 What is Compositionality?

Although the notion of similarity is intuitive and works well at the word level, it is less productive to consider phrases and full sentences to be similar whenever they occur in a similar context. Firstly, we know that language is compositional, since the number of potential sentences humans can produce are larger than the amount a single human ever produces. Secondly, data sparsity issues arise when treating sentences as individual expressions and computing direct co-occurrence statistics for them. So the non-trivial task of producing vector representations for phrases and sentences is the main challenge of compositional distributional semantics.

Current approaches to composition of word embeddings can be roughly be classified as algebraic, neural, and formal. The algebraic approach defines a vectorial representation of arbitrary text as the result of summing or multiplying the individual word vectors; the idea of adding vectors together for composition was already worked out by Kintsch [Kin01] in the context of Latent Semantic Analysis [LD97]. Mitchell and Lapata [ML08; ML10] introduce multiple novel similarity datasets based on aggregate human similarity judgments for two-word combinations and experiment with additive and multiplicative models, as well as a dilation model. An example of the algebraic approach to general sentence embeddings is the work of [ALM19], that effectively computes a compressed weighted average of the individual word embeddings. The neural approach relies heavily on machine learning to learn vectorial representations for sentences, or to learn a generic sentence encoder that will embed any sentence. The approach of [Soc+13] uses a Recursive Neural Network to learn to classify the sentiment of a sentence, by computing the composition function that composes word vectors together into embeddings for larger phrases. More recent approaches use various deep neural networks to learn general sentence encoders [Con+17; Cer+18].

Both the algebraic and neural models are very general, in that they can create embeddings for arbitrary sentences. Unfortunately, they are mostly not very concerned with linguistic/grammatical information. Moreover, they often rely on task-specific data for training.

A structured attempt at providing a formal model of compositional distributional semantics has been presented in [CSC10]; these models start from the observation that vector spaces share the same structure as Lambek's most recent grammar formalism, pregroup grammar [Lam99], and interpret its derivations in terms of vector spaces and linear maps. What follows is an architecture that is familiar from logical formal semantics in Montague style [Mon70a], where the judgments of a grammar translate to a consistent semantic operation (read linear map) that acts on the individual word vectors to produce some vector in the sentence space. A number of subsequent investigations have shown that a similar interpretation is possible for other typelogical grammars, such as Lambek's original syntactic calculus [CGS13], Lambek-Grishin grammars [Wij14], and Combinatory Categorial Grammar [MCG14].

These formal models have a clear linguistic motivation, in that they follow directly the grammatical structure of sentences, but at the expense of not being easily generalisable.

One major issue for distributional semantics and especially compositional approaches therein is to find a suitable representation for function words. Without the power of formal semantics to assign constant meanings or to allow set-theoretic operations, distributional semantics does not have much to say about the meaning of logical words such as 'and', 'despite' and pronouns like 'his', 'which', 'that', let alone quantificational constituents ('all', 'some', 'more than half'). All of these words

have in common that they intuitively do not bear a contextual meaning: a function word may co-occur with any content word and so its distribution does not reveal much about its meaning, unless perhaps the notion of meaning is taken to be conversational[3]. To overcome this issue, Sadrzadeh, Clark, and Coecke [SCC13] rely on Frobenius algebras[4] to formalise the notion of *combining* and *dispatching* of information. This approach has seen applications to relative pronouns [SCC13; SCC14], coordination [Kar16], and to a lesser extent to some limited forms of ellipsis [KPS16]. In each of these, the Frobenius algebras allow one to use element wise multiplication of arbitrary tensors, corresponding to the usual intersective interpretation one finds in formal semantics [DSP91]. A treatment of quantification was also given using the bialgebraic nature of vector spaces over powersets of elements [HS19; Sad16].

In the next sections, we review the type-driven and the neural approach.

1.3 Type-Driven Approaches to Composition

In its most general form, a type-driven model of composition in distributional semantics relies on the assignment of (some form of) *types* to words that specify, in conjunction with some (form of) grammar, how words are to be composed into greater structures. An interpretation of these two ingredients then leads to a definition of semantic composition:

Definition 1 *Given a sequence of words $w_1 ... w_n$ with associated types $t_1 ... t_n$, some grammatical structure G, and an type-respecting interpretation of words σ and an interpretation of grammatical structures F, the compositional meaning of $w_1 ... w_n$ will be*

$$F(G)\big(\sigma(w_1) \ ... \ \sigma(w_n)\big)$$

This view is taken in the work of Montague [Mon70a; Mon73], who formulates a type-theoretic interpretation where words are to be interpreted as set-theoretic constructions, though the principle of composition stays the same, whether we work with sets or vector spaces.

Taking the type-driven view means having a very general framework for compositional distributional semantics, which comes with certain assumptions. A schematic of the framework is displayed in Figure 1.1. First, one has to commit to some form of syntax, which comprises the types and grammatical structures of the definition above. The types and grammar put constraints on the semantic representation of words, and of derivational structures, as type and grammar interpretations σ and F have to interpret types and structures respectively. The final phrase representation thus depends on (a) types, (b) grammatical structure, and (c) their respective interpretation.

[3] The work of Kruszewski et al. [Kru+16] gives a distributional semantic account of conversational negation.

[4] A notion we will introduce formally in Chapter 2.

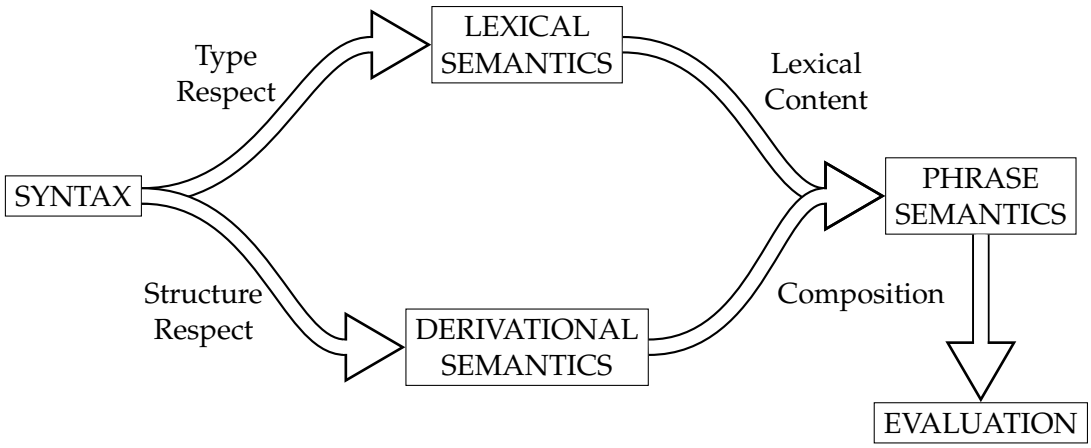


FIGURE 1.1: The general approach of type-driven compositional distributional semantics. Grammar puts constraints on the structure of lexical items, and the order of composition. Derivational and lexical semantics together lead to concrete phrase semantics which then are evaluated on a variety of tasks (e.g. sentence similarity, disambiguation, paraphrasing).

Algebraic Models A very simple way of implementing a type-driven compositional distributional semantics is to use an algebraic model: here, one assumes that the *lexical content* of any word is given by the same type of distributional representation, i.e. a vector, and that a single linear algebraic operation interprets the function of the grammatical structure on the meaning of a phrase. For example, an *additive* model assigns meaning to a sentence by element wise sum of all the vectors of the words in that sentence. Similarly, one defines a multiplicative model by taking element wise multiplication. Another option is a weighted sum. We list these models in Table 1.1, and refer to Mitchell and Lapata [ML08; ML10] for the first experimental studies on algebraic models of composition in a distributional setting.

Model	**Sentence**	**Embedding**
Additive	$w_1...w_n$	$\overrightarrow{w_1} + ... + \overrightarrow{w_n}$
Multiplicative	$w_1...w_n$	$\overrightarrow{w_1} \odot ... \odot \overrightarrow{w_n}$
Weighted Sum	$w_1...w_n$	$\alpha_1\overrightarrow{w_1} + ... + \alpha_n\overrightarrow{w_n}$

TABLE 1.1: Algebraic models of composition for distributional semantics.

Tensor-Based Models A next step in implementing a type-driven models comes from the tensor-based approach [CSC10; BZ10; PB+14]: here, one assumes that the *type* or grammatical category of a word dictates its tensor rank, promoting words with complex types to multi-linear maps, analogous to a set-theoretic semantics where words with complex types are assigned to functions in a function space given by their type. Then, composition becomes tensor contraction, which is simply the application of a multi-linear map to its arguments. For example, a transitive verb

may be assigned the syntactic type $(np\backslash s)/np$, meaning it composes with the object of the verb, and afterwards with the subject of the verb to form a sentence. Its semantic representation then will be a cube $\overline{\overline{V}}$ which may be applied to any pair of subject and object vectors using tensor contraction, to give back a vector. If we write down the entries of $\overline{\overline{V}}$ as V_{ijk} and the entries of a subject vector $\overrightarrow{s}$ as s_l and those for an object vector $\overrightarrow{o}$ as o_m, then we write for the application:

$$\overrightarrow{s}^T \times \overline{\overline{V}} \times \overrightarrow{o} = \sum_{ijk} s_i V_{ijk} o_k$$

Here, the unification of indices shows that we are multiplying those elements in the first dimension of the cube with those of the subject vector in an element wise fashion, and the same for the object vector, which leaves us with a vector for the subject-verb-object combination. In concrete experiments, this particular way of composing tensors is only one of the implementations, and we will call any method that uses different rank tensors for its individual word representations a tensor-based model. For example, Milajevs et al. [Mil+14] consolidates and evaluates a number of composition models that were tested before in the literature on tensor-based models [GS11a; Gre+13; GS15; KS13; KSP13; KS14], which we list in Table 1.2. The crucial extra operation used here is the outer product between vectors. Given vectors $\overrightarrow{a}$ with entries a_i and $\overrightarrow{b}$ with entries b_j, their outer product will be given by

$$\overrightarrow{a} \otimes \overrightarrow{b} = \sum_{ij} a_i b_j$$

That is, from such two vectors we create a matrix with entries given by the possible ways of multiplying one element from $\overrightarrow{a}$ with one element from $\overrightarrow{b}$. This operation generalised to higher-order tensors, which we will discuss in further detail in Chapter 2. Interestingly, all of these tensor-based models assume a transitive verb to be represented by a matrix, which is both a theoretical simplification of the pure tensor-based setting (as the sentence space is neglected), as well as a practical compromise as this representation leads to impressive experimental results.

1.3.1 Learning the Content of Word Tensors

Taking on a type-driven view amounts to saying that the grammatical type of a word has a direct influence on its semantic representation, so that the latter is a combination of lexical semantics and syntactic information, more specifically that the representation of a sentence ought to follow the building blocks of sentence structure. In a tensor-based model, word representations can be arbitrary rank tensors. As we cover a tensor learning architecture and its evaluation in this thesis we review here some of the related work on learning tensor representations for words.

Name	Acronym	Formula
Relational	**REL**	$\overrightarrow{verb} \odot (\overrightarrow{subj} \otimes \overrightarrow{obj})$
Kronecker	**KRON**	$(\overrightarrow{verb} \otimes \overrightarrow{verb}) \odot (\overrightarrow{subj} \otimes \overrightarrow{obj})$
Copy Subject	**CS**	$\overrightarrow{subj} \odot (\overrightarrow{verb} \times \overrightarrow{obj})$
Copy Object	**CO**	$\overrightarrow{obj} \odot (\overrightarrow{verb}^T \times \overrightarrow{subj})$
Frobenius Additive	**FA**	$\overrightarrow{subj} \odot (\overrightarrow{verb} \times \overrightarrow{obj}) + \overrightarrow{obj} \odot (\overrightarrow{verb}^T \times \overrightarrow{subj})$
Frobenius Multiplicative	**FM**	$\overrightarrow{subj} \odot (\overrightarrow{verb} \times \overrightarrow{obj}) \odot \overrightarrow{obj} \odot (\overrightarrow{verb}^T \times \overrightarrow{subj})$
Frobenius Outer	**FO**	$\overrightarrow{subj} \odot (\overrightarrow{verb} \times \overrightarrow{obj}) \otimes \overrightarrow{obj} \odot (\overrightarrow{verb}^T \times \overrightarrow{subj})$

TABLE 1.2: Table of all tensor-based composition models for transitive sentences (*subj verb obj*) that were evaluated by Milajevs et al. [Mil+14].

Although there are quite a few established methods around that learn distributional and distributed vector embeddings at the word level, the number of techniques that are used to obtain high-rank tensors are limited.

Adjectives For adjectives, matrices can be learned by regression to approximate holistic adjective-noun phrase vectors [BZ10], and more recently, by lifting the skip-gram model of Mikolov et al. [Mik+13] to learn a transformation between fixed vectors for nouns and adjective-noun combinations [MC15].

Verbs For verbs, the relational model of Grefenstette and Sadrzadeh [GS11a] takes the sum of the outer products of the vectors for observed subjects and objects of a given transitive verb. Later work follows the learning method of Baroni et al. and applies a multi-step regression algorithm to learn a verb cube by first approximating a holistic subject-verb vector, and subsequently a holistic verb-object matrix [Gre+13]. A different approach is the plausibility model of Polajnar, Rimell, and Clark [PRC14], in which a verb matrix or cube is learnt to optimise a model that distinguishes between observed subject-verb-object triples (plausible) and generated triples (implausible).

All in all, the status of type-driven compositional distributional models seems to be this: these models derive their strength from an elegant theory, and the presence of an explanation of how the meaning of sentences should be derived. On the other hand, there is a limited number of efficient techniques for estimating the content of the tensors involved, resulting in a lack of scalability.

1.4 Neural Approaches to Composition

An approach that is seemingly orthogonal to that of type-driven composition, is the purely neural approach. In such a scenario, one treats again the type of word representations to be fixed, assigning a vector to each word. Then, an objective function is optimised that maps multiple word vectors onto a sentence vector. Such models are sentence encoders. A different style of models learns to assign vectors to words, depending on the other words they occur with in a sentence, using a language modelling approach. Here, context is incorporated both in the vector representations themselves, as a language model computes the probability of a word occurring given the previously encountered words, and in the overarching sentence representation: the particular vector assigned to each word is a function from all other words in the sentence. We review both approaches here.

1.4.1 Sentence Encoders

Skip-Thoughts The skip-thoughts models of Kiros et al. [Kir+15] uses an encoder-decoder model to learn arbitrary sentence representations in an unsupervised manner. The model essentially lifts the idea behind most word embeddings to the sentence level: given a sentence, its context is given by preceding and following sentences.

The skip-thoughts model consists of three components: a sentence encoder (a recurrent neural network) for a target sentence, and two *conditional* decoders, one for decoding the previous sentence and one for decoding the next sentence. Given a triple of sentences (s_{i-1}, s_i, s_{i+1}), the encoder produces a hidden state vector $\mathbf{v}_i$ for the target sentence, and the two decoders use this as a condition for trying to generate the previous and following sentences s_{i-1}, s_{i+1}. The decoders are essentially language models, as for each timestep, they are given the actual (previous) words in the sentence that they are trying to produce. For example, if the target sentence is *I could see the cat on the steps* with representation $\mathbf{v}$, and the previous sentence would be *I got back home*, then at the second timestep the backward decoder is trying to predict the word *back*, given $\mathbf{v}$ *and* the sequence *I got*.

From a high-level perspective, the skip-thoughts model lifts the distributional hypothesis to the level of sentences: sentences that co-occur with similar sentences tend to have similar meaning.

InferSent This model learns sentence representations using a supervised natural language inference task. Essentially, the proposal of the authors [Con+17] is to learn a general encoder for arbitrary sentences, by connecting the sentence encoding vector to a three-way classification problem on a natural language inference task.

Although the authors compare various different deep neural network models, their published model is based on a bi-directional LSTM with max pooling. The overall architecture is highlighted in Figure 1.2. The general purpose encoder is

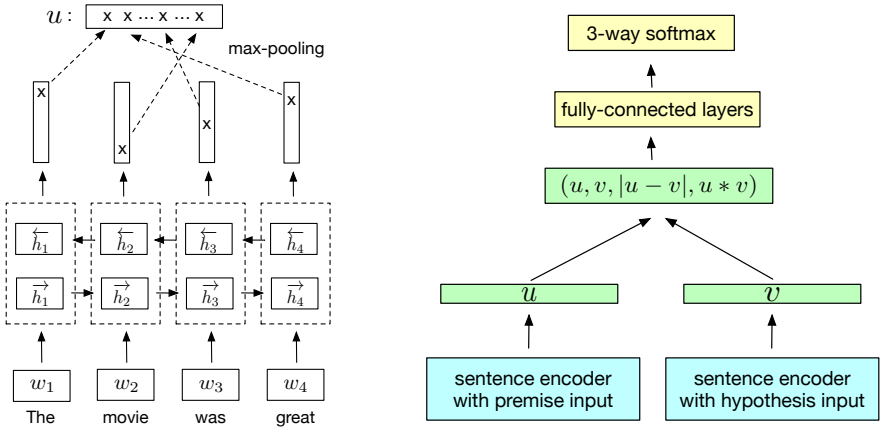


FIGURE 1.2: Figures from [Con+17] explaining the system's architecture: the sentence encoder uses a bidirectional LSTM, and applies a max pooling attention mechanism (left). The supervised task is natural language inference: the encoder gives a vector for premise and for hypothesis, and their concatenation, absolute element wise difference, and element wise multiplication are passed through a feed-forward three-way classifier (contradiction, neutral, entailment).

trained on the Stanford Natural Language Inference dataset of Bowman et al. [Bow+15], but is consequently evaluated on multiple tasks.

Universal Sentence Encoder A multi-task learning version of InferSent is Google's Universal Sentence Encoder [Cer+18], which uses two different model architectures (Transformer and Deep Averaging Network) and trains both a Skip-Thought objective, and the SNLI supervised task, in addition to a number of additional classification tasks, such as sentiment analysis and semantic textual similarity.

The Transformer model was introduced in [Vas+17], and uses static input with an *attention* mechanism rather than the dynamic recursive neural networks, as displayed in Figure 1.3. The attention mechanism takes a word vector (the query) as input and the vector representations for all other words in the sentence (the keys and the values) and produces an output vector that represents the relevant contextual information for the given word in the sentence. In this way, there is no need to rely on a recursive mechanism in the way an LSTM would do.

Deep averaging networks [Iyy+15] were introduced and shown to outperform neural network models that explicitly try to model compositionality (e.g. by recursing over a tree). The architecture is fairly straightforward and displayed in Figure 1.4: first, the embeddings for the words in a sentence are averaged, the result of which is passed through two non-linear feedforward layers, and finally a softmax classification is applied.

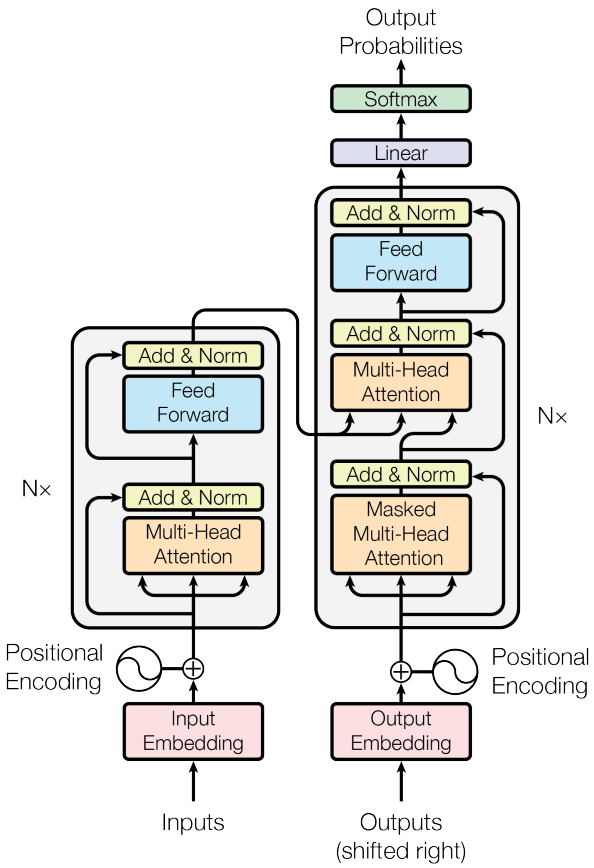


FIGURE 1.3: Figure from [Vas+17], explaining the attention mechanism used.

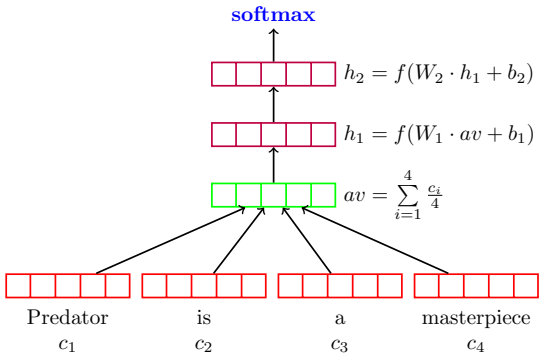


FIGURE 1.4: Figure from [Iyy+15], explaining a deep averaging network architecture. The input vectors for all words are linearly averaged, after which two extra layers manipulate this average by means of the softmax classification objective of the network.

1.4.2 Contextualised Embeddings

Contextualised embeddings provide a middle ground between word embeddings and sentence encodings: given a word in a sentence, these models give back an embedding that *depends* on the context of all other words in the sentence. By merging the different embeddings for the words in the sentence, one can then give back a sentence vector for further processing. We highlight two popular models, that are based on the task of *language modelling*.

ELMo The "Embeddings from Language Models" from Peters et al. [Pet+18] works in two steps: first, there is an unsupervised stage where a bidirectional neural language model based on LSTMs is trained on large amounts of textual data, using a multi-layer LSTM. The second step is task-specific and learns how to combine the internal layers of the model to provide embeddings that perform well on the given task.

A language model tries to predict the next word in a sequence based on previous words, computing the probability

$$p(w_k|w_1, ..., w_{k-1})$$

For a full sequence, the probability of that sequences is given by the product of intermediate steps:

$$p(w_1, ..., w_n) = \prod_{k=1}^{n} p(w_k|w_1, ..., w_{k-1})$$

The language model that ELMo uses is bidirectional: while one LSTM models a forward language model, another LSTM models the backward language model, the two sharing the initial embeddings for each word (input layer) and the softmax classifier that predict the next (or previous) word (output layer). The schema is laid out in Figure 1.5.

The supervised training then is based on the hidden layers of the biLSTM. As Figure 1.5 shows, the model has two hidden layers for each LSTM. Given some training task (say a classification task), one uses the pretrained ELMo model, but learns a linear combination of the hidden vector representations of the words in a sequence to output a final word vector representation, that is then used in the task at hand.

BERT The "Bidirectional Encoder Representation from Transformers" of Devlin et al. [Dev+19] also use a language modelling approach, just like ELMo, but base themselves on the Transformer architecture of Vaswani et al. [Vas+17], in which not a recursive neural network is used, but rather a sequence to sequence model with a self-attention mechanism. The major issue the authors addressed with BERT was the problem of jointly learning left-to-right and right-to-left encoders, as doing so allows the model to predict the missing word using the attention mechanism itself. So

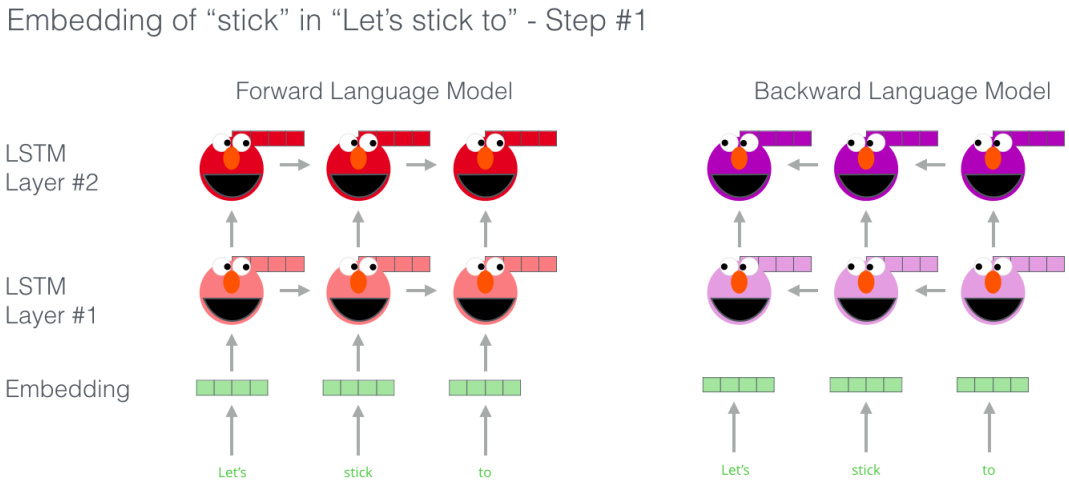


FIGURE 1.5: Figure from [Pet+18], explaining the ELMo pretraining step. Where the forward language model learns to predict the embedding for the next word given the previous word embedding, the backward language model works the other way around. After pretraining, the model gives three layers of both forward and backward looking embeddings, that need to be mixed into a sentence embedding.

the authors replace the language modelling object by a “masked language model", in which an arbitrary word of a sequence is masked and the model needs to predict it. The model can be fine-tuned on a task by additional training, this time replacing the input and output with task-specific training instances.

1.5 Evaluating Sentence Embeddings

Because sentence embeddings are relevant for many natural language processing applications, there are a number of different types of datasets for the evaluation of such embeddings. On the one hand, there are large-scale datasets that try to address general *textual similarity* or *natural language inference*. Examples of textual similarity are the Semantic Textual Similarity datasets (STS), collected over the course of five years during the shared tasks of the *SEM conferences between 2012 and 2016, and the SICK dataset introduced by Marelli et al. [Mar+14]. For natural language inference, the most prominent datasets are the Stanford Natural Language Inference dataset [Bow+15], with its multigenre variant MNLI [WNB18] and its crosslingual version XNLI [Con+18].

Although these datasets are used to test many state of the art models, they are often not focussed enough to properly assess the compositional approaches that we are interested in. Moreover, given that they may contain general arbitrary length sentences makes it difficult to evaluate the effect of particular composition choices. Finally, it is not yet clear to what extent the datasets facilitate the development of properly generalising models [TC19].

The datasets that we focus on in this thesis come from the work of Mitchell and Lapata [ML08; ML10], and Sadrzadeh and colleagues [GS11a; KSP13; KS13], describing intransitive sentences and transitive sentence, respectively. As we define a compositional vector space model of ellipsis with anaphora, we also introduce extended datasets with verb phrase elliptical sentences [WS19a; WS19b]. We list all the datasets that we use in this thesis in the table below:

Name	Reference	# of Pairs	Categories
ML2008	[ML08]	200	SV
ML2010	[ML10]	200	SV & VO
GS2011	[GS11a]	200	SVO
KS2013	[KSP13]	100	SVO
KS2014	[KS13]	108	SVO
MLELLDIS	[WS19a]	240	VP-Elliptical SV
ELLDIS	[WS19b]	400	VP-Elliptical SVO
ELLSIM	[WS19b]	432	VP-Elliptical SVO

Neural models For evaluation we also make use of pretrained sentence encoders and contextualised embedding models. We list here the implementations used:

Name	Link
Skip-Thoughts	`github.com/ryankiros/skip-thoughts`
Doc2Vec	`github.com/jhlau/doc2vec`
InferSent	`github.com/facebookresearch/InferSent`
Universal Sentence Encoder	`tfhub.dev/google/universal-sentence-encoder/2`
ELMo	`tfhub.dev/google/elmo/2`
BERT	`github.com/imgarylai/bert-embedding`

1.6 This Thesis in Context

This thesis was born out of a motivation to investigate the significance of linguistic knowledge in a distributional setting. While the tensor-based modellings explicitly allow the grammatical structure of text to guide the distributional interpretation of the composition of word embeddings, at the time the research of this thesis was done, many deep neural network approaches were developed, that don't explicitly incorporate linguistic knowledge, but rather assume that this knowledge somehow *emerges* from simple textual properties such as co-occurrences, language models, and so on. Hence, this thesis developed itself almost as a piece of comparative research: what can type-driven models offer that neural approaches can't? On the theoretical side, we argue that they offer explanatory power over purely neural models: grammar explains the structure of a sentence, the principle of compositionality explains how the meaning of a sentence must depend on both the meaning of individual expressions and the way they are (syntactically) put together. On the technical side,

however, type-driven models have a disadvantage over neural approaches in that they have not been shown to scale up. Hence, the topic of this thesis is about extending the existing models in theory as well as in practice. And in the latter point, the last two chapters of this thesis — each constituting a separate study during the course of this PhD — offer a comparative experimental perspective: implementing a type-driven model and contrasting it with neural approaches to sentence embeddings, which models are more in line with human judgments of sentence similarity?

Chapter 2

Categorical Distributional Semantics

Chapter Abstract

This chapter lays the foundations of compositional distributional semantics in a type-driven framework, as briefly discussed in Chapter 1. We define the relevant categories that are used in the categorically driven composition model of Coecke, Sadrzadeh, and Clark [CSC10], and expand slightly to include related work on extending this model to Lambek Calculus [CGS13]. We briefly discuss Montague style formal semantics, and string diagrammatic languages for Lambek Calculus [Wij17].

The central challenge in this thesis is that of compositionality in the context of distributional semantics: if we have good distributional representations for words, what constitutes a good distributional representation of a sentence? As discussed in the previous chapter, the methods that learn distributional word representations generally do not lift well to the sentence level, as there are simply too many possible ways of composing words into larger phrases, making it infeasible to learn sentence representations.

Compositional semantics is concerned not with word level meaning representation, but with how to compose the meaning of individual words into the meaning of sentences. It is often attributed to Gottlob Frege, and formalised in the language of universal algebra by Montague [Mon70b], that the meaning of a sentence may be described as a function of its individual word meanings and the way they are combined:

$$F(G)\big(\sigma(w_1)\ ...\ \sigma(w_n)\big)$$

Here, G is some grammatical structure, F a structure preserving map passing from syntactic to semantic structure, and σ a semantic map assigning the meaning representations of individual words. The biggest issue of type-theoretic formal semantics is that individual word meaning are described in holistic terms, using predefined set-theoretic models of the world. So the big innovation caused by distributional

models of word meaning quite naturally led researchers to combine the complementary strengths of both worlds: take the crisp compositional semantics from the world of types, and insert the lexical content of word embeddings.

In this chapter we describe two ways of achieving such a compositional distributional model: first, we review the work done by Coecke, Sadrzadeh, and Clark [CSC10] and others [CGS13; SCC13; GS11a; KS14; Kar16] on a categorical model of composition for distributional semantics, and then we briefly review the work of Muskens and Sadrzadeh [MS16; MS17] where not category theory, but lambda calculus plays the role of mediator between syntax and semantics.

2.1 Categorical Composition

The work of Coecke, Sadrzadeh, and Clark [CSC10] uses category theory to unify the seemingly unrelated structures underlying grammar and distributional representations. Although a tour into these references might daunt the reader slightly, the main point of the approach is this: finite dimensional vector spaces and linear maps between them, the mathematical structure used in distributional semantics, can be seen as an example of a compact closed category. Lambek's pregroup grammar [Lam99] can also be seen as an example of a compact closed same category. This means that we can related the syntactic structures we get from a grammar, to the operations on vector representations of word meaning. In later work, Coecke, Grefenstette, and Sadrzadeh [CGS13] show how also the Lambek Calculus can be treated as a category and the authors consider the categorical version of Montague's homomorphic formulation of compositionality to describe a *functorial passage* from the Lambek Calculus to finite-dimensional vector spaces.

Here we work out the formal prerequisites to understand this approach. But to be able to define the relevant formalisms we need to first establish some basic categorical notions.

2.1.1 Monoidal Categories

First, a category $\mathbf{C}$ consists of a collection of objects $Ob(\mathbf{C})$, denoted by A, B etc., and a collection of arrows $Ar(\mathbf{C})$, denoted by f, g etc. Each arrow has a domain and codomain, for which we colloquially write $f : A \to B$. Moreover, we can write $Hom^{\mathbf{C}}(A, B)$ to denote all the arrows with the same domain and codomain. The defining arrows of any category are identity arrows, and the presence of composition. That is, for every object A there is an arrow $id_A : A \to A$, and for every two arrows $f : A \to B$ and $g : B \to C$ there is a composite arrow $g \circ f : A \to C$. These arrows need to satisfy some basic equations, expressing the redundancy of composing with an identity arrow, and associativity of arrow composition:

$$f \circ id_A = f = id_B \circ f \qquad \text{for } f : A \to B,$$

$$h \circ (g \circ f) = (h \circ g) \circ f \qquad \text{for } f : A \to B, g : B \to C, h : C \to D.$$

A *functor* is a mapping between categories $F : \mathbf{C} \to \mathbf{D}$ that assigns to each object A in $Ob(\mathbf{C})$ an object $F(A)$ in $Ob(\mathbf{D})$ and to each arrow $f : A \to B$ in $Ar(\mathbf{C})$ an arrow $F(f) : F(A) \to F(B)$ in $Ar(\mathbf{D})$ such that identities and composition are preserved:

1. $F(g \circ f) = F(g) \circ F(f)$,
2. $F(id_A) = id_{F(A)}$.

Functors are the categorical analogue of a homomorphism, i.e. a structure-preserving map. Functors that have two arguments are called *bifunctors* and have the following restrictions:

- Identities should be preserved, so $F(id_{(A,B)}) = id_{F(A,B)}$,
- Composition should be preserved, so $F(k \circ g, h \circ f) = F(k, h) \circ F(g, f)$.

Finally, a natural isomorphism is a map that abstract away over objects, while respecting a notion of isomorphims: first, a natural transformation is a collection of maps $\theta_A : F(A) \to G(A)$ between functors F, G such that for any such map either applying the natural transformation or the functor first will give the same map:

$$\begin{array}{ccc} F(A) & \xrightarrow{\theta_A} & G(A) \\ {\scriptstyle F(f)}\downarrow & & \downarrow{\scriptstyle G(f)} \\ F(B) & \xrightarrow[\theta_B]{} & G(B) \end{array}$$

The categories that will be relevant for this thesis are mostly monoidal, that is to say, they are endowed with a bifunctor $\otimes : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$, and a unit object I, with natural isomorphisms $\alpha_{A,B,C} : (A \otimes B) \otimes C \to A \otimes (B \otimes C)$, $\lambda_A : I \otimes A \to A$ and $\rho_A : A \otimes I \to A$, such that the diagrams below commute:

$$\begin{array}{ccccc} & & (A \otimes (B \otimes C)) \otimes D & \xrightarrow{\alpha_{A,B\otimes C,D}} & A \otimes ((B \otimes C) \otimes D) \\ & {\scriptstyle \alpha_{A,B,C} \otimes id_D}\nearrow & & & \searrow{\scriptstyle id_A \otimes \alpha_{B,C,D}} \\ ((A \otimes B) \otimes C) \otimes D & & & & A \otimes (B \otimes (C \otimes D)) \\ & {\scriptstyle \alpha_{A\otimes B,C,D}}\searrow & & \nearrow{\scriptstyle \alpha_{A,B,C\otimes D}} & \\ & & (A \otimes B) \otimes (C \otimes D) & & \end{array}$$

$$\begin{array}{ccc} (A \otimes I) \otimes B & \xrightarrow{\alpha_{A,I,B}} & A \otimes (I \otimes B) \\ & {\scriptstyle \rho_A \otimes id_B}\searrow \quad \swarrow{\scriptstyle id_A \otimes \lambda_B} & \\ & A \otimes B & \end{array}$$

2.1.2 Closing the $\otimes$

The linguistic structures that we will talk about use a monoidal bifunctor $\otimes$ to model concatenation or merging of information, whereas in vector spaces the operation $\otimes$ interprets this merging by the tensor product between vector spaces. Both pregroup grammar and Lambek Calculus, the formalisms of interest in this chapter, have their own type of closure with respect to the $\otimes$.

Compact Closed Categories First, we define closure by duality, which leads us to autonomous and compact closed categories. An autonomous category is a monoidal category $(\mathbf{C}, \otimes, \alpha, I, \lambda, \rho)$ such that for every object A in $Ob(\mathbf{C})$ there exist objects A^l and A^r (called left and right adjoints) and for every A there exist morphisms

$$A^l \otimes A \xrightarrow{\epsilon^l} I \xrightarrow{\eta^l} A \otimes A^l \qquad A \otimes A^r \xrightarrow{\epsilon^r} I \xrightarrow{\eta^r} A^r \otimes A$$

which respect the following commuting diagrams:

$$\begin{array}{ccc} (A \otimes A^l) \otimes A & \xrightarrow{\alpha_{A,A^l,A}} & A \otimes (A^l \otimes A) \\ {\scriptstyle \eta^l \otimes id_A}\uparrow & & \downarrow{\scriptstyle id_A \otimes \epsilon^l} \\ I \otimes A & & A \otimes I \\ {\scriptstyle \lambda_A^{-1}}\uparrow & & \downarrow{\scriptstyle \rho_A} \\ A & \xrightarrow[id_A]{} & A \end{array} \qquad \begin{array}{ccc} A^l \otimes (A \otimes A^l) & \xrightarrow{\alpha^{-1}_{A^l,A,A^l}} & (A^l \otimes A) \otimes A^l \\ {\scriptstyle id_{A^l} \otimes \eta^l}\uparrow & & \downarrow{\scriptstyle \epsilon^l \otimes id_{A^l}} \\ A^l \otimes I & & I \otimes A^l \\ {\scriptstyle \rho^{-1}_{A^l}}\uparrow & & \downarrow{\scriptstyle \lambda_A} \\ A^l & \xrightarrow[id_{A^l}]{} & A^l \end{array}$$

$$\begin{array}{ccc} A \otimes (A^r \otimes A) & \xrightarrow{\alpha^{-1}_{A,A^r,A}} & (A \otimes A^r) \otimes A \\ {\scriptstyle id_A \otimes \eta^r}\uparrow & & \downarrow{\scriptstyle \epsilon^r \otimes id_A} \\ A \otimes I & & I \otimes A \\ {\scriptstyle \rho_A^{-1}}\uparrow & & \downarrow{\scriptstyle \lambda_A} \\ A & \xrightarrow[id_A]{} & A \end{array} \qquad \begin{array}{ccc} (A^r \otimes A) \otimes A^r & \xrightarrow{\alpha_{A^r,A,A^r}} & A^r \otimes (A \otimes A^r) \\ {\scriptstyle \eta^r \otimes id_{A^r}}\uparrow & & \downarrow{\scriptstyle id_{A^r} \otimes \epsilon^r} \\ I \otimes A^r & & A^r \otimes I \\ {\scriptstyle \lambda^{-1}_{A^r}}\uparrow & & \downarrow{\scriptstyle \rho_{A^r}} \\ A^r & \xrightarrow[id_{A^r}]{} & A^r \end{array}$$

In the case that the monoidal category is also symmetric, the left and right dual objects collapse into one dual object (up to isomorphism) and we speak of a *compact closed category* and denote the dual of A by A^*.

Biclosed Monoidal Categories The second way of closing the category is by directly closing the bifunctor $\otimes$, rather than through dual objects. We call a monoidal category $(\mathbf{C}, \otimes, \alpha, I, \lambda, \rho)$ *left closed* if it has a bifunctor $\Rightarrow$: $\mathbf{C}^{\mathrm{op}} \times \mathbf{C} \to \mathbf{C}$ (i.e. contravariant in its first argument, covariant in its second argument) together with a

natural isomorphism specified by $\beta_{A,B,C} : Hom^{\mathbf{C}}(A \otimes B, C) \to Hom^{\mathbf{C}}(B, A \Rightarrow C)$. Similarly, a *right closed* monoidal category is obtained in the presence of a bifunctor $\Leftarrow\colon \mathbf{C} \times \mathbf{C}^{\mathrm{op}} \to \mathbf{C}$ and a natural isomorphism $\gamma_{A,B,C} : Hom^{\mathbf{C}}(A \otimes B, C) \to Hom^{\mathbf{C}}(A, C \Leftarrow B)$. Of course, a biclosed monoidal category is one which is both left and right closed. Symmetric monoidal categories with either closure induce a biclosed structure.

Frobenius Algebras Subsequent work on modelling natural language meaning in a categorical model by Sadrzadeh, Clark, and Coecke [SCC13; SCC14] have used additional structure that is available in the category of finite dimensional vector spaces (which we will define below), called a Frobenius algebra. This structure delivers operations that can be used to expand vectors into matrices, or to compress matrices in vectors, which is used to model relative pronouns. Formally, a Frobenius algebra in a symmetric monoidal category $(\mathcal{C}, \otimes, I)$ is a tuple $(X, \Delta, \iota, \mu, \zeta)$ where, for X an object of $\mathcal{C}$, the first triple below is an internal comonoid and the second one is an internal monoid.

$$(X, \Delta, \iota) \qquad (X, \mu, \zeta)$$

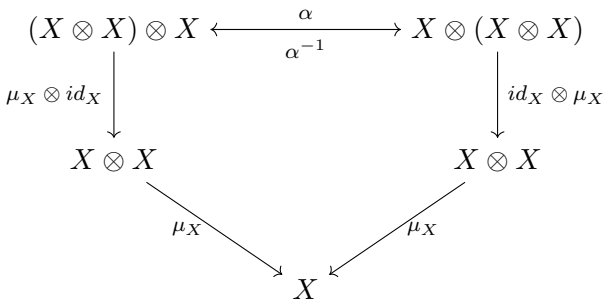
This means that we have a coassociative map Δ and and its counit ι:

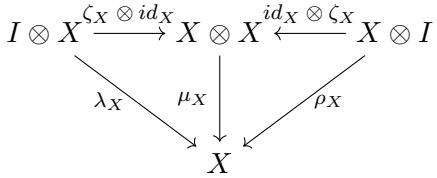
$$\Delta\colon X \to X \otimes X \qquad \iota\colon X \to I$$

and an associative map μ and its unit ζ:

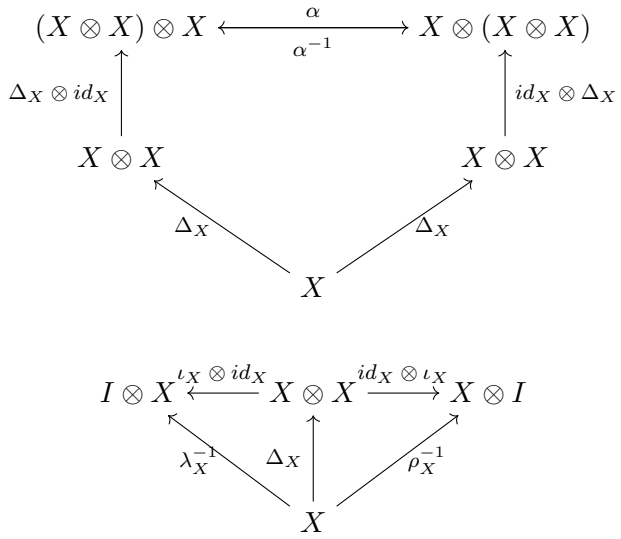
$$\mu\colon X \otimes X \to X \qquad \zeta\colon I \to X$$

as morphisms of our category $\mathcal{C}$. Being an internal monoid means that the following equations are satisfied:





whereas for the internal comonoid, the arrows are simply reversed:



The Δ and μ morphisms satisfy the *Frobenius condition* given below

$$(\mu \otimes id_X) \circ (id_X \otimes \Delta) = \Delta \circ \mu = (id_X \otimes \mu) \circ (\Delta \otimes id_X)$$

Informally, the comultiplication Δ expands the information contained in one object into two objects; the multiplication μ combines the information of two objects into one. The Frobenius condition then states that combining of information followed by an expansion of information, is compatible with expanding one half of the original, granted that one combines the obtained extra object with the original. It is moreover important to define the properties of a special and commutative Frobenius Algebras: in the case of a commutative Frobenius Algebra we have that

$$\sigma \circ \Delta = \Delta \qquad \text{and} \qquad \mu \circ \sigma = \mu$$

so that the Frobenius expansion and combining are insensitive to the ordering of information. In the case of a special Frobenius Algebra we moreover have that

$$\mu \circ \Delta = id$$

which tells us that combining information that itself was expanded, doesn't have any effect. Finally, it is important to note that the tensor product of a symmetric Frobenius Algebra is itself a symmetric Frobenius Algebra, where the structure carries over from the original algebras:

$$\Delta_{A \otimes B} \coloneqq \Delta_A \otimes \Delta_B$$

$$\mu_{A \otimes B} \coloneqq (\mu_A \otimes \mu_B) \circ id_A \otimes \sigma_{B,A} \otimes id_B$$

We define the concrete operations of a Frobenius Algebra below in Section 2.1.6.

2.1.3 Pregroups as an autonomous category

We start out with discussing Lambek's pregroup grammar [Lam99], and will see how it naturally forms an autonomous category. As with typelogical grammar, in pregroup grammar words are associated types and sequences of words are deemed *grammatical* when their associated sequence of types leads one to derive a distinguished (sentence) type. Formally, a pregroup is a partially ordered monoid $(A, 1, \cdot, \leq)$ with, for every element x, a left and right adjoint x^l,x^r that satisfy

$$x^l \cdot x \leq 1 \leq x \cdot x^l \qquad x \cdot x^r \leq 1 \leq x^r \cdot x$$

From this, one derives

$$1^l = 1 = 1^r \qquad x^{lr} = x = x^{rl} \qquad (x \cdot y)^l = y^l \cdot x^l \qquad (x \cdot y)^r = y^r \cdot x^r$$

In order to construct a sentence, one associates types to words. As an example, consider "Bill sings and Hannah dances". Assuming basic types np (noun phrase) and s (sentence), we get the type assignment with reduction of Figure 2.1.

Bill	sings	and	Hannah	dances	
np	$np^r \cdot s$	$s^r \cdot s \cdot s^l$	np	$np^r \cdot s$	$\leq$
	s	$s^r \cdot s \cdot s^l$		s	$\leq$
		s			

FIGURE 2.1: A pregroup derivation of "Bill sings and Hannah dances".

Pregroups are algebraic structures, and they naturally form an autonomous category: for a pregroup $(A, 1, \cdot, \leq)$, the objects of the category **Preg** are the elements in A, its monoidal tensor is $\cdot$, and its unit is 1. An arrow between objects $f : A \to B$ exists whenever there is a reduction $A \leq B$. The conditions on the dualising objects are now given by reductions of the sort

$$a \leq a \cdot 1 \leq a \cdot a^r \cdot a \leq 1 \cdot a \leq a$$

Note that in the original formulation, this category is posetal, having at most one arrow between two objects. But, as noted in Kartsaklis [Kar15], this is not wanted in general as multiple reductions may be possible, leading to different semantics. It is solved by reformulating the categorification of pregroups in the language of free 2-categories [PL07]. For our purposes it suffices to simply note that multiple arrows between the same objects may exist, and that all the equations of a compact closed category are easily satisfied.

2.1.4 Lambek Calculus as a Biclosed Monoidal Category

Most of the work in this thesis works with extensions of the Lambek Calculus, whether in its original associative formulation of Lambek [Lam58] or its nonassociative refinement [Lam61]. We present it here as a 'deductive system' after Lambek [Lam68], where one defines axioms and inference rules. The reason to do so is that this style of presentation directly corresponds to the presentation style commonly used in category theory.

The core component of the (nonassociative) Lambek Calculus is given in Figure 3.1. It starts as the most basic deductive system with an axiom scheme (the identity arrow on types) and composition (of arrows). Next, the inference rules defining the scheme between the concatenation connective $\otimes$ and its residuals $\backslash$ and $/$ are given. The monotonicity rules of Figure 2.3 are derived rules of inference. To define

$$\frac{}{1_A : A \longrightarrow A} \qquad \frac{f : A \longrightarrow B \quad g : B \longrightarrow C}{g \circ f : A \longrightarrow C}$$

$$\frac{f : A \otimes B \longrightarrow C}{\triangleright f : A \longrightarrow C/B} \qquad \frac{f : A \otimes B \longrightarrow C}{\triangleleft f : B \longrightarrow A\backslash C}$$

$$\frac{g : A \longrightarrow C/B}{\triangleright^{-1} g : A \otimes B \longrightarrow C} \qquad \frac{g : B \longrightarrow A\backslash C}{\triangleleft^{-1} g : A \otimes B \longrightarrow C}$$

FIGURE 2.2: Lambek's nonassociative calculus of [Lam61], presented as a deductive system.

$$\frac{f : A \to C \quad g : B \to D}{f \otimes g : A \otimes B \to C \otimes D} \otimes \qquad \frac{f : A \to C \quad g : B \to D}{g/f : B/C \to D/A} / \qquad \frac{f : A \to C \quad g : B \to D}{f\backslash g : C\backslash B \to A\backslash D} \backslash$$

$$f \otimes g := \triangleright^{-1}((\triangleright \triangleleft^{-1} ((\triangleleft 1_{C\otimes D}) \circ g)) \circ f)$$
$$g/f := \triangleright(g \circ (\triangleleft^{-1}((\triangleleft \triangleright^{-1} 1_{B\backslash C}) \circ f)))$$
$$f\backslash g := \triangleleft(g \circ (\triangleright^{-1}((\triangleright \triangleleft^{-1} 1_{C\backslash B}) \circ f)))$$

FIGURE 2.3: Monotonicity rules, their derivation, and application and coapplication laws.

Lambek's original system **L** we add two arrows

$$\alpha_{A,B,C} : (A \otimes B) \otimes C \to A \otimes (B \otimes C)$$

$$\alpha^{-1}_{A,B,C} : A \otimes (B \otimes C) \to (A \otimes B) \otimes C$$

Deriving Sentences Just like with pregroup grammar, words are assigned types, this time from a set of basic types and the connectives $\otimes, \backslash, /$. We can recast the example from the previous section with Lambek types:

Bill	sings	and	Hannah	dances	
np	$np\backslash s$	$(s\backslash s)/s$	np	$np\backslash s$	$\vdash s$

However, contrary to pregroups, the Lambek Calculus is a typelogical grammar. That is, a sequence of words $w_1, ..., w_n$ is grammatical when the associated sequence of types $t(w_1), ..., t(w_n)$ induces a *proof* of $t(w_1) \otimes ... \otimes t(w_n) \to s$. For the example above we obtain the proof in Figure 2.4.

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{np \to np}\ 1_{np} \quad \overline{s \to s}\ 1_s \backslash}{np\backslash s \to np\backslash s}}{np \otimes np\backslash s \to s} \triangleleft^{-1} \quad \overline{s \to s}\ 1_s \backslash}{s\backslash s \to (np \otimes np\backslash s)\backslash s} \quad \dfrac{\dfrac{\overline{np \to np}\ 1_{np} \quad \overline{s \to s}\ 1_s \backslash}{np\backslash s \to np\backslash s}}{np \otimes np\backslash s \to s} \triangleleft^{-1} /}{(s\backslash s)/s \to ((np \otimes np\backslash s)\backslash s)/(np \otimes np\backslash s)} \triangleright^{-1}}{(s\backslash s)/s \otimes (np \otimes np\backslash s) \to (np \otimes np\backslash s)\backslash s} \triangleleft^{-1}}{(np \otimes np\backslash s) \otimes ((s\backslash s)/s \otimes (np \otimes np\backslash s)) \to s}$$

FIGURE 2.4: A proof of grammaticality for "Bill sings and Hannah dances".

Closure for Lambek Calculus The (associative) Lambek Calculus, in its deductive systems presentation, can be shown to form an instance of a biclosed monoidal category. Its objects are types that are built from a set of basic types and the connectives $\otimes, \backslash, /$. For the arrows, one defines an equivalence relation over proofs in which the categorical laws for a biclosed monoidal category are stipulated. That is, composing with an identity proof is vacuous, applying a residuation rule followed by its inverse gives you back the original arrow, and the monotonicity rules respect the bifunctor laws. For a full discussion we refer the reader to Lambek's work [Lam68; Lam88].

From Lambek to Lambek A natural question to ask is what the relation is that holds between the Lambek Calculus and pregroup grammars. The answer on the level of types and reductions/deductions is that pregroup reductions can be seen as the *image* of derivations in the Lambek Calculus, under a translation of syntactic types to pregroup types:

$$\lceil p \rceil = p \quad \lceil A \otimes B \rceil = \lceil A \rceil \cdot \lceil B \rceil \quad \lceil A\backslash B \rceil = \lceil A \rceil^r \cdot \lceil B \rceil \quad \lceil B/A \rceil = \lceil B \rceil \cdot \lceil A \rceil^l$$

Buszkowski [Bus01] shows that under this translation, pregroups can be seen as a special case of Lambek Calculus. This idea will reflect in the following sections, where we show, after giving a truth-conditional semantics, how finite dimensional spaces form a compact closed category, and how derivations in pregroup grammars and Lambek Calculus can both be interpreted as linear maps over vector spaces.

2.1.5 Formal Semantics for the Lambek Calculus

Before defining vector spaces as a possible semantics for Lambek and pregroup grammars, we briefly discuss formal semantics based on simply typed lambda calculi. The Curry-Howard correspondence is a well-known correspondence that states

that proofs in intuitionistic logic canonically correspond to *programs* in a simply typed lambda calculus[1]. Later on, in their book on categorical logic, Lambek and Scott [LS88] extended this correspondence to include category theory. Next to the proofs-as-programs connection between intuitionistic logic and (typed) lambda calculus, the added interpretation is that these programs form the defining instances of a Cartesian closed category. As the Lambek calculus is an intuitionistic system, it therefore allows for the formulation of an accompanying lambda calculus, which would be sensitive to the directionality of function application. Although a strict correspondence can be formulated (see for example [Wan90]), one often weakens the requirements of a strict correspondence and uses a standard formulation of typed linear lambda calculus, akin to the system introduced by Montague [Mon70a].

Typed Lambda Calculus We start out by giving a formal definition of typed lambda calculus (in the style of Church [Chu40]), which will serve as the abstract formal semantics for Lambek grammars.

Definition 2 *Given a countably infinite set of variables $V = \{x, y, z...\}$, terms of λ are as in the below grammar:*

$$M, N := V \mid \lambda x.M \mid M\ N \mid \langle M,\ N\rangle \mid \pi_1(M) \mid \pi_2(M)$$

Terms obey the standard α-, η- and β-conversion rules:

Definition 3 *For terms of λ we define three conversion relations:*

1. *α-conversion: for any term M we have*

$$M \quad =_\alpha \quad M[x \mapsto y]$$

provided that y is a fresh variable, i.e. it does not occur in M.

2. *η-conversion: for terms M we have*

$$\begin{array}{lll} \lambda x.M\ x & =_\eta & M \quad (x \text{ does not occur in } M) \\ \langle \pi_1(M),\ \pi_2(M)\rangle & =_\eta & M \end{array}$$

3. *β-conversion: for terms M we define*

$$\begin{array}{lll} (\lambda x.M)\ N & \to_\beta & M[x \mapsto N] \\ \pi_1(\langle M,\ N\rangle) & \to_\beta & M \\ \pi_2(\langle M,\ N\rangle) & \to_\beta & N \end{array}$$

We moreover write $M \twoheadrightarrow_\beta N$ whenever M converts to N in multiple steps.

[1] For an extensive review on this topic see the book of Sørensen and Urzyczyn [SU06].

Interpreting Lambek Calculus In order to give an interpretation of derivations in a Lambek grammar in formal semantics, we define a set D_e, the domain of entities, and the set D_t for the domain of truth values. Out of these, and with the constructions of the typed lambda calculus, we can form functions $D_{A\to B}$ from D_A to D_B, as well as pairs $D_A \times D_B$ using the standard set-theoretic Cartesian product.

We can now give an interpretation of types and proofs of the Lambek Calculus in typed linear lambda calculus. A usual interpretation of basic types np, n, s is to assign $\lfloor np \rfloor = e, \lfloor n \rfloor = e \to t, \lfloor s \rfloor = t$. Then, on complex types we have

$$\lfloor A \otimes B \rfloor = \lfloor A \rfloor \times \lfloor B \rfloor \quad \lfloor A/B \rfloor = \lfloor A \backslash B \rfloor = \lfloor A \rfloor \to \lfloor B \rfloor$$

For the proofs, we can interpret identity and composition straightforwardly by $\lambda x.x$ and $\lambda x.N\ (M\ x)$ for M and N the terms of the subproofs, respectively. The binary residuation rules correspond to application and abstraction depending on the direction in which the rule is applied:

$$\lfloor \triangleright M \rfloor = \lambda x\ y.M\ \langle x, y \rangle \qquad \lfloor \triangleleft M \rfloor = \lambda y\ x.M\ \langle x, y \rangle$$
$$\lfloor \triangleright^{-1} N \rfloor = \lambda \langle x, y \rangle.(N\ x)\ y \qquad \lfloor \triangleleft^{-1} N \rfloor = \lambda \langle x, y \rangle.(N\ y)\ x$$

The derived monotonicity rules get the interpretation below:

$$\lfloor M \otimes N \rfloor = \lambda \langle x, y \rangle.\langle M\ x, N\ y \rangle$$

$$\lfloor M \backslash N \rfloor = \lambda f\ x.N\ (f\ (M\ x))$$

$$\lfloor M / N \rfloor = \lambda f\ x.M\ (f\ (N\ x))$$

The associativity rules behave as an identity since associativity is implicit in lambda terms:

$$\lfloor \alpha(M) \rfloor = \lambda \langle x, y, z \rangle.M\ \langle x, y, z \rangle \qquad \lfloor \alpha^{-1}(M) \rfloor = \lambda \langle x, y, z \rangle.M\ \langle x, y, z \rangle$$

As an example, the proof from Figure 2.4 with proof term

$$\triangleleft^{-1}\ \triangleright^{-1}\ /(\backslash(\triangleleft^{-1} \backslash (1_{np}, 1_s), 1_s), \triangleleft^{-1} \backslash (1_{np}, 1_s))$$

is now interpreted as an abstract term

$$\lambda \langle x, y, z, u, v \rangle.z\ (u\ v)(xy)$$

given some suitable lexical specification of the meaning of the words in the sentence, the final meaning will be obtained by substituting these in, in the place of the variables. Assuming that the terms **bill**, **hannah** denote single entities (in D_e) and the constants for the verbs **sing**, **dance** denote characteristic functions from D_e to D_t, therefore characterising a set of entities, and the interpretation of 'and' is a map

$\lambda Q^t.\lambda P^t.P \wedge Q$, we apply the abstract term to these constants and get a concrete meaning term

$$(\textbf{sing bill}) \wedge (\textbf{dance hannah})$$

The formal semantic modelling lends itself to clear logical reasoning thanks to its toolset of logical connectives and corresponding set-theoretic operations. However, it is not clear how to obtain the crisp entries of the actual sets that will form the content of the lexical semantics. In that sense vector space semantics offers the benefit of being able to learn the content of words directly from corpus data. However, we include the formal semantics review here as we will make use of a hybrid lambda vector modelling in Chapter 3. In the next section, we discuss the categorical formulation of vector space semantics for pregroup and Lambek grammars.

2.1.6 Vector Spaces as a Semantic Category

Finite dimensional vector spaces and linear maps between them can be considered a category, which we refer to as **FVect**. In this category, the objects are vector spaces, and the arrows are linear maps. We consider the tensor product between vector spaces, $\otimes$, as the monoidal bifunctor of **FVect**, with the field of the real numbers $\mathbb{R}$ as its unit I; identity maps, composition and tensor product are defined as usual.

Compact Closure of FVect Since any vector space A has a dual space A^*, **FVect** forms a compact closed category, with the ϵ, η maps instantiated with the concrete maps below, respectively taking inner products and producing identity tensors. Since bases of vector spaces are fixed in concrete models, there is a canonical way of defining a basis for a *dual space*, so that $V^* \cong V$. In concrete models we may therefore collapse the adjoints completely, leading to the following concrete definition:

$$\epsilon_V : V \otimes V \to \mathbb{R} \quad \text{given by} \quad \sum_{ij} v_{ij}(\vec{e}_i \otimes \vec{e}_j) \mapsto \sum_i v_{ii}$$

$$\eta_V : \mathbb{R} \to V \otimes V \quad \text{given by} \quad \lambda \mapsto \sum_i \lambda(\vec{e}_i \otimes \vec{e}_i)$$

Interpreting Pregroup Grammar Given that **Preg** forms an autonomous category and **FVect** forms a compact closed category, their shared structure allows one to interpret derivations of a pregroup grammar in terms of linear maps over vector spaces [CSC10; Kar15] by means of a *functorial passage*. On the type level, we have some interpretation of basic types into basic vector spaces ($\lceil n \rceil = \lceil np \rceil = N, \lceil s \rceil = S$), concatenation is interpreted as tensor product ($\lceil a \cdot b \rceil = \lceil a \rceil \otimes \lceil b \rceil$), both left and right adjoints for a type are interpreted by the dual space, which in concrete models collapses to the same space ($\lceil a^l \rceil = \lceil a^r \rceil = \lceil a \rceil$). Then, both contraction rules of pregroups become inner products in **FVect**, whereas the expansions are instantiated by the η maps.

The running example "Bill sings and Hannah dances" can now be given an interpretation: given vectors for each of the words, $\overrightarrow{bill}, \overrightarrow{hannah} \in N, \overline{sings}, \overline{dances} \in$

$N \otimes S, \overline{\overline{and}} \in S \otimes S \otimes S$, the derivation of Figure 2.1 is given as

$$(id_S \otimes \epsilon_S) \circ (\epsilon_N \otimes \epsilon_S \otimes id_S \otimes id_S \otimes \epsilon_N \otimes id_S) : N \otimes N \otimes S \otimes S \otimes S \otimes S \otimes N \otimes N \otimes S \to S$$

Concretely, this will link together the subject dimensions of the verb vectors with respective subjects, to then let the coordinator 'and' merge the result into a single sentence space:

$$(\overline{\overline{and}} \times_2 (\overline{dances} \times_1 \overrightarrow{hannah})) \times_1 (\overline{sings} \times_1 \overrightarrow{bill})$$

Interpreting Lambek Calculus In later work on categorical distributional semantics, Coecke, Grefenstette, and Sadrzadeh [CGS13] consider the Lambek Calculus as an alternative to pregroup grammar, interpreting Lambek types as vector spaces and the deduction rules as linear maps. At the type level, the interpretation functor $\lceil \cdot \rceil$ assigns a vector space to the atomic types of **L**, the binary type-forming operators are interpreted as

$$\lceil A \otimes B \rceil = \lceil A \rceil \otimes \lceil B \rceil \quad \lceil A/B \rceil = \lceil A \rceil \otimes \lceil B \rceil^* \quad \lceil A \backslash B \rceil = \lceil A \rceil^* \otimes \lceil B \rceil$$

Interpretation: proofs From the linear maps interpreting the premises of the **L** inference rules, we want to compute the linear map interpreting the conclusion. Identity and composition are immediate: $\lceil 1_A \rceil = 1_{\lceil A \rceil}$, $\lceil g \circ f \rceil = \lceil g \rceil \circ \lceil f \rceil$. For the residuation inferences, from the map $\lceil f \rceil : \lceil A \rceil \otimes \lceil B \rceil \longrightarrow \lceil C \rceil$ interpreting the premise, we obtain

$$\lceil \triangleright f \rceil = \lceil A \rceil \xrightarrow{1_{\lceil A \rceil} \otimes \eta_{\lceil B \rceil}} \lceil A \rceil \otimes \lceil B \rceil \otimes \lceil B \rceil^* \xrightarrow{\lceil f \rceil \otimes 1_{\lceil B \rceil^*}} \lceil C \rceil \otimes \lceil B \rceil^*$$

$$\lceil \triangleleft f \rceil = \lceil B \rceil \xrightarrow{\eta_{\lceil A \rceil} \otimes 1_{\lceil B \rceil}} \lceil A \rceil^* \otimes \lceil A \rceil \otimes \lceil B \rceil \xrightarrow{1_{\lceil A \rceil^*} \otimes \lceil f \rceil} \lceil A \rceil^* \otimes \lceil C \rceil$$

For the inverses, from maps $\lceil g \rceil : \lceil A \rceil \longrightarrow \lceil C/B \rceil$, $\lceil h \rceil : \lceil B \rceil \longrightarrow \lceil A \backslash C \rceil$ for the premises, we obtain

$$\lceil \triangleright^{-1} g \rceil = \lceil A \rceil \otimes \lceil B \rceil \xrightarrow{\lceil g \rceil \otimes 1_{\lceil B \rceil}} \lceil C \rceil \otimes \lceil B \rceil^* \otimes \lceil B \rceil \xrightarrow{1_{\lceil C \rceil} \otimes \epsilon_{\lceil B \rceil}} \lceil C \rceil$$

$$\lceil \triangleleft^{-1} h \rceil = \lceil A \rceil \otimes \lceil B \rceil \xrightarrow{1_{\lceil A \rceil} \otimes \lceil h \rceil} \lceil A \rceil \otimes \lceil A \rceil^* \otimes \lceil C \rceil \xrightarrow{\epsilon_{\lceil A \rceil} \otimes 1_{\lceil C \rceil}} \lceil C \rceil$$

For monotonicity, the case of parallel composition is immediate: $\lceil f \otimes g \rceil = \lceil f \rceil \otimes \lceil g \rceil$. For the $\backslash, /$ cases, from $\lceil f \rceil : \lceil A \rceil \longrightarrow \lceil B \rceil$ and $\lceil g \rceil : \lceil C \rceil \longrightarrow \lceil D \rceil$, we obtain

$$\lceil f/g \rceil = \qquad\qquad \lceil f\backslash g \rceil =$$

$$\begin{array}{c} \lceil A \rceil \otimes \lceil D \rceil^* \\ \Big\downarrow \lceil f \rceil \otimes \eta_{\lceil C \rceil} \otimes 1_{\lceil D \rceil^*} \\ \lceil B \rceil \otimes \lceil C \rceil^* \otimes \lceil C \rceil \otimes \lceil D \rceil^* \\ \Big\downarrow 1_{\lceil B \rceil \otimes \lceil C \rceil^*} \otimes \lceil g \rceil \otimes 1_{\lceil D \rceil^*} \\ \lceil B \rceil \otimes \lceil C \rceil^* \otimes \lceil D \rceil \otimes \lceil D \rceil^* \\ \Big\downarrow 1_{\lceil B \rceil \otimes \lceil C \rceil^*} \otimes \epsilon_{\lceil D \rceil} \\ \lceil B \rceil \otimes \lceil C \rceil^* \end{array} \qquad \begin{array}{c} \lceil B \rceil^* \otimes \lceil C \rceil \\ \Big\downarrow 1_{\lceil B \rceil^*} \otimes \eta_{\lceil A \rceil} \otimes \lceil g \rceil \\ \lceil B \rceil^* \otimes \lceil A \rceil \otimes \lceil A \rceil^* \otimes \lceil D \rceil \\ \Big\downarrow 1_{\lceil B \rceil^*} \otimes \lceil f \rceil \otimes 1_{\lceil A \rceil^* \otimes \lceil D \rceil} \\ \lceil B \rceil^* \otimes \lceil B \rceil \otimes \lceil A \rceil^* \otimes \lceil D \rceil \\ \Big\downarrow \epsilon_{\lceil B \rceil} \otimes 1_{\lceil A \rceil^* \otimes \lceil D \rceil} \\ \lceil A \rceil^* \otimes \lceil D \rceil \end{array}$$

Interpretation for the associativity rules is obtained via the standard associativity maps of **FVect**: $\lceil \alpha f \rceil = f \circ \alpha$ and $\lceil \alpha^{-1} f \rceil = f \circ \alpha^{-1}$.

The proof in Figure 2.4 can now be interpreted as a linear map. The proof itself is encoded by the term

$$\lhd^{-1} \rhd^{-1} /(\backslash(\lhd^{-1}\backslash(1_{np}, 1_s), 1_s), \lhd^{-1}\backslash(1_{np}, 1_s))$$

which translates to exactly the same linear map as in the pregroup case, i.e. we end up with

$$\left(\overline{\overline{and}} \times_2 (\overline{dances} \times_1 \overrightarrow{hannah})\right) \times_1 (\overline{sings} \times_1 \overrightarrow{bill})$$

This is not surprising, as the interpretation of **L** is preserved under the translation to pregroup reductions given above.

Frobenius Algebras for Vector Spaces As mentioned in the section above, the category **FVect** also possesses more structure, namely that of Frobenius Algebras. Every vector space V contains such an algebra $(V, \Delta, \mu, \iota, \zeta)$. The map Δ takes a vector and places its values on the diagonal of a square matrix, whereas μ extracts the diagonal from a square matrix. The ι and ζ maps respectively sum the coefficients of a vector or introduce a vector with the value 1 for all of its coefficients. All maps are given below:

$$\begin{array}{llcl} \Delta_V : V \to V \otimes V & \text{given by} & \sum_i v_i \vec{e}_i & \mapsto & \sum_i v_i (\vec{e}_i \otimes \vec{e}_i) \\ \iota_V : V \to \mathbb{R} & \text{given by} & \sum_i v_i \vec{e}_i & \mapsto & \sum_i v_i \\ \mu_V : V \otimes V \to V & \text{given by} & \sum_{ij} v_{ij} (\vec{e}_i \otimes \vec{e}_j) & \mapsto & \sum_i v_{ii} \vec{e}_i \\ \zeta_V : \mathbb{R} \to V & \text{given by} & \lambda & \mapsto & \sum_i \lambda \vec{e}_i \end{array}$$

This technically defines a special commutative Frobenius Algebra: embedding a vector on the diagonal of a matrix is insensitive to symmetry, as the resulting matrix is diagonal. Moreover, putting a vector on the diagonal of a matrix and then extracting this diagonal returns the original vector. Frobenius algebras have been used to deal with relative pronouns [SCC13; SCC14] and with coordination [Kar16]. In the concrete example of "Bill sings and Hannah dances", the semantic type of the coordinator 'and' will be $S \otimes S \otimes S$, and one uses the μ_S map to effectuate the element wise product between both conjuncts. That is, with this instantiation, the effect of the computation

$$(\overline{\overline{and}} \times_2 (\overline{dances} \times_1 \overrightarrow{hannah})) \times_1 (\overline{sings} \times_1 \overrightarrow{bill})$$

will be

$$(\overline{dances} \times_1 \overrightarrow{hannah}) \odot (\overline{sings} \times_1 \overrightarrow{bill}) \tag{2.1}$$

reflecting the conjunctive interpretation of 'and', by sharing the weights associated with the coordinated subclauses.

In the following section we will look at ways to make the algebraic presentation more accessible through the use of *string diagrams*.

2.1.7 Proofs and Pictures

To support a clear exposition of the essential reasoning happening in syntax and semantics in the categorical framework one often relies on *string diagrams*: graphical representations of the objects and arrows in a category that are sound and complete with respect to the category they describe. This then means that we can picture words and the relation that holds between them in the syntax (as well as in the semantics) in a much more intuitive way than we would in the case of, say, a sequent calculus. In the case of vector spaces it means that it is not necessary to look at detailed equations in order to understand how vectors, matrices and higher-order tensors are combined to produce a sentence representations.

Coecke, Sadrzadeh, and Clark [CSC10] make extensive use of string diagrams for compact closed categories, whereas Sadrzadeh, Clark, and Coecke [SCC13] include also the graphical representations for Frobenius Algebras. In the work of Wijnholds [Wij17], a graphical language for biclosed magmatic categories[2] is developed to reason about proofs of the Lambek Calculus. We outline both languages below.

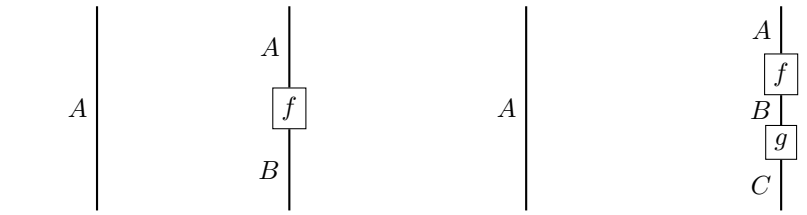
String Diagrams for Compact Closed Categories and Frobenius Algebras

In general, string diagrams form graphical representations of arrows in a given category. We draw from top to bottom, and represent the arrows of a category as *boxes* that connect *wires* together. Specific structures, like compact closure, and Frobenius

[2] These are essentially monoidal categories but without a unit object or associativity.

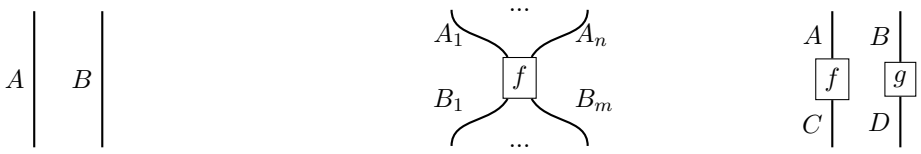
Algebras, are then depicted as special constructions on these boxes and wires. Drawing from top to bottom, we represent the objects and arrows of a general category as

Object	**Morphism**	**Identity**	**Composition**
A	$f : A \to B$	$id_A : A \to A$	$g \circ f$



and those of a monoidal category with

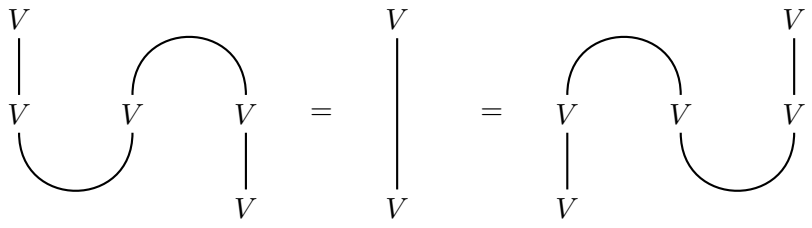
Tensor	**Unit**	**Morphism**	**Tensor**
$A \otimes B$	I	$f : A_1 \otimes ... \otimes A_n \to B_1 \otimes ... \otimes B_m$	$f \otimes g$



We can see that the monoidal bifunctor $\otimes$ is depicted by juxtaposing arrows, or by having multiple incoming and outgoing wires. For compact closed categories, the graphical language is one of *cups* and *caps*. Recall that we have a reduction arrow $\epsilon : V \otimes V \to I$ that 'cancels out' two elements, and an expansion $\eta : I \to V \otimes V$ that 'introduces' two elements. These are drawn as connecting two objects either as a cup in the case of ϵ or as a cap in the case of η:

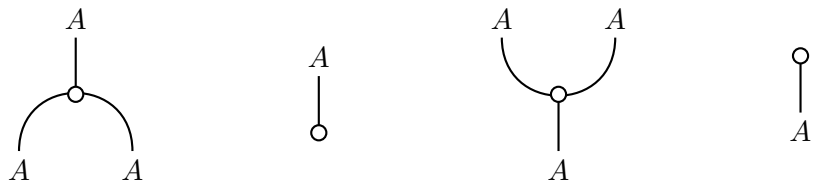
$$\epsilon : V \otimes V \to I \qquad \eta : I \to V \otimes V$$

The defining equations of a compact closed category now are visually testified by

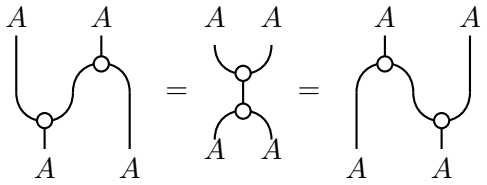


Frobenius Diagrams Given that Frobenius algebras are defined on individual objects, we represent this graphically by having internal nodes for an object.

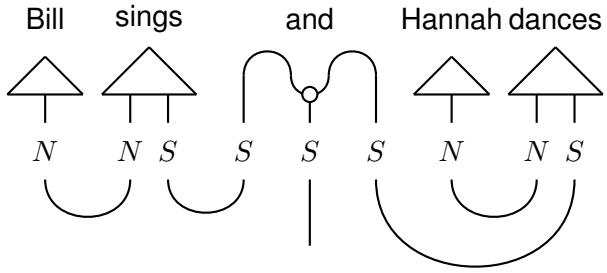
$\Delta : A \to A \otimes A \qquad \iota : A \to I \qquad \mu : A \otimes A \to A \qquad \zeta : I \to A$



The Frobenius condition can now be drawn as



As derivations in **Preg** and **FVect** have the same structure (save for symmetry), we can use the compact closed category diagrams together with the Frobenius algebras to directly capture both syntactic and semantic structure. Drawing triangles for lexical entries, the graphical representation of Equation 2.1 is as follows:



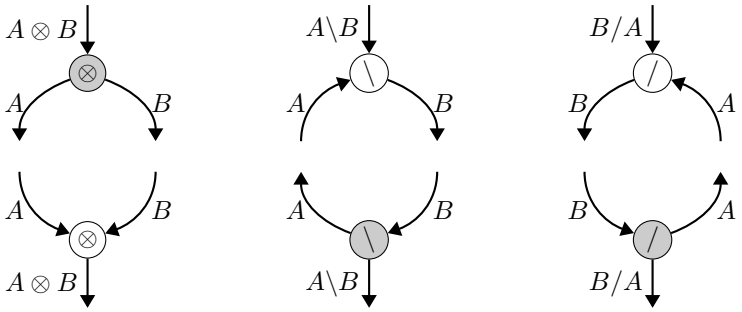
Here, we can capture both the derivational aspect of a sentence representation (the wiring below the types) and the lexical representations (the triangles/arrows above the types) in a single diagram. Some diagrams may originally contain constructions that may be simplified, for example by using the visual equations above. When a diagram is fully simplified into a normal form, like the diagram above, we can read off the simplest algebraic form of Equation 2.1:

$$(\overline{dances} \times_1 \overrightarrow{hannah}) \odot (\overline{sings} \times_1 \overrightarrow{bill}) \tag{2.2}$$

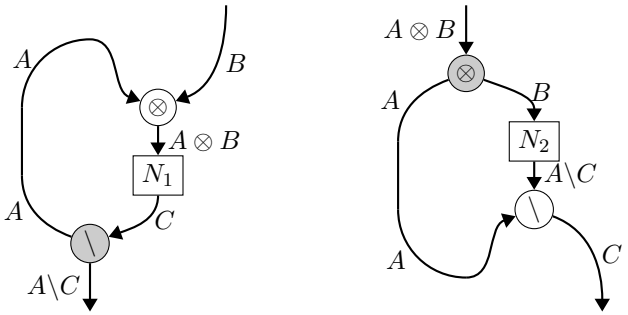
Throughout this thesis we will use these diagrams to represent the meaning of sentences in a vector based model.

String Diagrams for Lambek Calculus

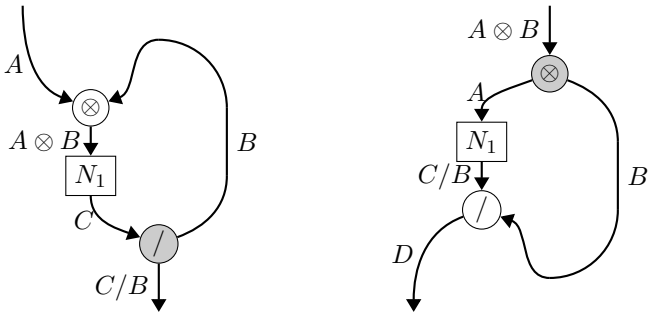
Wijnholds [Wij17] takes a different approach to string diagrams by considering *categorical proof nets*: graphical representations of the deductions in the Lambek Calculus that also satisfy the categorical axioms of a biclosed monoidal category. In short, one extends the language of string diagrams with links for each connective in the calculus (top: destructor links, bottom: constructor links):



These have the property that a destructor link attached to a constructor link is the same as an identity, whereas the inverse attachment is the same as having two loose arrows. Closure of the category is then given by mixing the $\otimes$ links with their residual links:



and



Indeed, Wijnholds [Wij17] shows that this graphical language can be defined in a fully inductive manner, bypassing the need for correctness criteria that one would have in the case of proof nets. Throughout this thesis, we will make informal use of this language, where we drop the links for $\otimes$ whenever possible; this does not automatically give a graphical language for monoidal biclosed categories as it has not been shown to be complete, but it gives a good intuition of the flow of information in a proof. For example, the proof of Figure 2.4 may be graphically presented as in Figure 2.5.

Moreover, the diagrams of this form can be fairly easily seen to be translatable to the diagrams of a compact closed category. As we interpret all the connectives using the tensor product of vector spaces, applications become tensor contractions, we can *unfold* any type into a juxtaposed depiction of wires, where the application nodes for $\backslash, /$ are interpreted using cups, as shown in Figure 2.6. In the symmetric situation of the co-application rule, we interpret it using caps.

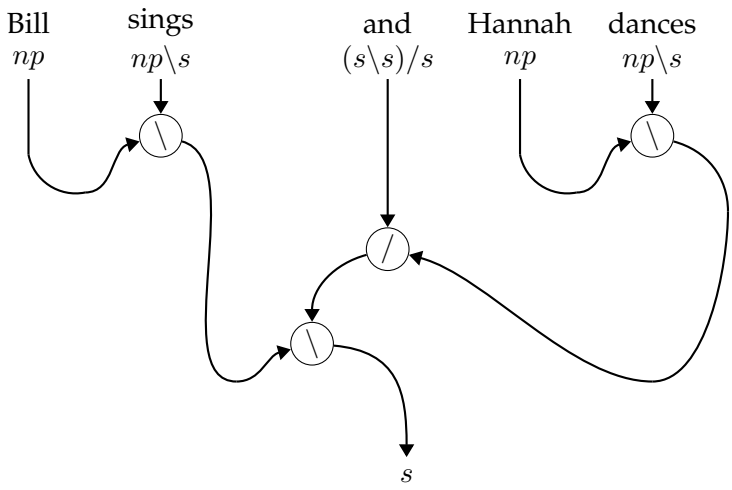


FIGURE 2.5: Information flow for the derivation in Figure 2.4.

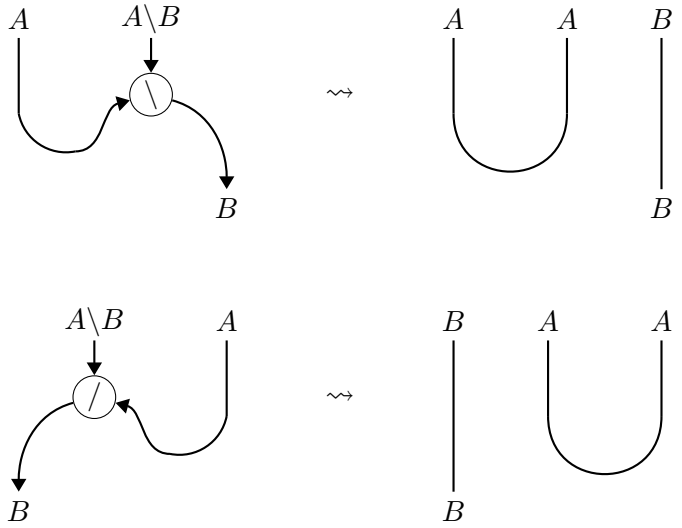


FIGURE 2.6: Translation of the Lambek diagrams for left/right application to string diagrams of a compact closed category.

Part II

Theory

Chapter 3

Ellipsis, Anaphora, and Parasitic Gaps

Chapter Abstract

The categorical model described in Chapter 2 provides a direct interpretation of type-logical grammar into vector spaces and linear maps using the language of category theory. In this chapter, we give a number of extensions of the categorical model, that handle relative pronouns, parasitic gapping, and verb phrase ellipsis with anaphora. We also introduce an alternative modelling for verb phrase ellipsis using the framework of Muskens and Sadrzadeh [MS17]. We highlight the pros and cons of both approaches, part of which rests on evaluating which we discuss in Chapter 4. Material is drawn from Moortgat and Wijnholds [MW17], Wijnholds and Sadrzadeh [WS18; WS19a], and Sadrzadeh, Moortgat, and Wijnholds [SMW19].

As discussed in Chapter 2, the categorical view on compositionality in distributional models of language relies on a direct mapping from syntax to semantics. In the pregroup case of Coecke, Sadrzadeh, and Clark [CSC10], the structure of the grammar is (nearly) the same as the vector semantics, as pregroup grammar instantiates an autonomous category, where finite-dimensional vector spaces are an instance of a compact closed category. In subsequent work of Coecke, Grefenstette, and Sadrzadeh [CGS13] it was shown that the connection between syntax and vector space semantics could be made through a *functorial passage*, and so it becomes possible to treat the Lambek Calculus [Lam58] as a grammatical underpinning for composition in vector spaces.

The abundance of work in typelogical grammar, however, already indicates that the task of modelling natural language syntax was not at an end when Lambek introduced his first grammar formalism; the work of Steedman [Ste00], Morrill and Merenciano Saladrigas [MMS96], and Moortgat [Moo97] highlights three prominent strands of typelogical grammar. All of these systems present improvements of the Lambek calculus that can treat a variety of linguistic phenomena that are not possible to address with the core Lambek Calculus, or would give undesired semantics.

Incorporating the variety of different extensions of the Lambek Calculus and merging them within the categorical framework of compositional distributional models poses several challenges; there is the treatment of Combinatory Categorial Grammar in a distributional model by Maillard, Clark, and Grefenstette [MCG14], of which parts are also reported in the PhD thesis of Edward Grefenstette [Gre13], for the symmetric system of Moortgat [Moo09] there is the treatment of Wijnholds [Wij14]. The displacement calculus of Morrill, Valentín, and Fadda [MVF11] has not been treated before; the fact that this calculus separates syntax and semantics by means of compiling in structural rules [Val14] makes it more difficult to find a faithful interpretation for the displacement connectives that govern the syntax of discontinuity in this system. In a multimodal Lambek grammar [Moo96] this problem is not present, as every (non context free) syntactic mechanism is treated by pure structural rules, allowing the grammar to be presented in the style of a *deductive system* [Lam68]. In such a system, one separates logical rules, which describe only the behaviour of the connectives of the logic, from structural rules, which describe the non-logical behaviour of the structures that the logic operates on. Taking this approach, the transition to vector space semantics is made easier, as a deductive system describes the skeletal structure of the interpreting semantic category, and we only need to find suitable operations that adhere to the structural rules of the grammar. Structural rules also make the connection between semantic operations more intuitive: to find a syntactic counterpart of the Frobenius Algebras used by Sadrzadeh, Clark, and Coecke [SCC13], one defines a structural rule that is akin to the rule of contraction one finds in classical logic. This type of argument is present in the approach of Wijnholds [Wij14], and we continue along those lines here, using a variation of the Lambek Calculus with control modalities of [Moo96]. The control modalities refer to a pair of unary operations that facilitate a *controlled* version of structural reasoning, where the application of structural rules may be licensed only on structures that involve the modalities [KM97].

In this chapter we focus on three instances of *ellipsis*, a linguistic phenomenon in which an overt syntactic element provides the semantic content for one or more syntactic elements in a sentence that are not physically realised. This very broad definition informally refers to a case in which a part of a sentence is missing, but can be recovered from the context of the sentence it occurs in.

We start off with a description of ellipsis and the particular cases of gapping with relative pronouns, parasitic gapping, and ellipsis with anaphora. We continue by presenting the Lambek Calculus with control modalities [Moo96] including its interpretation in vector spaces. We then review the use of Frobenius Algebras of previous work [SCC13; SCC14] and how they neatly combine with the modalities to give an account of pronoun relativisation in English and Dutch (material from Moortgat and Wijnholds [MW17]). Then, we extend the lexical approach to copying behaviour using Frobenius Algebras to the case of parasitic gapping (material from Sadrzadeh, Moortgat, and Wijnholds [SMW19]). Finally, we argue that the

case of verb phrase ellipsis with anaphora does not lend itself very well for a lexical copying approach. We propose a modification of the modal Lambek Calculus with a controlled form of contraction, and give an analysis of verb phrase ellipsis with anaphora. For this we develop both a categorical and a lambda-based interpretation (material from Wijnholds and Sadrzadeh [WS18] and Wijnholds and Sadrzadeh [WS19a]).

3.1 Two Types of Ellipsis

Ellipsis in general is the phenomenon of omission, and we will loosely define it as the phenomenon in which an overt syntactic element provides the semantic content for one or more syntactic elements in a sentence that are not physically realised. In other words, we say that a phrase is elliptical whenever is has missing parts, that can be however be recovered within the context of the phrase.

Ellipsis is a very broad phenomenon and can be classified into several types (see [Mer01] for more elaborate discussion). In this chapter we focus on two broad types of ellipsis: the case of (parasitic) gapping, and the case of verb phrase ellipsis with anaphora. The case of gapping is directly relevant for a compositional distributional analysis as the work of Sadrzadeh, Clark, and Coecke [SCC13] models pronoun relativisation in English, which is a type of gapping. As we will see in this chapter, gapping has a natural extension into a parasitic variant, which we model in a very similar way as has been done for relative pronouns. On the other hand, verb phrase ellipsis is very pervasive in natural language (see [BS11] for a corpus investigation), but poses a different challenge: where the cases of gapping presented below always come with a relative pronoun and/or a coordinator, the examples of verb phrase ellipsis above show that it is the auxiliary verb that mediates the elided content.

Some examples of gapping with relative pronouns are given in Equation 3.1, where examples (a) and (b) respectively have an object- and subject-relative reading. A more complex variant is *parasitic* gapping, given in example (c). Here, there is a gap that is dependent on another, primary gap. A more general case of gapping is example (d) that does not contain a relative pronoun, but just the coordinator 'and'.

$$
\begin{array}{ll}
a & \text{papers that Bob rejected } \_ \text{ (immediately)} \qquad \text{(obj rel)} \\
b & \text{animals who } \_ \text{ eat humans (ravenously)} \qquad \text{(subj rel)} \\
c & \text{papers that reviewers rejected } \_ \text{ without reading } \_ \text{ (carefully)} \\
d & \text{Mary ate potatoes and John } \_ \text{ salad}
\end{array}
\tag{3.1}
$$

In both examples (a) and (b), there is the relative pronoun that has a clear role in the construction of the relative clause. In example, (c), there additionally is the other

functional word 'without', that can serve as a mediator for the second, parasitic, gap. In example (d), the gap is not to be filled by a noun phrase, but by the single verb.

Verb phrase ellipsis — as the name suggests — covers cases in which the elided element is a verb phrase, but has the requirement that an auxiliary verb is present, indicating the location of the elided verb phrase. This is one of the aspects that distinguishes verb phrase ellipsis from the gapping examples above. The below examples are all instances of verb phrase ellipsis, where the location of the elided verb phrase is *marked* by an auxiliary verb. In examples (e) and (f) there is an explicit coordinator present, but example (g) shows that this need not be the case. In all cases, the examples are ideally in bidirectional entailment with their resolved variants. For example, sentence (e) should entail the sentence "Alice drinks and Bob drinks" and vice versa.

$$\begin{array}{ll} e & \text{Alice drinks and Bob does too} \\ f & \text{Kim wears a hat but Sandy does not} \\ g & \text{John slept, Mary did too} \end{array} \tag{3.2}$$

A more complicated example of verb phrase ellipsis is one where in addition to the verb phrase ellipsis site there is also an anaphora present. Such an example induces an ambiguity, as in the example of Equation 3.3, where the ambiguous phrase (h) has two readings (i) and (j).

$$\begin{array}{lll} h & \text{"Gary loves his code and Bill does too"} & \text{(ambiguous)} \\ i & \text{"Gary loves Gary's code and Bill loves Gary's code"} & \text{(strict)} \\ j & \text{"Gary loves Gary's code and Bill loves Bill's code"} & \text{(sloppy)} \end{array} \tag{3.3}$$

The case of ellipsis traditionally has been approached both as a syntactic problem within categorial grammars [KL17; Jäg06; Jäg98; MMS96; Hen95] as well as a semantic problem by directly appealing to their lambda calculi term logics [DSP91; Kem+15]. The research within categorial grammar either suggests that elliptic phenomena should be treated using a specific controlled form of copying of information at the antecedent and moving it to the site of ellipsis, e..g in [Jäg98; MMS96], or by maintaining a non-directional functional type (meaning that it is not sensitive to where its argument occurs, before or after it), which is backward/forward looking, e.g. in [KL17; Jäg06]. The first proposal can also be implemented using different modal Lambek Calculi, e.g. that developed in [Moo97] and the second one using Displacement Calculus [MV10]; Abstract Categorial Grammars of [Mus03; Gro01], which allow for a separation of syntax and semantics within a categorial grammar and allow for freedom of copying and movement at the semantic side, can also be employed.

Modelling ellipsis in a compositional distributional model poses a number of challenges: first, in the process of create phrase meaning from individual word meanings, one has to associate analytical lexical specifications to functional words such as coordinators, relative pronouns, and ellipsis markers (in the case of verb

phrase ellipsis), since they should not be addressed distributionally (lexical semantics). The second challenge is that the semantic representations we wish to obtain are non-linear: in the case of relative pronouns, the semantic information of the head noun should be reused to serve as the subject of the verb phrase in the body of the relative clause; when a parasitic gap is present, an additional reuse should be licensed; in the case of verb phrase ellipsis, the main verb phrase needs to be made available to the auxiliary verb in order to complete the final semantics. Somehow a model needs to account for how these non-linearities are obtained (derivational semantics).

As argued in the introduction to this chapter, for a compositional distributional modelling we want to start with a system that makes a clear separation between logical and structural rules, as this allows a neat interface with vector semantics in a categorical setting. Therefore, we use the Lambek Calculus with control modalities of Moortgat [Moo96]. Given the freedom to define new structural rules while keeping the base system intact, we will use (variants of) this system throughout this chapter. For relative pronouns, we use the original system with leftward or rightward extraction rules to model pronoun relativisation in English and Dutch. We extend this setting to the case of parasitic gaps by relying on lexical polymorphism of the coordinator 'without'. Then, we introduce a variant of the proof system that does have access to a controlled form of contraction, which allows us to copy in syntax, a mechanism we can then use to model verb phrase ellipsis with anaphora.

3.2 Categorical Distributional Semantics with Lambek Calculus and Modalities

We start out with the definition of $\mathbf{NL}_\Diamond$, the multimodal extension of the Lambek Calculus [Moo96]. Similar to Chapter 2, we present this system in a 'deductive systems' format, where one defines axioms and inference rules. The reason to do so is that this style of presentation directly corresponds to the presentation style commonly used in category theory.

The base component of $\mathbf{NL}_\Diamond$ is the nonassociative Lambek Calculus, already defined in Section 2.1.4. The philosopy behind unary modalities as an addition to the nonassociative Lambek Calculus is that of structural control: full blown associativity or commutativity in the grammar logic would overgenerate, i.e. render all kinds of ungrammatical sentences grammatical; but many natural language phenomena do require some form of restructuring of linguistic resources before semantic analysis proceeds: besides the examples of ellipsis given in the previous section, cases of long-distance dependencies warrant the exchange of information between non-adjacent positions. The unary modalities allow one to control the behaviour of the grammar by licensing or blocking certain applications of structural rules. The mathematical details have been worked out by Kurtonina and Moortgat [KM97], where

$$\frac{}{1_A : A \longrightarrow A} \qquad \frac{f : A \longrightarrow B \quad g : B \longrightarrow C}{g \circ f : A \longrightarrow C}$$

$$\frac{f : \Diamond A \longrightarrow B}{\nabla f : A \longrightarrow \Box B} \qquad \frac{f : A \otimes B \longrightarrow C}{\triangleright f : A \longrightarrow C/B} \qquad \frac{f : A \otimes B \longrightarrow C}{\triangleleft f : B \longrightarrow A\backslash C}$$

$$\frac{g : A \longrightarrow \Box B}{\nabla^{-1} g : \Diamond A \longrightarrow B} \qquad \frac{g : A \longrightarrow C/B}{\triangleright^{-1} g : A \otimes B \longrightarrow C} \qquad \frac{g : B \longrightarrow A\backslash C}{\triangleleft^{-1} g : A \otimes B \longrightarrow C}$$

$$\alpha^l_\Diamond : \Diamond A \otimes (B \otimes C) \longrightarrow (\Diamond A \otimes B) \otimes C \qquad \alpha^r_\Diamond : (A \otimes B) \otimes \Diamond C \longrightarrow A \otimes (B \otimes \Diamond C)$$

$$\sigma^l_\Diamond : \Diamond A \otimes (B \otimes C) \longrightarrow B \otimes (\Diamond A \otimes C) \qquad \sigma^r_\Diamond : (A \otimes B) \otimes \Diamond C \longrightarrow (A \otimes \Diamond C) \otimes B$$

FIGURE 3.1: The logic $\mathbf{NL}_\Diamond$, presented as a deductive system.

$$\frac{f : A \longrightarrow B}{\Diamond f : \Diamond A \longrightarrow \Diamond B} \qquad \frac{f : A \longrightarrow B}{\Box f : \Box A \longrightarrow \Box B}$$

$$\frac{f : A \longrightarrow B \quad g : C \longrightarrow D}{f/g : A/D \longrightarrow B/C} \qquad \frac{f : A \longrightarrow B \quad g : C \longrightarrow D}{f\backslash g : B\backslash C \longrightarrow A\backslash D}$$

$$\frac{f : (\Diamond A \otimes B) \otimes C \longrightarrow D}{\widehat{\alpha}^l_\Diamond f : \Diamond A \otimes (B \otimes C) \longrightarrow D} \qquad \frac{f : A \otimes (B \otimes \Diamond C) \longrightarrow D}{\widehat{\alpha}^r_\Diamond f : (A \otimes B) \otimes \Diamond C \longrightarrow D}$$

$$\frac{f : B \otimes (A \otimes \Diamond C) \longrightarrow D}{\widehat{\sigma}^l_\Diamond f : \Diamond A \otimes (B \otimes C) \longrightarrow D} \qquad \frac{f : (A \otimes \Diamond C) \otimes B \longrightarrow D}{\widehat{\sigma}^r_\Diamond f : (A \otimes B) \otimes \Diamond C \longrightarrow D}$$

FIGURE 3.2: Monotonicity rules, and structural rules in rule form.

the modalities allow one to embed the associative Lambek Calculus in the nonassociative variant, given the presence of modalities allowing to control the associativity. Vice versa, the nonassociative Lambek Calculus can be embedded in the associative calculus, by decorating formulas in such a way that associativity cannot be applied. In this chapter we assume the first variant where the modalities allow to control specific structural behaviour desirable for linguistic applications. Formally, one adds two unary connectives $\Diamond, \Box$, which exhibit a residuation, or, categorically speaking, a adjunction where $\Diamond A \to B$ iff $A \to \Box B$. We present the full calculus in Figure 3.1, with derived monotonicity rules and rule form of the structural rules displayed in Figure 3.2.

Interpretating the modalities The modalities of the system $\mathbf{NL}_\Diamond$ exhibit only a syntactic role when decorating formulas: they either license or block structural control. Hence, it is not surprising that the interpretation of the modalities is vacuous on the level of types. That is, we have $\lceil \Diamond A \rceil = \lceil \Box A \rceil = \lceil A \rceil$, while leaving the binary type-forming operators as in Section 2.1.6. On the level of proofs, the interpretation of the monotonicity rules for $\Diamond, \Box$ become identities: $\lceil \Diamond f \rceil = \lceil \Box f \rceil = \lceil f \rceil$. We moreover

need to give an interpretation of the structural rules in a compact closed category. As defined in Chapter 2, a compact closed category is symmetric monoidal. Hence we can interpret the structural rules from 3.1 using the associator and the symmetry maps of the compact closed category. For the controlled associativity maps, this means simply a rebracketing:

$$\lceil \widehat{\alpha}^r_\diamond f \rceil = (\lceil A \rceil \otimes \lceil B \rceil) \otimes \lceil C \rceil \xrightarrow{\alpha} \lceil A \rceil \otimes (\lceil B \rceil \otimes \lceil C \rceil) \xrightarrow{\lceil f \rceil} \lceil D \rceil$$

$$\lceil \widehat{\alpha}^l_\diamond f \rceil = \lceil A \rceil \otimes (\lceil B \rceil \otimes \lceil C \rceil) \xrightarrow{\alpha^{-1}} (\lceil A \rceil \otimes \lceil B \rceil) \otimes \lceil C \rceil \xrightarrow{\lceil f \rceil} \lceil D \rceil$$

For the controlled commutativity, we rely on bracketing as well as reordering of elements:

$$\lceil \widehat{\sigma}^r_\diamond f \rceil = \qquad\qquad \lceil \widehat{\sigma}^l_\diamond f \rceil =$$

$$\begin{array}{c} (\lceil A \rceil \otimes \lceil B \rceil) \otimes \lceil C \rceil \\ \downarrow{\scriptstyle \alpha} \\ \lceil A \rceil \otimes (\lceil B \rceil \otimes \lceil C \rceil) \\ \downarrow{\scriptstyle 1_{\lceil A \rceil} \otimes \sigma_{\lceil B \rceil, \lceil C \rceil}} \\ \lceil A \rceil \otimes (\lceil C \rceil \otimes \lceil B \rceil) \\ \downarrow{\scriptstyle \alpha^{-1}} \\ (\lceil A \rceil \otimes \lceil C \rceil) \otimes \lceil B \rceil \\ \downarrow{\scriptstyle \lceil f \rceil} \\ \lceil D \rceil \end{array} \qquad \begin{array}{c} \lceil A \rceil \otimes (\lceil B \rceil \otimes \lceil C \rceil) \\ \downarrow{\scriptstyle \alpha^{-1}} \\ (\lceil A \rceil \otimes \lceil B \rceil) \otimes \lceil C \rceil \\ \downarrow{\scriptstyle \sigma_{\lceil A \rceil, \lceil B \rceil} \otimes 1_{\lceil C \rceil}} \\ (\lceil B \rceil \otimes \lceil A \rceil) \otimes \lceil C \rceil \\ \downarrow{\scriptstyle \alpha} \\ \lceil B \rceil \otimes (\lceil A \rceil \otimes \lceil C \rceil) \\ \downarrow{\scriptstyle \lceil f \rceil} \\ \lceil D \rceil \end{array}$$

This is the basic system that allows us to model pronoun relativisation in an SVO language like English (using the rightward extraction rules) and a SOV language like Dutch (using the leftward extraction rules). In the next section we show how we can derive such cases using $\mathbf{NL}_\diamond$, and how we can give a concrete vector semantics which is in exact agreement with the modelling of Sadrzadeh, Clark, and Coecke [SCC13].

3.3 Lexicon versus Derivation in Pronoun Relativisation

We demonstrate the application of the system $\mathbf{NL}_\diamond$ on the case of relative pronouns in English and Dutch. In prior work, Sadrzadeh, Clark, and Coecke [SCC13] show that Frobenius Algebras have applications for pronoun relativisation in English: the

meanings of 'who' and 'whom' serve as discriminators between a subject- or object-relative reading in English, as the examples below show:

$$\begin{array}{lll} a & \text{humans who eat animals} & \text{(subject rel)} \\ b & \text{humans whom animals eat} & \text{(object rel)} \end{array} \tag{3.4}$$

Before proceeding to explain the Frobenius Algebra approach, and our incarnation using the system $\mathbf{NL}_\Diamond$, we review a formal semantics account of relative pronouns. Later on, we will discuss the similarities and contrasts with the vector semantics account that we present below.

Formal Semantics for Relative Pronouns

In the formal semantics account, the interpretation homomorphism sends syntactic types to their semantic counterparts. Syntactic types are built from atoms, for example s, np, n for sentences, noun phrases and common nouns; assuming semantic atoms e, t and function types built from them, one can set $\lfloor s \rfloor = t$, $\lfloor np \rfloor = e$, $\lfloor n \rfloor = e \to t$, and $\lfloor A/B \rfloor = \lfloor B \backslash A \rfloor = \lfloor B \rfloor \to \lfloor A \rfloor$, like in Section 2.1.5. Each semantic type A is assigned an interpretation domain D_A, with $D_e = E$, for some non-empty set E (the discussion domain), $D_t = \{0, 1\}$ (truth values), and $D_{A \to B}$ functions from D_A to D_B.

In this setup, a syntactic derivation $A_1, \ldots A_n \Rightarrow B$ is interpreted by means of a linear lambda term M of type $\lceil B \rceil$, with parameters x_i of type $\lceil A_i \rceil$ — linearity resulting from the fact that the syntactic source doesn't provide the copying/deletion operations associated with the structural rules of Contraction and Weakening.

The proof term M is an instruction for meaning assembly that abstracts from lexical semantics. In (3.5) below, one finds the proof terms for English subject (a) and object (b) relativisation. The parameter w stands for the head noun, f for the verb, y and z for its object and subject arguments; parameter x for the relative pronoun has type $(e \to t) \to (e \to t) \to e \to t$.

$$\begin{array}{lll} (a) & n, (n\backslash n)/(np\backslash s), (np\backslash s)/np, np \Rightarrow n & (x_{who}\ \lambda z^e.(f^{e\to e\to t}\ y^e\ z^e)\ w^{e\to t}) \\ (b) & n, (n\backslash n)/(s/np), np, (np\backslash s)/np \Rightarrow n & (x_{who}\ \lambda y^e.(f^{e\to e\to t}\ y^e\ z^e)\ w^{e\to t}) \end{array} \tag{3.5}$$

To obtain the interpretation of "humans who eat animals" vs "humans who(m) animals eat", one substitutes lexical meanings for the parameters of the proof terms. In the case of the open class items 'humans', 'eat', 'animals', these will be non-logical constants with an interpretation depending on the model. For the relative pronoun, we substitute an interpretation independent of the model, expressed in terms of the logical constant $\wedge$, leading to the final interpretations of (3.7), after normalisation.

$$x_{who} := \lambda x^{e\to t} \lambda y^{e\to t} \lambda z^e.((x\ z) \wedge ((y\ z)) \tag{3.6}$$

$$\begin{array}{ll} (a) & \lambda x.((\text{HUMAN } x) \wedge (\text{EAT ANIMAL } x)) \\ (b) & \lambda x.((\text{HUMAN } x) \wedge (\text{EAT } x \text{ ANIMAL})) \end{array} \tag{3.7}$$

Notice that the lexical meaning recipe for the relative pronoun goes beyond linearity: to express the set intersection interpretation, the bound z variable is copied over the conjuncts of $\wedge$. By encapsulating this copying operation in the lexical semantics, one avoids compromising the derivational semantics.

Frobenius semantics for pronoun relativisation

To treat the cases of subject-relative and object-relative pronoun interpretations, the Frobenius copying map Δ and the deletion map ι are used in the lexical meaning of the relative pronouns 'who' and 'whom'. The pregroup type for the subject-relative pronoun 'who' is defined as $n^r \cdot n \cdot s^l \cdot np$ with mapping to the semantic type $N \otimes N \otimes S \otimes N$. The semantic effect of the relative pronoun here is to merge the information contained in the head noun (*humans*) with the body of the relative clause by means of the merging map μ. Since the body of the relative clause constitutes a verb phrase and not a noun, the sentence dimension of the verb phrase is summed over by means of the ι map. So, the formal description of the map for 'who' is

$$(id_N \otimes \mu_N \otimes \iota_S \otimes id_N) \circ (\eta_N \otimes \eta_N)$$

The definition of the object-relative pronoun 'whom' proceeds in the same way, except that the pregroup type is different since the noun and verb in the body of the relative clause are now swapped: the syntactic type is $n^r \cdot n \cdot np^{ll} \cdot s^l$ which translates to the semantic type $N \otimes N \otimes N \otimes S$. The map for this pronoun is then given by

$$(id_N \otimes \mu_N \otimes id_N \otimes \iota_S) \circ (\eta_N \otimes \eta_N)$$

These formal maps can be graphically depicted in the string diagrammatic language we introduced in Section 2.1.7, where the μ map is indicated by a white circle with three outgoing wires, and the ι map by a white circle with a single outgoing wire:

Here, the pregroup types may be seen as a translation of Lambek types $(n\backslash n)/(np\backslash s)$ (*who*) and $(n\backslash n)/(s/np)$ (*whom*), and so the diagrams above state that a matrix in the space $S \otimes N$ or $N \otimes S$ (a verb phrase) will be projected down into a noun space, after which the resulting vector (rightmost N) is multiplied (the white dot in the diagrams) with the vector of a noun (leftmost N) to give a vector (middle N) that is the coordination of the head noun with the relative clause. Concretely, this gives the

below algebraic realisation of the subject- and object relative clauses.

$$\begin{array}{lll} a & \overrightarrow{humans} \odot \iota_S(\overline{eat} \times \overrightarrow{animals}) & \text{(subject rel)} \\ b & \overrightarrow{humans} \odot \iota_S(\overrightarrow{animals}^{\top} \times \overline{eat}) & \text{(object rel)} \end{array} \tag{3.8}$$

Using the diagrams for the relative pronouns above, one can model the subject relative reading with the diagram in Figure 3.3, and the object relative reading with the diagram in Figure 3.4.

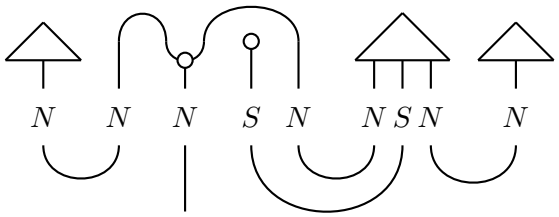


FIGURE 3.3: The semantic information flow of a subject relative clause.

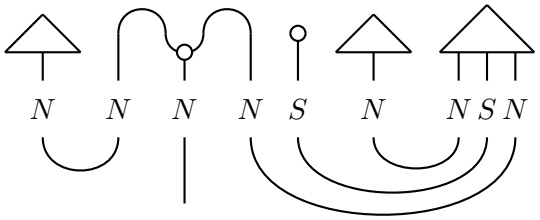


FIGURE 3.4: The semantic information flow of an object relative clause.

The absence of any form of commutativity in pregroups results in the inability to reorder any type information. As a result, the analysis given above is not robust against *non-peripheral* cases of pronoun relativisation. A prime example of this is the case of an adverbial modification: *humans whom animals eat ravenously*. Here, the usual typing for the adverb, $(np\backslash s)\backslash(np\backslash s)$ needs access to the verb phrase which is only eligible for combining until it has itself been combined with the relative pronoun first. To mend this and give a fully derivational semantics for relative pronouns in English, we make use of the English version of $\mathbf{NL}_\Diamond$, which uses the rightward extraction rules. This system allows for a controlled version of commutativity, that interacts properly with the type of the relative pronoun. We show how one can use $\mathbf{NL}_\Diamond$ types for the relative pronoun, how we can derive proofs for the cases above, and how types and proofs translate to give exactly the diagrams in Figures 3.3 and 3.4. For these cases, the proof system allows reasoning about a hypothetical noun phrase that takes the place of the subject (or object) of the main verb in the relative clause, to be consumed by the representation of the relative pronoun, linking it back to the head noun. For more involved examples where the hypothetical noun

phrase actually occurs not at the end but within the subclause, the (commutativity) extraction rules are needed.

First, we define the $\mathbf{NL}_\Diamond$ types with their pregroup translation in Figure 3.5 for the case of English relative pronouns. For the head noun and noun phrase in the relative clause body we assume the standard typings n and np, the verb is represented by the type $(np\backslash s)/np$. The adverb 'ravenously' is assigned the higher-order type $(np\backslash s)\backslash(np\backslash s)$: given a verb phrase, it will act as a modifier of that verb phrase, hence on the level of types it acts as an identity map. As the Lambek type encodes a higher-order function, its translation to pregroup types switches the order of the basic types, with an iterated right adjoint on the leftmost np type reflecting the fact that the adverb is a verb phrase modifier. The typing for the relative pronouns in-

Word	Lambek type	pregroup type
humans	n	n
who	$(n\backslash n)/(\Diamond\Box np\backslash s)$	$n^r \cdot n \cdot s^l \cdot np$
whom	$(n\backslash n)/(s/\Diamond\Box np)$	$n^r \cdot n \cdot np^{ll} \cdot s^l$
animals	np	np
eat	$(np\backslash s)/np$	$np^r \cdot s \cdot np^l$
ravenously	$(np\backslash s)\backslash(np\backslash s)$	$s^r \cdot np^{rr} \cdot np^r \cdot s$

FIGURE 3.5: A lexicon in $\mathbf{NL}_\Diamond$ for pronoun relativisation in English, with pregroup translation.

volves the unary control modalities: the type $\Diamond\Box np\backslash s$ (resp. $s/\Diamond\Box np$) expresses a verb phrase where the noun phrase component may occur in any rightward position. In other words, the typing for the relative pronouns encodes the fact that an incomplete relative clause body is expected in order to complete the full relative clause. The direction of the missing head noun is dictated by the fact that 'who' is a subject-relative pronoun and 'whom' is an object-relative pronoun. The pregroup image of the types correspond to the semantic interpretation in the diagrams given in Sadrzadeh, Clark, and Coecke [SCC13], here in Figures 3.3 and 3.4.

The derivations for the subject relative and object relative cases are shown in Figures 3.6 and 3.7. In the subject relative case, the transitive verb can be combined with its object to form the body of the relative clause, which is now missing the head noun of the full phrase. By the fact that there is a proof of $\Diamond\Box np \to np$ the relative pronoun can consume the verb phrase of the body of the relative clause, to then be combined with the head noun. The object relative case is different, as the non-associative nature of the types does not allow the main verb to be combined with the subject in the body of the relative clause, which is why here the use of the rightward controlled associativity postulate comes in to rebracket and let the derivation go through. The controlled symmetry is needed when the location of the hypothetical noun phrase (signified by $\Diamond\Box np$) is not fully peripheral, as in "men that animals eat ravenously". For this, we give the derivation in Figure 3.8.

$$\dfrac{\dfrac{\overline{n \to n}\ 1_n \quad \overline{n \to n}\ 1_n \setminus}{n\backslash n \to n\backslash n} \quad \dfrac{\dfrac{\dfrac{\overline{np \to np}\ 1_{np}}{\Box np \to \Box np}\ \Box}{\Diamond\Box np \to np}\ \nabla^{-1} \quad \overline{s \to s}\ 1_s \setminus}{np\backslash s \to \Diamond\Box np\backslash s} /}{(n\backslash n)/(\Diamond\Box np\backslash s) \to (n\backslash n)/(np\backslash s)} \\ \dfrac{}{((n\backslash n)/(\Diamond\Box np\backslash s)) \otimes (np\backslash s) \to n\backslash n}\ \triangleright^{-1} \quad \overline{np \to np}\ 1_{np} \\ \dfrac{}{(np\backslash s) \to ((n\backslash n)/(\Diamond\Box np\backslash s))\backslash(n\backslash n)} \\ \dfrac{}{(np\backslash s)/np \to (((n\backslash n)/(\Diamond\Box np\backslash s))\backslash(n\backslash n))/np} / \\ \dfrac{}{((np\backslash s)/np) \otimes np \to ((n\backslash n)/(\Diamond\Box np\backslash s))\backslash(n\backslash n)}\ \triangleright^{-1} \\ \dfrac{}{((n\backslash n)/(\Diamond\Box np\backslash s)) \otimes (((np\backslash s)/np) \otimes np) \to n\backslash n}\ \triangleleft^{-1} \\ \underset{humans}{n} \otimes (((n\backslash n)/(\Diamond\Box np\backslash s)) \otimes (((np\backslash s)/np) \otimes \underset{animals}{np})) \to n \quad \triangleleft^{-1}$$

FIGURE 3.6: Derivation for "humans who eat animals".

$$\dfrac{\dfrac{\overline{n \to n}\ 1_n \quad \overline{n \to n}\ 1_n \setminus}{n\backslash n \to n\backslash n} \quad \dfrac{\dfrac{\dfrac{\dfrac{\overline{np \to np}\ 1_{np} \quad \overline{s \to s}\ 1_s \setminus}{np\backslash s \to np\backslash s} \quad \dfrac{\dfrac{\overline{np \to np}\ 1_{np}}{\Box np \to \Box np}\ \Box}{\Diamond\Box np \to np}\ \nabla^{-1}}{(np\backslash s)/np \to (np\backslash s)/\Diamond\Box np} / \\ \dfrac{}{(np\backslash s)/np \otimes \Diamond\Box np \to np\backslash s}\ \triangleright^{-1} \\ \dfrac{}{np \otimes ((np\backslash s)/np \otimes \Diamond\Box np) \to s}\ \triangleleft^{-1} \\ \dfrac{}{(np \otimes (np\backslash s)/np) \otimes \Diamond\Box np \to s}\ \widehat{\alpha}^r_{\diamond} \\ \dfrac{}{np \otimes (np\backslash s)/np \to s/\Diamond\Box np}\ \triangleright \\ \dfrac{}{(n\backslash n)/(s/\Diamond\Box np) \to (n\backslash n)/(np \otimes ((np\backslash s)/np))} / \\ \dfrac{}{((n\backslash n)/(s/\Diamond\Box np)) \otimes (np \otimes ((np\backslash s)/np)) \to n\backslash n}\ \triangleright^{-1} \\ \underset{humans}{n} \otimes (((n\backslash n)/(s/\Diamond\Box np)) \otimes (\underset{animals}{np} \otimes ((np\backslash s)/np))) \to n \quad \triangleleft^{-1}$$

FIGURE 3.7: Derivation for "humans whom animals eat".

$$\dfrac{\dfrac{\overline{n \to n}\ 1_n \quad \overline{n \to n}\ 1_n \setminus}{n\backslash n \to n\backslash n} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{np \to np}\ 1_{np} \quad \overline{s \to s}\ 1_s \setminus}{np\backslash s \to np\backslash s} \quad \dfrac{\dfrac{\overline{np \to np}\ 1_{np}}{\Box np \to \Box np}\ \Box}{\Diamond\Box np \to np}\ \nabla^{-1}}{(np\backslash s)/np \to (np\backslash s)/\Diamond\Box np} / \quad \dfrac{\overline{np \to np}\ 1_{np} \quad \overline{s \to s}\ 1_s}{np\backslash s \to np\backslash s} \setminus}{(np\backslash s)/np \otimes \Diamond\Box np \to np\backslash s}\ \triangleright^{-1} \\ \dfrac{}{(np\backslash s)\backslash(np\backslash s) \to ((np\backslash s)/np \otimes \Diamond\Box np)\backslash(np\backslash s)}\ \triangleleft^{-1} \\ \dfrac{}{((np\backslash s)/np \otimes \Diamond\Box np) \otimes (np\backslash s)\backslash(np\backslash s) \to np\backslash s}\ \widehat{\sigma}^r_{\diamond} \\ \dfrac{}{((np\backslash s)/np \otimes (np\backslash s)\backslash(np\backslash s)) \otimes \Diamond\Box np \to np\backslash s}\ \triangleleft^{-1} \\ \dfrac{}{np \otimes (((np\backslash s)/np \otimes (np\backslash s)\backslash(np\backslash s)) \otimes \Diamond\Box np) \to s}\ \widehat{\alpha}^r_{\diamond} \\ \dfrac{}{(np \otimes ((np\backslash s)/np \otimes (np\backslash s)\backslash(np\backslash s))) \otimes \Diamond\Box np \to s}\ \triangleright \\ \dfrac{}{np \otimes ((np\backslash s)/np \otimes (np\backslash s)\backslash(np\backslash s)) \to s/\Diamond\Box np} / \\ \dfrac{}{(n\backslash n)/(s/\Diamond\Box np) \to (n\backslash n)/(np \otimes ((np\backslash s)/np \otimes (np\backslash s)\backslash(np\backslash s)))} \\ \dfrac{}{((n\backslash n)/(s/\Diamond\Box np)) \otimes (np \otimes (((np\backslash s)/np) \otimes (np\backslash s)\backslash(np\backslash s))) \to n\backslash n}\ \triangleright^{-1} \\ \underset{humans}{n} \otimes ((n\backslash n)/(s/\Diamond\Box np) \otimes (\underset{animals}{np} \otimes ((np\backslash s)/np) \otimes (np\backslash s)\backslash(np\backslash s)))) \to n \quad \triangleleft^{-1}$$

FIGURE 3.8: Derivation for "humans whom animals eat ravenously".

FIGURE 3.9: Syntactic and semantic information flow for a subject relative reading (above) and an object relative reading (below) in English.

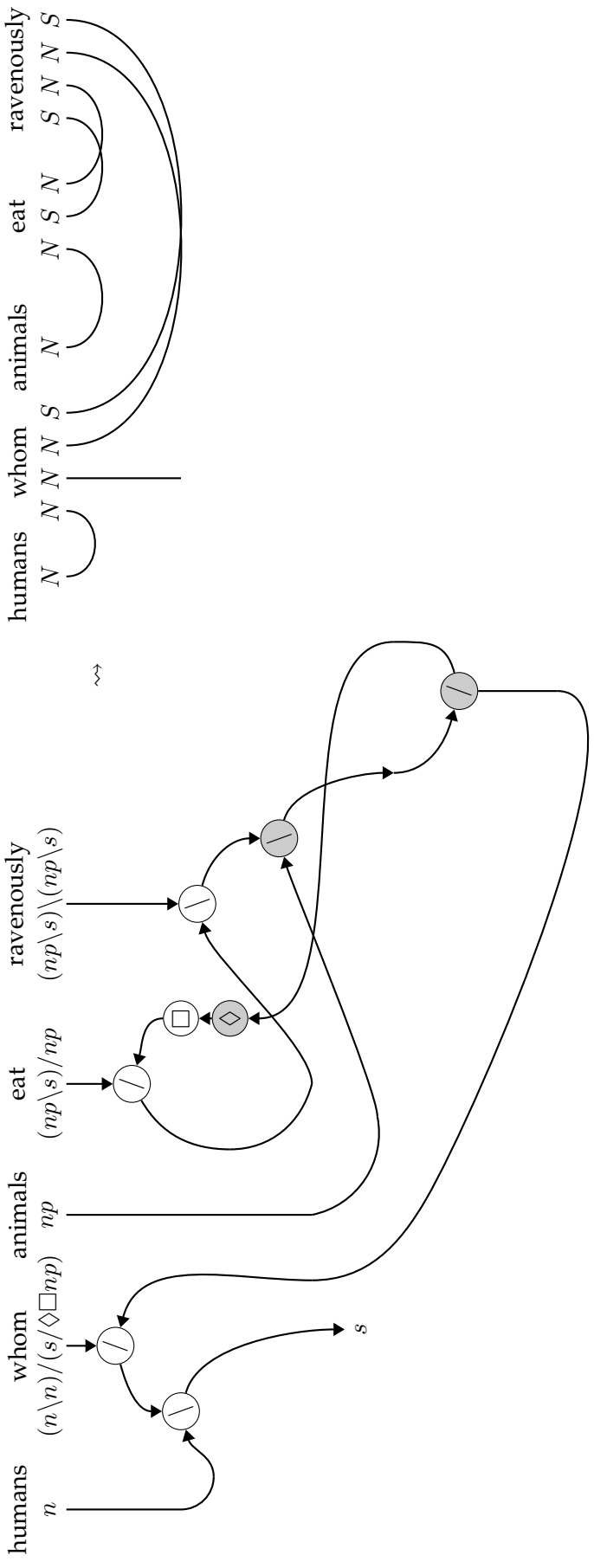


FIGURE 3.10: Syntactic and semantic information flow for a non-peripheral object relative reading in English.

We wish to represent these derivations in some graphical notation to clarify the flow of information. For this, we use the diagrammatic notation introduced in Section 2.1.7, as it provides a coherent graphical language for the categorical formulation of the calculus we use [Wij17]. Here we give the diagrams toegether with their translation to the string diagrams of a compact closed category. For the simple relative pronoun cases they are given in Figure 3.9, where the translated string diagrams in fact coincide with those in Figures 3.3 and 3.4 after 'plugging in' the lexical specification of the constituent words. For the more complex example involving an adverbal modification the diagrams are given in Figure 3.10. In the diagram in Figure 3.9 we can see how the hypothetical argument may be introduced by preceding the np arrow by a $\Diamond$ and a $\Box$ link, in the top figure it occupies the subject position, in the bottom it takes object position; after introducing the hypothetical noun phrase we use a constructor link for $\backslash$ to 'construct' the expected type for the relative pronoun. In the case of the gap being followed by an adjective, where one requires restructuring of the hypothetical noun phrase, the fact that the $\Diamond\Box np$ arrow is present allows us to reroute its wire to a rightmost position where it will again be used to construct the desired type $s/\Diamond\Box np$.

3.3.1 Dutch Pronoun Relativisation

We now demonstrate the application of $\mathbf{NL}_\Diamond$ to the case of Dutch. Being a verb final language, the situation of pronoun relativisation for Dutch is different to English: relative clauses are now ambiguous between a subject and an object relativisation reading. This is shown in the phrase below, which is ambiguous between the two distinct meanings in a and b above (repeated here):

$$\begin{array}{lll} a & \text{humans who eat animals} & \text{(subject rel)} \\ b & \text{humans whom animals eat} & \text{(object rel)} \\ c & \text{mensen die dieren eten} & \text{(ambiguous)} \end{array} \tag{3.9}$$

For Dutch, then, the goal is to find a single type assignment for the relative pronoun 'die' that allows for simultaneous derivation of both reading a and b. Assigning similar types to the nouns and verbs, n for 'mensen', np for 'dieren', and $np\backslash(np\backslash s)$ for the verb 'eten', reflecting the verb final nature of the verb usage in a pronoun relative clause. The single type assignment for 'die' is the same as for the relative pronoun 'that' in English: $(n\backslash n)/(\Diamond\Box np\backslash s)$. The inherent ambiguity of a typical pronoun relative clause like the one in c is now encoded by means of *derivational ambiguity*, multiple proofs for the same surface form, though with the two different readings as a result.

The derivations agree on the initial steps

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{n \longrightarrow n} \quad \overline{n \longrightarrow n}}{n\backslash n \longrightarrow n\backslash n}\backslash \quad \dfrac{\vdots}{np \otimes (np\backslash(np\backslash s)) \longrightarrow \Diamond\Box np\backslash s}}{(n\backslash n)/(\Diamond\Box np\backslash s) \longrightarrow (n\backslash n)/(np \otimes (np\backslash(np\backslash s)))}/}{((n\backslash n)/(\Diamond\Box np\backslash s)) \otimes (np \otimes (np\backslash(np\backslash s))) \longrightarrow n\backslash n}\triangleright^{-1}}{n \otimes (((n\backslash n)/(\Diamond\Box np\backslash s)) \otimes (np \otimes (np\backslash(np\backslash s)))) \longrightarrow n}\triangleleft^{-1} \tag{3.10}$$

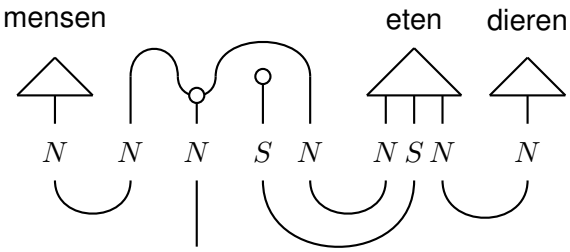
but then diverge in how the relative clause body is derived:

$$\dfrac{\overline{np \longrightarrow np} \quad \dfrac{\dfrac{\dfrac{\overline{np \longrightarrow np}}{\Box np \longrightarrow \Box np}\Box}{\Diamond\Box np \longrightarrow np}\nabla^{-1} \quad \overline{s \longrightarrow s}\backslash}{np\backslash s \longrightarrow \Diamond\Box np\backslash s}\boxed{}}{np\backslash(np\backslash s) \longrightarrow np\backslash(\Diamond\Box np\backslash s)}\backslash}{np \otimes (np\backslash(np\backslash s)) \longrightarrow \Diamond\Box np\backslash s}\triangleleft^{-1}$$

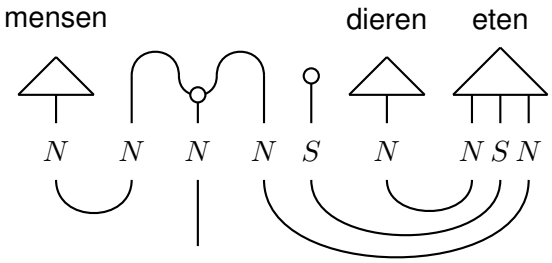
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{np \longrightarrow np}}{\Box np \longrightarrow \Box np}\Box}{\Diamond\Box np \longrightarrow np}\nabla^{-1} \quad \dfrac{\overline{np \longrightarrow np} \quad \overline{s \longrightarrow s}}{np\backslash s \longrightarrow np\backslash s}\backslash}{np\backslash(np\backslash s) \longrightarrow \Diamond\Box np\backslash(np\backslash s)}\boxed{}}{\Diamond\Box np \otimes (np\backslash(np\backslash s)) \longrightarrow np\backslash s}\triangleleft^{-1}}{np \otimes (\Diamond\Box np \otimes (np\backslash(np\backslash s))) \longrightarrow s}\triangleleft^{-1}}{\Diamond\Box np \otimes (np \otimes (np\backslash(np\backslash s))) \longrightarrow s}\widehat{\sigma}^{l}_{\Diamond}}{np \otimes (np\backslash(np\backslash s)) \longrightarrow \Diamond\Box np\backslash s}\triangleleft \tag{3.11}$$

In the derivation on the left, the $\Diamond\Box np$ hypothesis is linked to the *subject* argument of the verb; in the derivation on the right to the *object* argument, reached via the $\widehat{\sigma}^{l}_{\Diamond}$ reordering step. We show the information flow diagrams and mapping to a compact closed setting for these derivations in Figure 3.12.

The derivational ambiguity of (3.11) gives rise to two ways of obtaining a vector $\mathbf{v} \in \mathsf{N}$, as displayed in Figure 3.12. They differ in whether the index of the $\Diamond\Box np$ hypothesis in the relative pronoun type, contracts with index for the subject argument of the verb (3.9*a*) or with the direct object index *o* (3.9*b*). Filling in the lexical specification of [SCC13], it leads to exactly the two desired readings. The subject relative reading is expressed by



whereas the object relative reading is expressed by the following diagram:



The algebraic versions of the images above are given in Equations 3.12 and 3.13.

$$(3.9a) \quad = \quad \mathbf{mensen} \odot \Big[\Big(\sum_S \mathbf{eten} \Big)^{\mathsf{T}} \mathbf{dieren} \Big] \tag{3.12}$$

$$(3.9b) \quad = \quad \mathbf{mensen} \odot \Big[\Big(\sum_S \mathbf{eten} \Big) \mathbf{dieren} \Big] \tag{3.13}$$

3.3.2 Discussion

We briefly discuss the analogy between the formal semantics account of the previous section and the vector semantics modelling we presented before. As suggested in the previous section, the formal semantics approach to pronoun relativisation encapsulates the non-linear behaviour of the relative pronoun by means of a non-linear lexical lambda term, which uses the same variable twice. This allows for the syntactic engine to stay linear, thereby keeping its complexity low.

In the vector semantics version of pronoun relativisation, exactly the same approach is taken, which we illustrate in Figure 3.11. Although the proof system requires movement of information (as seen in the bottom information flow diagram of Figure 3.12), there is no duplication involved in the derivational aspect of the diagram. This is illustrated in the compact closed category diagrams of Figure 3.12, which only contain wires that link together information, and no expansions or merges of information via Frobenius operations. The non-linearity comes in only through the lexicon, where Frobenius operations μ and ι are used to respectively model the merging effect of the relative pronoun, and the pushing down of the verb phrase into a noun space.

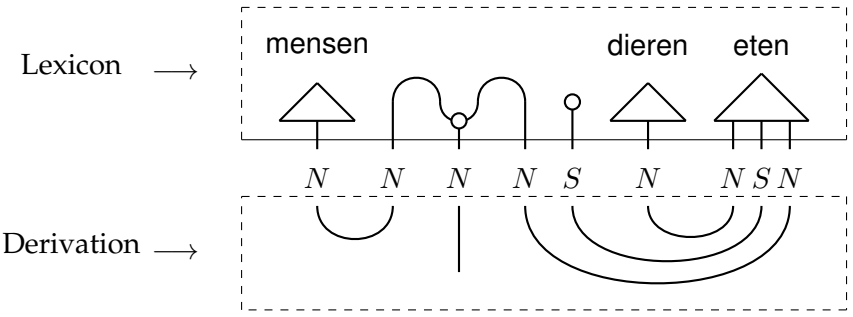


FIGURE 3.11: The division of labour between lexicon and derivation in pronoun relativisation illustrated. The top part indicates the lexical specification of the semantic elements to be combined, with a non-linearity given by the Frobenius operations in the relative pronoun. The bottom part shows the derivational semantics, which is entirely linear.

In this respect, the formal semantics account makes the same design choice regarding the division of labour between derivational and lexical semantics as the distributional account, where the extra expressivity of the Frobenius operations is called upon for specifying the lexical meaning recipe for the relative pronoun.

FIGURE 3.12: Information flow and CCC diagrams for subject (above) and object (below) relative readings in Dutch.

In the concrete vector modelling, we showed the the Frobenius μ map instantiates element wise multiplication. Whereas in the formal semantics account, set intersection is made possible by reusing a variable z to conjoin two predicates, in the case of the concrete vector modelling a similar merging effect is achieved by using element wise multiplication of the head noun with the conflated verb phrase in the relative clause body.

In the following section, we depart from this strategy of a linear derivational semantics and a non-linear lexical semantics. We consider verb ellipsis with anaphora and introduce a proof system that allows for a duplication of information, which then allows us to duplicate information in the derivational semantics as a result of translation (the now non-linear) proofs to linear maps with Frobenius Algebras.

3.4 Parasitic Gaps

When covering larger fragments of language, one may wonder how far we can extend the approach outlined above to cover more complex phenomena, where linear derivations are combined with complex lexical specifications that allow one to use the Frobenius operations to multiply and merge, insert and delete information. Although we will argue later on that the general phenomena of ellipsis may not always be treated using this methodology but rather should rely on a copying mechanism in the grammar (as is also argued for by Morrill and Valentín [MV15] and Morrill and Valentín [MV16]), the particular case of *parasitic gapping* can be largely treated by clever syntactic typing and lexical meaning assignment. We show how common forms of parasitic gaps can be given a vector semantics.

As the name suggests, a *parasitic* gap is felicitous only in the presence of a primary gap. We expand on the examples introduced in Section 3.1 and give the examples in (3.14) to illustrate the relevant gap patterns with relative clause constructions.

The case of object relativisation in (a) has a single gap (indicated by ␣) for the unexpressed direct object of *rejected*. The relative clause in (d) has two gaps: the primary one is for the object of *rejected* as in (a); the secondary, parasitic gap (marked by the rightmost ␣) is the unexpressed object of *reading*. The parasitic gap occurs here in a phrase that by itself would be an island for extraction, compare (d) with the ungrammatical (c).

$$
\begin{array}{ll}
a & \text{papers that Bob rejected ␣ (immediately)} \\
b & \text{Bob left without closing the window} \\
c & {}^{*}\text{window that Bob left without closing ␣} \\
d & \text{papers that reviewers rejected ␣ without reading ␣ (carefully)}
\end{array}
\tag{3.14}
$$

For more examples and linguistic analysis, we refer the reader to [CP01]. Here, we build on the previous section: using the grammar logic $\mathbf{NL}_{\Diamond}$ with the rightward extraction rules for English, we propose a type assignment to treat the examples above relying on type polymorphism for the coordinating words 'and' and 'without'.

3.4.1 Deriving Parasitic Gaps

For the case of non-subject relativisation (3.14a), the procedure is exactly the same as in Section 3.3: the relative pronoun 'that' is typed as a functor that produces a noun modifier $n\backslash n$ in combination with a sentence that contains an unexpressed np hypothesis ("Bob rejected ␣ immediately"). The gap subtype is modally decorated as $\Diamond\Box np$. The $\Diamond$ marking allows it to cross phrase boundaries on its way to the phrase-internal position adjacent to the transitive verb 'rejected'. At that point, the licensing $\Diamond$ has done its work, and can be cleaned up thanks to $\Diamond\Box np \to np$, which provides the np object required by 'rejected'.

The gap-less example (3.14b) provides the motivation for the basic type assignment to *without* as a functor combining with a non-finite gerund clause gp to produce a verb-phrase modifier $iv\backslash iv$. In order to block the ungrammatical (3.14c), we follow [Mor12] and lock the $iv\backslash iv$ result type with $\Box$; the matching $\Diamond$ needed to unlock it effectively demarcates the modifier phrase *without closing the window* as an island, making it inaccessible for the $\alpha^r_\diamond, \sigma^r_\diamond$ extraction postulates.

To account for the double use of the gap in (3.14d) we replace *syntactic* copying via controlled contraction by *lexical* polymorphism: *without* is treated as as a polymorphic item on a par with coordinators *and, but*. These chameleon words are associated with a type *schema* that allows them to adapt to their syntactic context. For the conjunction *and*, the schematic form is $(X\backslash X)/X$; from a basic instantiation $X = s$ for sentence conjunction, one then obtains derived instantiations $X = np\backslash s$ (verb phrase conjunction), $X = (np\backslash s)/np$ (transitive verb conjunction), etc, with a predictable interpretation derived from the basic conjunction in type s. Similarly, for *without*, we have the polymorphic schema

$$\text{without} :: \Box(X\backslash X)/Z$$

with basic instantiation $X = iv$, $Z = gp$. Uniformly dividing the subtypes iv and gp by $\Diamond\Box np$ produces the derived instantiations $X = iv/\Diamond\Box np$ and $Z = gp/\Diamond\Box np$ for the parasitic gapping example (3.14d). We summarise the type assignments in the lexicon below.

Word	Lambek type
papers	n
that	$(n\backslash n)/(s/\Diamond\Box np)$
Bob	np
rejected	$(np\backslash s)/np$
reading	gp/np
immediately, carefully	$iv\backslash iv$
without^{b,c}	$\Box(iv\backslash iv)/gp$
without^{d}	$\Box((iv/\Diamond\Box np)\backslash(iv/\Diamond\Box np))/(gp/\Diamond\Box np)$

With the syntactic typing in hand, we can derive the examples in 3.14. The derivation

for the non-subject relativisation is already given in Figure 3.8, the derivation for the double parasitic gap in 3.14d is given in Figure 3.13. In the same way we did in Section 3.3, we can depict the proof of Figure 3.13 using the diagrammatic language for the Lambek Calculus with modalities. This representation is given in Figure 3.14.

3.4.2 Frobenius Semantics for Parasitic Gaps

To concretise the vectorial semantics of parasitic gaps, we define an interpretation $\lceil\cdot\rceil$ that sends syntactic types and derivations to the corresponding components of the Compact Closed Category of **FVect**, along the lines of Section 3.3. However, for the analysis above we use three basic types, which are mapped as below:

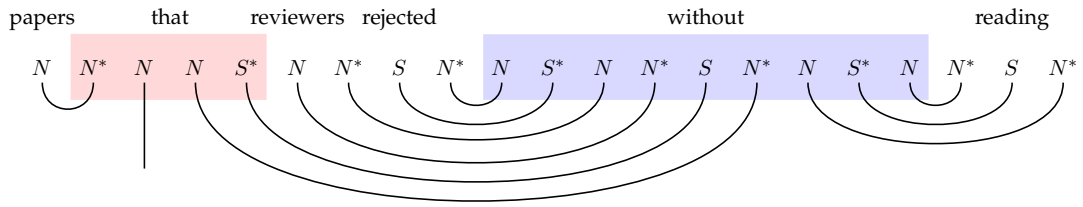
$$\begin{aligned} \lceil np\rceil &= \lceil n\rceil = N, \\ \lceil s\rceil &= S, \\ \lceil gp\rceil &= N^* \otimes S, \end{aligned}$$

Notice that gp is mapped to $N^* \otimes S$, making the understood subject of the gerund semantically recoverable. With $X = iv/\Diamond\Box np$ and $Z = gp/\Diamond\Box np$, the syntactic derivation of (d)

papers	that	reviewers	rejected	without	reading	
n	$(n\backslash n)/(s/\Diamond\Box np)$	np	$(np\backslash s)/np$	$(\Box(X\backslash X))/Z$	gp/np	$\vdash n$

is graphically depicted in Figure 3.14. The typing for *without* ensures that we insert the appropriate missing hypotheses in direct object positions of *rejected* and *reading*, which is attested by the $\Diamond, \Box$ links in the diagram. After creating the adjunct phrase *without reading*, the additional $\Box$ decoration forces the adjunct to be demarcated with a $\Diamond$, which would block derivations of the ungrammatical example in (3.14c). Then, composition with the subject *reviewers* is done with the resulting type providing the np hypothesis that is given in the primary gap type for *that*.

Semantically, the interpretation will map to the following contractions in the interpreting CCC (red: $\lceil$that$\rceil$, blue: $\lceil$without$\rceil$):



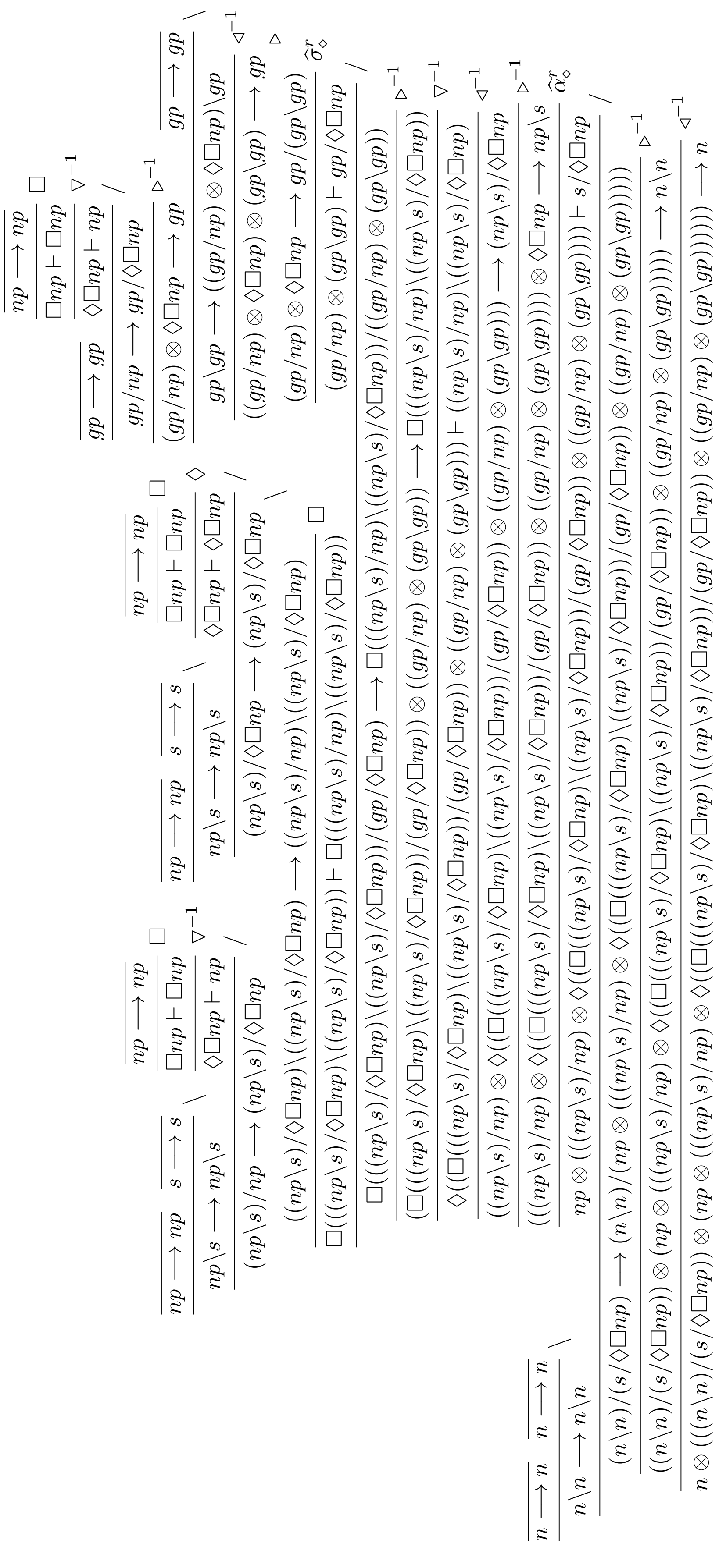


FIGURE 3.13: "papers that reviewers rejected ␣ without reading ␣ carefully"

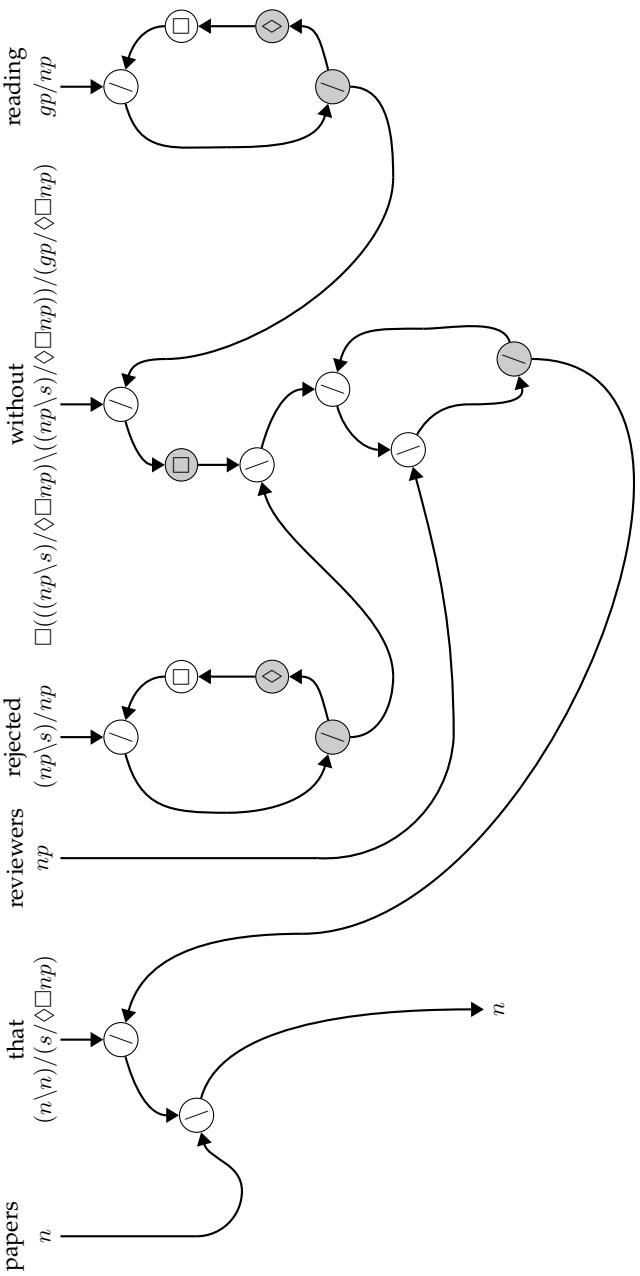


FIGURE 3.14: Information flow for the double parasitic gap.

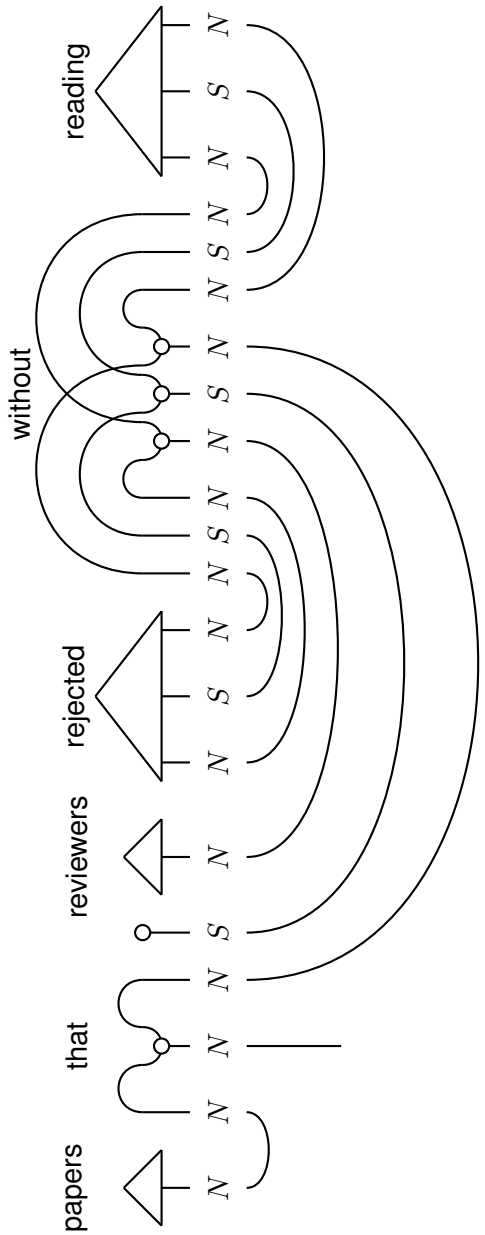
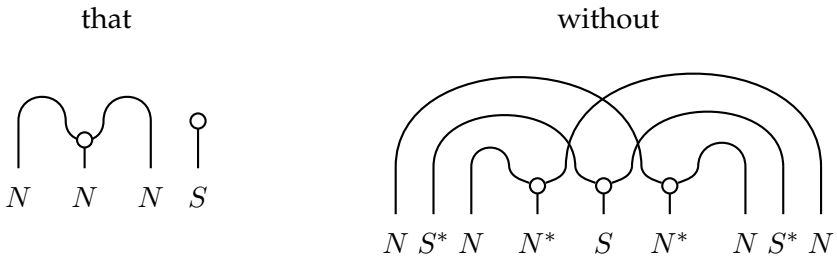
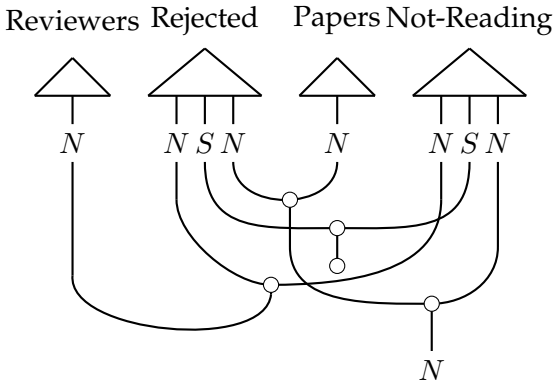


FIGURE 3.15: Semantic information flow for the double parasitic gap, not in normal form yet.

For the lexical meaning we take the following approach: the diagram for *that* is as developed in [SCC13]. The diagram for *without* now coordinates the understood the three subjects and object, as well as the sentence types. Both diagrams are depicted below.



Combining the derivational semantics of the string diagram above with the lexical specifications of *that* and *without*, we obtain the complete CCC diagram in Figure 3.15, which, after simplifying the wiring, gives the normal form below:



The above diagrams are morphisms of a symmetric compact closed category with Frobenius algebras and can be written down in that language e.g. as done in [SCC13; MW17]. Here, we provide the closed linear algebraic form of the above normal form. For $\overline{\overline{\text{Rejected}}}$ and $\overline{\overline{\text{Not-Reading}}}$ the rank 3 tensors interpreting *rejected* and *(without) reading*, and ι the unit of the Frobenius coalgebra, this is

$$\overrightarrow{\text{Papers}} \odot (\iota_S \otimes id_N)\big(\overrightarrow{\text{Reviewers}}^T \times (\overline{\overline{\text{Rejected}}} \odot \overline{\overline{\text{Not-Reading}}})\big)$$

This says that we take the element wise multiplication of both cubes, and contract them with the subject *Reviewers*; then, we collapse the resulting matrix into a vector and intersect this with the head noun *Papers*.

3.4.3 Discussion

The concrete modelling presented above will produce a sentence representation that is analogous to the usual formal semantics account: where we use element wise multiplication, in formal semantics the coordinator is modelled using set intersection. Thinking of both of these operations as a merging operation, the example above is analogous to selecting those papers that were both rejected and not reviewed, by reviewers.

The main difference with formal semantics, however, is that both common nouns and proper noun phrases are mapped onto a single vector space (a set-theoretic account represents common nouns as functions from entities to truth values), and as a result the way to extract a noun is to merge the information encoded in the sentence dimension by means of the ι. Although this map provides a concrete example of how to handle relative pronouns, one may argue that conflating information is not the desired behaviour for the relative pronoun. As a first step towards a general model, we give the general normal form, given some modelling of the relative pronoun, in Figure 3.16.

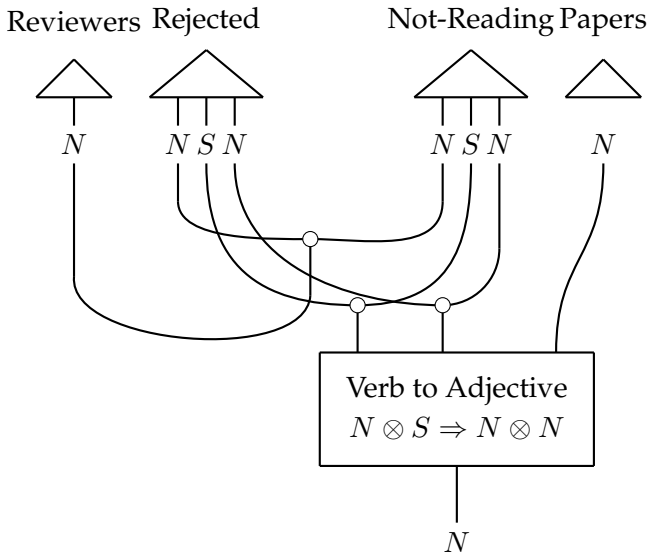


FIGURE 3.16: General normal form for a sentence with a parasitic gap; the relative pronoun is now a general map that transforms a verb phrase ($N \otimes S$) into an adjective ($N \otimes N$).

Given the syntactic type of the relative pronoun ($(n\backslash n)/(s/\Diamond\Box np)$), under our type translation the relative pronoun effectively is some map from a verb phrase ($N \otimes S$) to an adjective modifying a (common) noun ($N \otimes N$). With this generalisation, we are not bound anymore to a specific implementation of the relative pronoun, although the proposed account for now gives a workable account for experimentation.

We suggest here, that a data-driven approach may lend itself for modelling the relative pronoun, as it essentially binds a verb phrase to its adjectival form. For example, a verb phrase can occur in adjectival form, e.g. “papers that Bob rejected" vs “rejected (by Bob) papers”. In such cases, we would expect to get the same meaning representation, which crucially relies on being able to project either an adjective onto a verb phrase or vice versa.

3.5 Verb Phrase Ellipsis and Anaphora

We turn now to the third type of ellipsis we wish to model in a compositional distributional setting: verb phrase ellipsis with anaphora. As discussed in Section 3.1, verb phrase ellipsis is distinct from (parasitic) gapping in the sense that there is always an auxiliary verb present to mark the location of the elided verb phrase, but not always a relative pronoun or a coordinator. This is even more so in the presence of anaphora, where a resolution needs to take place that is fully induced by the anaphora itself, independent of any coordinating element earlier in the sentence. Moreover, in the presence of anaphora, verb phrase elliptical phrases may induce an ambiguity between strict and sloppy readings, besides from the fact that the anaphor also needs to be resolved.

We remind the reader of the relevant examples of verb phrase ellipsis in Equations 3.15 and 3.16. In example (a) the elided verb phrase is marked by the auxiliary verb, and it is ideally in bidirectional entailment with 3.15(b), i.e. (a) entails (b) and (b) entails (a).

$$\begin{array}{ll} a & \text{Alice drinks and Bob does too} \\ b & \text{Alice drinks and Bob drinks} \end{array} \tag{3.15}$$

In the presence of anaphora, a verb phrase elliptical phrase induces an ambiguity, where the ambiguous phrase 3.16(a) has two readings 3.16(b) and 3.16(c).

$$\begin{array}{lll} a & \text{“Gary loves his code and Bill does too”} & \text{(ambiguous)} \\ b & \text{“Gary loves Gary's code and Bill loves Gary's code”} & \text{(strict)} \\ c & \text{“Gary loves Gary's code and Bill loves Bill's code”} & \text{(sloppy)} \end{array} \tag{3.16}$$

In a formal semantics account, the first example could be analysed with the auxiliary verb as an identity function on the main verb of the sentence and an intersective meaning for the coordinator. Somehow the parts need to be appropriately combined to produce the reading (b) for sentence (a):

$$\begin{array}{c} \text{does too} : \lambda x.x \\ \text{and} : \lambda x.\lambda y.(x \wedge y) \end{array} \quad \text{should give} \quad \mathbf{drinks}(\mathbf{alice}) \wedge \mathbf{drinks}(\mathbf{bill})$$

The second example would assume the same meaning for the coordinator and auxiliary but now the possessive pronoun “his” gets a more complicated term: $\lambda x.\lambda y.\mathbf{owns}(x, y)$. The analysis then somehow should derive two readings:

$$\begin{array}{ll} \mathbf{loves}(\mathbf{gary}, x) \wedge \mathbf{owns}(\mathbf{gary}, x) \wedge \mathbf{loves}(\mathbf{bill}, x) & \text{(strict)} \\ \mathbf{loves}(\mathbf{gary}, x) \wedge \mathbf{owns}(\mathbf{gary}, x) \wedge \mathbf{loves}(\mathbf{bill}, y) \wedge \mathbf{owns}(\mathbf{bill}, y) & \text{(sloppy)} \end{array}$$

In the case of pronoun relativisation the relative pronoun carries a complex logical type that facilitates controlled reasoning about how to treat the apparently misplaced head noun. For parasitic gapping, there is moreover a coordinator present that can encode controlled access to copying of the semantic content of overt syntactic elements. Contrary to the cases of pronouns relativisation and parasitic gapping,

in the case of verb phrase ellipsis there is the ellipsis marker that can carry a modally decorated type. This is because the antecedent to be recovered is not a distinct word without meaning; it is a contentful verb phrase. So rather than annotate a relative pronoun or a coordinator with a modal type that allows for long-distance dependencies to be respected, we want to assign a type to the ellipsis marker in such a way that this allows the duplication of the antecedent verb phrase and the movement of this copy into the correct position in the sentence. In this way, we avoid the issue of overloading the lexicon that may arise when opting for lexical polymorphism. The copying approach is schematically depicted in Figure 3.17.

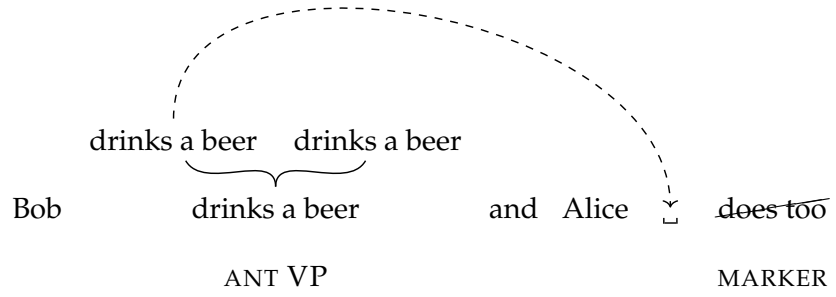


FIGURE 3.17: General strategy for 'copying and movement' for verb phrase ellipsis, shown for "Bob drinks a beer and Alice does too". The antecedent has to be copied, and the copy has to be moved (rightward) into correct position in the sentence.

Verb phrase ellipsis, and especially verb phrase ellipsis with anaphora, does not lend itself to the type of lexical modelling used before; contrary to the approach advocated above for parasitic gapping and pronoun relativisation, the presence of ellipsis is typically indicated by an ellipsis marker, which signals a request for access to the semantic content occurring elsewhere in the phrase, but not necessarily to be obtained via coordination. For anaphora we argue that the same kind of request is posed, where the anaphora should elicit access to an element occurring elsewhere (typically to its left), and hence this should be modelled in the specification of its type, but it's access requirements modelled in the derivation itself. For this reason, we to opt for the use of a controlled contraction rule to facilitate syntactic 'copying' for the modelling of verb phrase ellipsis with anaphora distributionally. This 'copying' should not be understood as an overt process; rather, the contraction rule allows duplicate uses of the same information to be merged together as will become apparent in our analysis below.

Another motivation for the use of a controlled contraction comes from the fact that previous work shows that the kind of non-linear behaviour we wish to model can be done distributionally by means of the Frobenius expansion map, with signature $A \to A \otimes A$. Thus, the plan is to switch from the non-associative Lambek Calculus with extraction postulates $\mathbf{NL}_\Diamond$ to a similar extension to the associative

Lambek Calculus. We define custom control postulates that — similar to the multimodal system used above — allow for $\Diamond$ decorated formulas to be extracted into a position adjoining their undecorated counterpart. But in addition we allow the $\Diamond$ decorated formula to be contracted with their undecorated neighbour type. By means of lexical specification in the ellipsis marker, we then license the introduction of such $\Diamond$ formulas.

The approach we introduce is not entirely unprecented; Jäger [Jäg98] introduces a similar logic for verb phrase ellipsis, and we will show that our approach is a simpler, unary implementation of essentially the same system. However, our approach is much more suited for implementation in a compositional distributional model.

3.5.1 A Proof System for Controlled Copying

Similar to the previous sections, our starting point is the (associative) Lambek Calculus, enhanced with control modalities $\Diamond, \Box$ which form a residuated pair. Again, the modalities have a purely syntactical role: instead of directly allowing the copying of resources, the system is designed such that a type that is labelled with a $\Diamond$ can be contracted. This prevents the overgeneration of a general contraction rule, but license the multiple use of the (meaning of) words that respect the type that is decorated with $\Diamond$. The residuated $\Box$ modality allows for the system to operate on the subtype of a $\Diamond$ decorated type without losing track of the latter's position. The full proof system is given in Figure 3.18.

$$\frac{}{1_A : A \longrightarrow A} \qquad \frac{f : A \longrightarrow B \quad g : B \longrightarrow C}{g \circ f : A \longrightarrow C}$$

$$\frac{f : \Diamond A \longrightarrow B}{\nabla f : A \longrightarrow \Box B} \qquad \frac{f : A \otimes B \longrightarrow C}{\triangleright f : A \longrightarrow C/B} \qquad \frac{f : A \otimes B \longrightarrow C}{\triangleleft f : B \longrightarrow A\backslash C}$$

$$\frac{g : A \longrightarrow \Box B}{\nabla^{-1} g : \Diamond A \longrightarrow B} \qquad \frac{g : A \longrightarrow C/B}{\triangleright^{-1} g : A \otimes B \longrightarrow C} \qquad \frac{g : B \longrightarrow A\backslash C}{\triangleleft^{-1} g : A \otimes B \longrightarrow C}$$

$$\frac{f : (A \otimes B) \otimes C \longrightarrow D}{\widehat{\alpha^{-1}}(f) : A \otimes (B \otimes C) \longrightarrow D} \qquad \frac{f : A \otimes (B \otimes C) \longrightarrow D}{\widehat{\alpha}(f) : (A \otimes B) \otimes C \longrightarrow D}$$

$$\frac{f : \Diamond A \otimes A \longrightarrow B}{\widehat{C}(f) : A \longrightarrow B} \qquad \frac{f : A \otimes (\Diamond B \otimes C) \longrightarrow D}{\widehat{M}(f) : (\Diamond B \otimes A) \otimes C \longrightarrow D} \qquad \frac{f : \Diamond A \otimes (\Diamond B \otimes C) \longrightarrow D}{\widehat{S}(f) : \Diamond B \otimes (\Diamond A \otimes C) \longrightarrow D}$$

FIGURE 3.18: $\mathbf{L}_F$. Residuation rules and structural postulates for controlled copying and moving (rule form). The names of the rules are given by the term symbols in the conclusion of each rule.

Combining $\Diamond$ with a $\backslash$ or $/$ creates the behaviour wanted for ellipsis: an ellipsis marker will generally be annotated with a type $\Diamond A \backslash B$, where the expected $\Diamond A$ typically does not occur in the given sequence of types to be proven. Rather, this 'missing copy' is inserted and by means of the structural rules extracted to the left until it appears beside a an undecorated witness of type A. If we read such a derivation the other way around, from bottom to top, it has the effect that the candidate for ellipsis gets copied and its content is reused as an argument for the ellipsis marker. This is expressed in Figure 3.19.

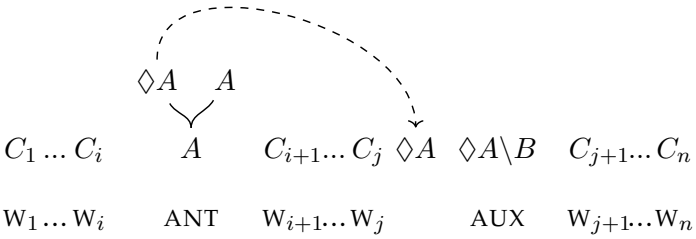


FIGURE 3.19: General strategy for ellipsis resolution in $\mathbf{L}_{\Diamond,\mathbf{F}}$. The antecedent is copied, and the $\Diamond$ decorated copy is moved directly left of the marker which consumes the copy.

To this end, the modalities license access to (limited forms of) contraction and commutativity, through the use of structural rules (see Figure 3.18). The structural rules can also be stipulated in equivalent axiomatic form, but for the purpose of parsing it is more useful to consider rule form. We want to have a system that enjoys decidability: though this is not a straightforward property in the presence of just contraction[1], our system becomes decidable easily if we put a bound on the number of contractions in a proof.

3.5.2 Relation to related approaches

The system $\mathbf{L}_F$ as we defined it above is closely related to the multimodal system **LA** introduced by Jäger [Jäg98] to deal with ellipsis. In this subsection we show how our system in fact can be derived from Jäger's system by means of a translation. The system **LA** introduces a second triple of connectives $\sim, \hookrightarrow, \hookleftarrow$, with the same residuation behaviour as $\otimes, \backslash, /$. In addition, three structural rules are defined that govern the structural behaviour of copying and movement. The non-Lambek part of the system is given in Figure 3.20, and we give a translation from **LA** to $\mathbf{L}_F$ and show that this translation preserves all theorem of the Jäger's original system, but using only a pair of unary control modalities. The fact that we translate from a residuated triple of binary connectives to a adjoint pair of unary connectives also means that a full equivalence theorem is harder to achieve. We refer the reader to to

[1] See the discussion in Katalin Bimbó's monograph [Bim14] and the results of [SO96; CH16].

Section 5.3 of the dissertation of Versmissen [Ver96] for an interesting discussion on this.

$$\frac{f : A \sim B \longrightarrow C}{\blacktriangleright f : A \longrightarrow C \hookleftarrow B} \qquad \frac{f : A \sim B \longrightarrow C}{\blacktriangleleft f : B \longrightarrow A \hookrightarrow C}$$

$$\frac{g : A \longrightarrow C \hookleftarrow B}{\blacktriangleright^{-1} g : A \sim B \longrightarrow C} \qquad \frac{g : B \longrightarrow A \hookrightarrow C}{\blacktriangleleft^{-1} g : A \sim B \longrightarrow C}$$

$$\frac{f : A \sim A \longrightarrow B}{\widehat{C}(f) : A \longrightarrow B} \qquad \frac{f : A \otimes (B \sim C) \longrightarrow D}{\widehat{IM}(f) : (B \sim A) \otimes C \longrightarrow D} \qquad \frac{f : A \sim (B \sim C) \longrightarrow D}{\widehat{P}(f) : B \sim (A \sim C) \longrightarrow D}$$

FIGURE 3.20: Extending **L** to **LA**: logical rules (top) and structural rules (bottom, rule form).

First, we translate the formulas of **LA** to formulas of $\mathbf{L}_F$: basic types are unchanged, as well as the Lambek connectives $\otimes, \backslash, /$:

$$p^\circ = p \qquad (A \otimes B)^\circ = A^\circ \otimes B^\circ \qquad (A/B)^\circ = A^\circ/B^\circ \qquad (A\backslash B)^\circ = A^\circ\backslash B^\circ$$

The extra residuated triple, however, is now interpreted using the unary modalities:

$$(A \sim B)^\circ = \Diamond A \otimes B \qquad (A \hookrightarrow B)^\circ = \Diamond A^\circ\backslash B^\circ \qquad (A \hookleftarrow B)^\circ = \Box(A^\circ/B^\circ)$$

With the given translation, we can show that the system $\mathbf{L}_F$ is a translation of **LA**, i.e. all deductions of the latter are deductions of $\mathbf{L}_F$:

Theorem 1 *LA* $\vdash A \to B$ *only if* $\mathbf{L}_F \vdash A^\circ \to B^\circ$.

Proof 1 *By induction on the length of derivations in* **LA**. *As the translation directly converts the structural rules of* **LA** *to those of* $\mathbf{L}_F$, *and the Lambek connectives remain identical under translation, we only have to consider the logical rules for* $\sim, \hookrightarrow, \hookleftarrow$.

1. *For* $\blacktriangleright$ *and* $\blacktriangleright^{-1}$ *we show*

$$\dfrac{\dfrac{\dfrac{\vdots}{\Diamond A \otimes B \longrightarrow C}\,IH}{\Diamond A \longrightarrow C/B}\,\triangleright}{A \longrightarrow \Box(C/B)}\,\triangledown \qquad \dfrac{\dfrac{\dfrac{\vdots}{A \longrightarrow \Box(C/B)}\,IH}{\Diamond A \longrightarrow C/B}\,\triangledown^{-1}}{\Diamond A \otimes B \longrightarrow C}\,\triangleright^{-1}$$

2. *For* $\blacktriangleleft$ *and* $\blacktriangleleft$ *we have*

$$\dfrac{\dfrac{\vdots}{\Diamond A \otimes B \longrightarrow C}\,IH}{B \longrightarrow \Diamond A\backslash C}\,\triangleleft \qquad \dfrac{\dfrac{\vdots}{B \longrightarrow \Diamond A\backslash C}\,IH}{\Diamond A \otimes B \longrightarrow C}\,\triangleleft^{-1}$$

3.5.3 Deriving Ellipsis with Anaphora

We have argued above and opted for a mechanism by which the verb phrase, in an elliptical phrase like "Alice drinks and Bob does too", is used twice; the proof system handles this by means of the structural rules of controlled contraction and movement. Figure 3.21 shows the derivation for the phrase — skipping trivial applications of associativity — with the copy of the verb phrase highlighted in red. Its graphical representation is shown in Figure 3.22. The derivation follows the general pattern depicted in Figure 3.19, with the auxiliary verb "does too" marking the ellipsis site and therefore typed $\Diamond(np\backslash s)\backslash(np\backslash s)$; requiring a copied verb phrase from *somewhere* to its left, it will return a verb phrase. Reading from bottom to top, first a contraction is applied to copy the verb phrase, marking the copy with a control modality. Then, the copy of the verb phrase is structurally moved rightward until it is in the right position to interact with the auxiliary ellipsis marker. It is not hard to see, then, that this allows the meaning of the verb phrase to be multiplied, have the copied version interact with the auxiliary to give the meaning of the whole phrase in a similar way as the meaning of the expanded phrase "Alice drinks and Bob drinks" would be computed.

$$\frac{\dfrac{\dfrac{np \longrightarrow np \quad s \longrightarrow s}{np\backslash s \longrightarrow np\backslash s}\backslash}{np \otimes np\backslash s \longrightarrow s}\triangleleft^{-1} \quad s \longrightarrow s}{s\backslash s \longrightarrow (np \otimes np\backslash s)\backslash s}\backslash$$

$$\frac{\dfrac{\dfrac{\dfrac{np \longrightarrow np \quad s \longrightarrow s}{np\backslash s \longrightarrow np\backslash s}\backslash}{\Diamond(np\backslash s) \longrightarrow \Diamond(np\backslash s)}\Diamond \quad \dfrac{np \longrightarrow np \quad s \longrightarrow s}{np\backslash s \longrightarrow np\backslash s}\backslash}{\Diamond(np\backslash s)\backslash(np\backslash s) \longrightarrow \Diamond(np\backslash s)\backslash(np\backslash s)}\backslash}{\dfrac{\Diamond(np\backslash s) \otimes \Diamond(np\backslash s)\backslash(np\backslash s) \longrightarrow np\backslash s}{np \otimes (\Diamond(np\backslash s) \otimes \Diamond(np\backslash s)\backslash(np\backslash s)) \longrightarrow s}\triangleleft^{-1}}\triangleleft^{-1}$$

$$\frac{s\backslash s \longrightarrow (np \otimes np\backslash s)\backslash s \quad np \otimes (\Diamond(np\backslash s) \otimes \Diamond(np\backslash s)\backslash(np\backslash s)) \longrightarrow s}{(s\backslash s)/s \longrightarrow ((np \otimes np\backslash s)\backslash s)/(np \otimes (\Diamond(np\backslash s) \otimes \Diamond(np\backslash s)\backslash(np\backslash s)))}/$$

$$\frac{(s\backslash s)/s \longrightarrow ((np \otimes np\backslash s)\backslash s)/(np \otimes (\Diamond(np\backslash s) \otimes \Diamond(np\backslash s)\backslash(np\backslash s)))}{(s\backslash s)/s \otimes (np \otimes (\Diamond(np\backslash s) \otimes \Diamond(np\backslash s)\backslash(np\backslash s))) \longrightarrow (np \otimes np\backslash s)\backslash s}\triangleright^{-1}$$

$$\frac{(s\backslash s)/s \otimes (np \otimes (\Diamond(np\backslash s) \otimes \Diamond(np\backslash s)\backslash(np\backslash s))) \longrightarrow (np \otimes np\backslash s)\backslash s}{(np \otimes np\backslash s) \otimes ((s\backslash s)/s \otimes (np \otimes (\Diamond(np\backslash s) \otimes \Diamond(np\backslash s)\backslash(np\backslash s)))) \longrightarrow s}\triangleleft^{-1}$$

$$\frac{(np \otimes np\backslash s) \otimes ((s\backslash s)/s \otimes (np \otimes (\Diamond(np\backslash s) \otimes \Diamond(np\backslash s)\backslash(np\backslash s)))) \longrightarrow s}{(np \otimes np\backslash s) \otimes ((s\backslash s)/s \otimes (\Diamond(np\backslash s) \otimes (np \otimes \Diamond(np\backslash s)\backslash(np\backslash s)))) \longrightarrow s}M$$

$$\frac{(np \otimes np\backslash s) \otimes ((s\backslash s)/s \otimes (\Diamond(np\backslash s) \otimes (np \otimes \Diamond(np\backslash s)\backslash(np\backslash s)))) \longrightarrow s}{(np \otimes np\backslash s) \otimes (\Diamond(np\backslash s) \otimes ((s\backslash s)/s \otimes (np \otimes \Diamond(np\backslash s)\backslash(np\backslash s)))) \longrightarrow s}M$$

$$\frac{(np \otimes np\backslash s) \otimes (\Diamond(np\backslash s) \otimes ((s\backslash s)/s \otimes (np \otimes \Diamond(np\backslash s)\backslash(np\backslash s)))) \longrightarrow s}{(np \otimes (\Diamond(np\backslash s) \otimes np\backslash s)) \otimes ((s\backslash s)/s \otimes (np \otimes \Diamond(np\backslash s)\backslash(np\backslash s))) \longrightarrow s}M$$

$$\frac{(np \otimes (\Diamond(np\backslash s) \otimes np\backslash s)) \otimes ((s\backslash s)/s \otimes (np \otimes \Diamond(np\backslash s)\backslash(np\backslash s))) \longrightarrow s}{(np \otimes np\backslash s) \otimes ((s\backslash s)/s \otimes (np \otimes \Diamond(np\backslash s)\backslash(np\backslash s))) \longrightarrow s}C$$

FIGURE 3.21: Short hand derivation for "Alice drinks and Bob does-too". The copied verb and its subformulas are highlighted in red.

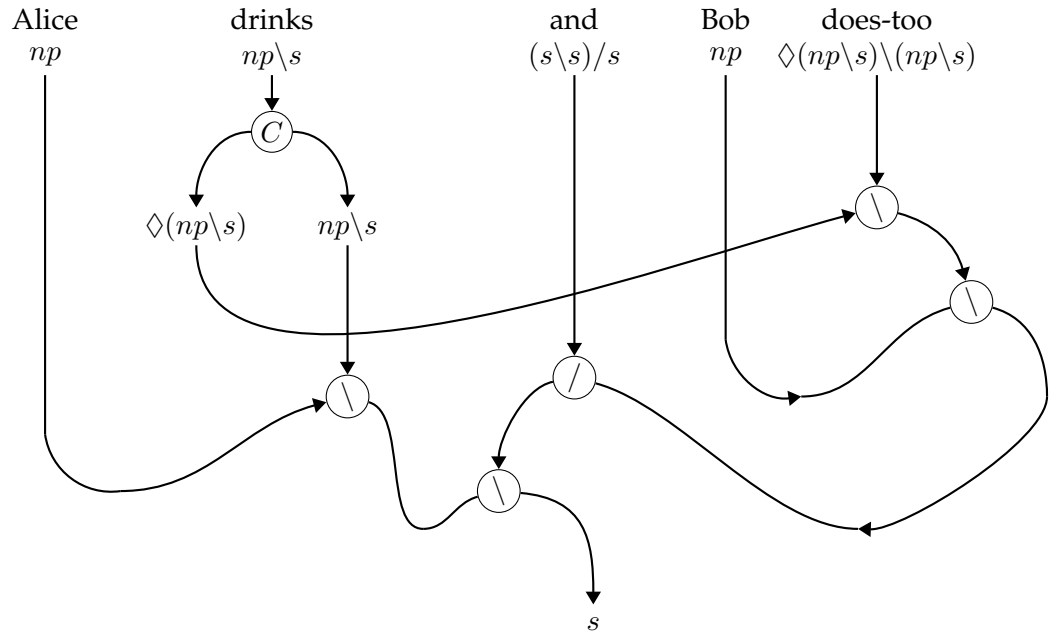


FIGURE 3.22: Syntactic Information flow for the derivation in Figure 3.21.

Structural Ambiguity

More complicated cases of ellipsis contain anaphora, and allow an interaction that leads to strict and sloppy readings. The example from Equation 3.3a, "Gary loves his code and Bill does too", is analysed in given proof system by allowing a choice of resolution: for the strict reading, in which both Gary and Bill love Gary's code, we first resolve the anaphora 'his' with the (only possible) antecedent 'Gary'. For a sloppy reading, in which each loves their own code, the ellipsis is resolved first, forcing the unresolved verb phrase "loves his code" to be copied and moved over to the place preceding the auxiliary verb marker. Then, the two separate anaphors are resolved with their closest antecedent to obtain the final interpretation. Because of the length of the actual derivations, they are split across four figures: Figures 3.23 and 3.24 shows the derivations of the subclause "Gary loves his code", either directly resolving with 'Gary' (3.23) or leaving the anaphor unresolved (3.24). The main derivations then use these subproofs to configure everything together into the strict (3.25) or sloppy (3.26) reading.

For the strict reading we show the information flow diagram in Figure 3.27, in which the two applications of the controlled contraction rule are indicated by a (C) node. Essentially, the noun phrase type for 'Gary' gets copied and resolved with the anaphor 'his', witnessed by the leftmost (C) node in the diagram. After forming a verb phrase of the correct type $np\backslash s$, this type gets copied and the $\Diamond$ decorated copy is moved over to be consumed by the auxiliary 'does too', the repeated application of the movement rule signified by the wire that crosses over the elements left of the auxiliary verb. For the sloppy case, the diagrammatic representation is in Figure

3.28. Here the situation is more complex: the full type for the (underived) verb phrase $(np\backslash s)/np \otimes \Diamond np\backslash(np/n) \otimes n$ is copied and the copy is moved over to the left of the auxiliary. Then, we have two parallel anaphor resolution steps, respectively resolving with 'Gary' on the left side of the diagram, and with 'Bill' on the right side.

The approach here is in line with proposals of Jacobson [Jac99] and Jaë [Jäg06]. Where Jacobson uses a combinator to effectuate a copying mechanism in the syntax, Jaëger's proposal introduces an extra implication connective | that is governed by a rule that compresses the use of modus ponens and contraction in a single rule application. Here, we separate the laws governing contraction and the swapping of formulas (permutation), and moreover leave the elicitation of this behaviour to the lexical specification of the anaphor.

$$\dfrac{\dfrac{\dfrac{\overline{np \longrightarrow np}}{\Diamond np \longrightarrow \Diamond np}\,\Diamond \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{np \longrightarrow np} \quad \overline{s \longrightarrow s}}{np\backslash s \longrightarrow np\backslash s}\,\backslash \quad \overline{np \longrightarrow np}}{(np\backslash s)/np \longrightarrow (np\backslash s)/np}\,/}{(np\backslash s)/np \otimes np \longrightarrow np\backslash s}\,\triangleright^{-1}}{np \longrightarrow ((np\backslash s)/np)\backslash(np\backslash s)}\,\triangleleft \quad \overline{n \longrightarrow n}}{np/n \longrightarrow (((np\backslash s)/np)\backslash(np\backslash s))/n}\,/}{\Diamond np\backslash(np/n) \longrightarrow \Diamond np\backslash((((np\backslash s)/np)\backslash(np\backslash s))/n)}\,\backslash}{\Diamond np \otimes \Diamond np\backslash(np/n) \longrightarrow (((np\backslash s)/np)\backslash(np\backslash s))/n}\,\triangleleft^{-1}}{(\Diamond np \otimes \Diamond np\backslash(np/n)) \otimes n \longrightarrow ((np\backslash s)/np)\backslash(np\backslash s)}\,\triangleright^{-1}}{(np\backslash s)/np \otimes ((\Diamond np \otimes \Diamond np\backslash(np/n)) \otimes n) \longrightarrow np\backslash s}\,\triangleleft^{-1}}{np \otimes ((np\backslash s)/np \otimes ((\Diamond np \otimes \Diamond np\backslash(np/n)) \otimes n)) \longrightarrow s}\,\triangleleft^{-1}$$

FIGURE 3.23: Derivation of "Gary loves his code" where the anaphor is resolved with antecedent 'Gary'.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{np \longrightarrow np}}{\Diamond np \longrightarrow \Diamond np}\,\Diamond \quad \dfrac{\overline{n \longrightarrow n} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{np \longrightarrow np} \quad \overline{s \longrightarrow s}}{np\backslash s \longrightarrow np\backslash s}\,\backslash \quad \overline{np \longrightarrow np}}{(np\backslash s)/np \longrightarrow (np\backslash s)/np}\,/}{(np\backslash s)/np \otimes np \longrightarrow np\backslash s}\,\triangleright^{-1}}{np \longrightarrow ((np\backslash s)/np)\backslash(np\backslash s)}\,\triangleleft}{np/n \longrightarrow (((np\backslash s)/np)\backslash(np\backslash s))/n}\,/}{\Diamond np\backslash(np/n) \longrightarrow \Diamond np\backslash((((np\backslash s)/np)\backslash(np\backslash s))/n)}\,\backslash}{\Diamond np \otimes \Diamond np\backslash(np/n) \longrightarrow (((np\backslash s)/np)\backslash(np\backslash s))/n}\,\triangleleft^{-1}}{(\Diamond np \otimes \Diamond np\backslash(np/n)) \otimes n \longrightarrow ((np\backslash s)/np)\backslash(np\backslash s)}\,\triangleright^{-1}}{(np\backslash s)/np \otimes ((\Diamond np \otimes \Diamond np\backslash(np/n)) \otimes n) \longrightarrow np\backslash s}\,\triangleleft^{-1}}{np \otimes ((np\backslash s)/np \otimes ((\Diamond np \otimes \Diamond np\backslash(np/n)) \otimes n)) \longrightarrow s}\,\triangleleft^{-1}}{np \otimes (\Diamond np \otimes ((np\backslash s)/np \otimes (\Diamond np\backslash(np/n) \otimes n))) \longrightarrow s}\,\overline{M}}{(\Diamond np \otimes np) \otimes ((np\backslash s)/np \otimes (\Diamond np\backslash(np/n) \otimes n)) \longrightarrow s}\,\overline{M}}{np \otimes ((np\backslash s)/np \otimes (\Diamond np\backslash(np/n) \otimes n)) \longrightarrow s}\,\overline{C}$$

FIGURE 3.24: Derivation of "Gary loves his code" where the anaphor is unresolved.

FIGURE 3.25: Symbolic proof for the strict reading of "Gary loves his code and Bill does too".

$$\vdots$$

$$\overset{Bill}{np} \otimes ((\overset{loves}{np\backslash s)/np} \otimes (\overset{his}{\Diamond np\backslash (np/n)} \otimes \overset{code}{n})) \longrightarrow s$$

$$\lhd$$

$$(np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n) \longrightarrow np\backslash s$$

$$\Diamond$$

$$\Diamond(loves(np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n)) \longrightarrow \Diamond(np\backslash s) \qquad \dfrac{\overline{np \longrightarrow np} \quad \overline{s \longrightarrow s}}{np\backslash s \longrightarrow np\backslash s}\ \backslash$$

$$\backslash$$

$$\Diamond(np\backslash s)\backslash(np\backslash s) \longrightarrow \Diamond((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n))\backslash(np\backslash s)$$

$$\lhd^{-1}$$

$$\Diamond((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n)) \otimes \Diamond(np\backslash s)\backslash(np\backslash s) \longrightarrow np\backslash s$$

$$\lhd^{-1}$$

$$np \otimes (\Diamond((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n)) \otimes \Diamond(np\backslash s)\backslash(np\backslash s)) \longrightarrow s$$

$$\vdots$$

$$\overset{Gary}{np} \otimes ((\overset{loves}{np\backslash s)/np} \otimes (\overset{his}{\Diamond np\backslash (np/n)} \otimes \overset{code}{n})) \longrightarrow s \qquad \overline{s \longrightarrow s}$$

$$\backslash$$

$$s\backslash s \longrightarrow (np \otimes ((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n)))\backslash s$$

$$/$$

$$(s\backslash s)/s \longrightarrow ((np \otimes ((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n)))\backslash s)/(np \otimes (\Diamond((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n)) \otimes \Diamond(np\backslash s)\backslash(np\backslash s)))$$

$$\rhd^{-1}$$

$$(s\backslash s)/s \otimes (np \otimes (\Diamond((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n)) \otimes \Diamond(np\backslash s)\backslash(np\backslash s))) \longrightarrow (np \otimes ((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n)))\backslash s$$

$$\lhd^{-1}$$

$$(np \otimes ((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n))) \otimes ((s\backslash s)/s \otimes (np \otimes (\Diamond((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n)) \otimes \Diamond(np\backslash s)\backslash(np\backslash s)))) \longrightarrow s$$

$$\overline{M}$$

$$(np \otimes ((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n))) \otimes ((s\backslash s)/s \otimes (\Diamond((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n)) \otimes (np \otimes \Diamond(np\backslash s)\backslash(np\backslash s)))) \longrightarrow s$$

$$\overline{M}$$

$$(np \otimes ((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n))) \otimes (\Diamond((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n)) \otimes ((s\backslash s)/s \otimes (np \otimes \Diamond(np\backslash s)\backslash(np\backslash s)))) \longrightarrow s$$

$$\overline{M}$$

$$(np \otimes (\Diamond((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n)) \otimes ((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n)))) \otimes ((s\backslash s)/s \otimes (np \otimes \Diamond(np\backslash s)\backslash(np\backslash s))) \longrightarrow s$$

$$\overline{C}$$

$$(np \otimes ((np\backslash s)/np \otimes (\Diamond np\backslash (np/n) \otimes n))) \otimes ((s\backslash s)/s \otimes (np \otimes \Diamond(np\backslash s)\backslash(np\backslash s))) \longrightarrow s$$

FIGURE 3.26: Symbolic proof for the sloppy reading of "Gary loves his code and Bill does too".

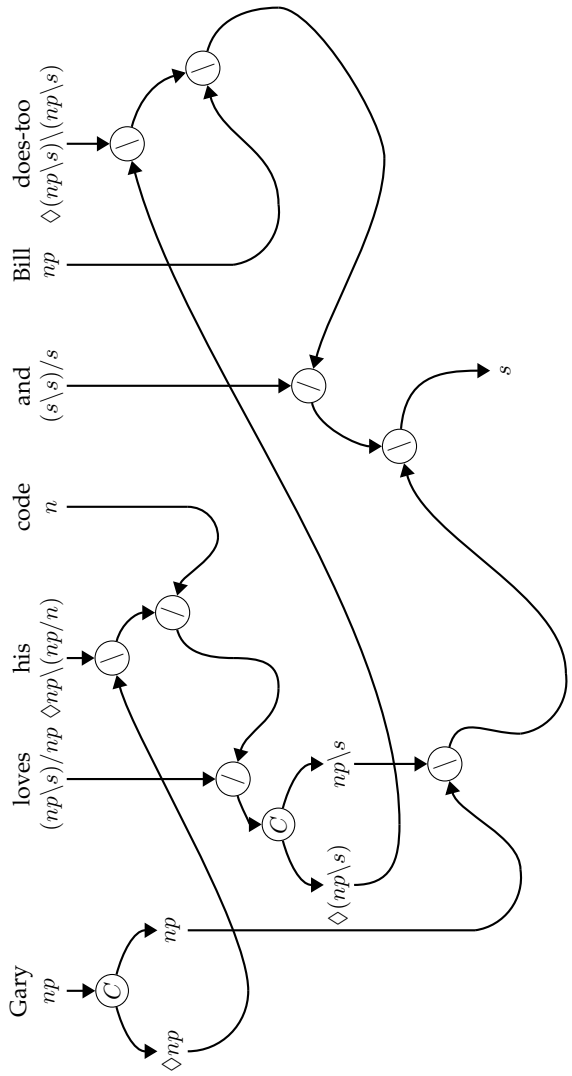


FIGURE 3.27: Syntactic information flow for the strict reading of "Bill loves his code and Gary does too".

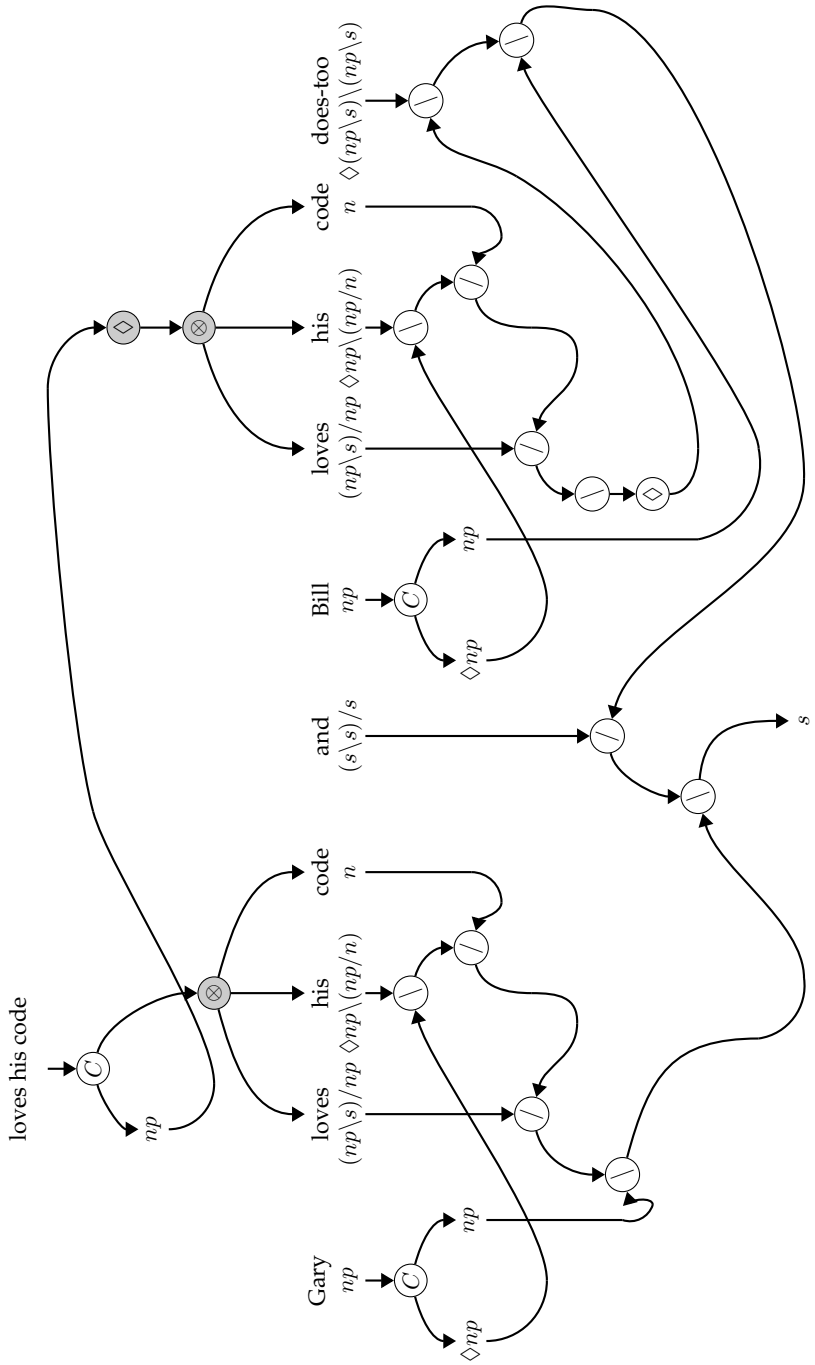


FIGURE 3.28: Syntactic information flow for the sloppy reading of "Bill loves his code and Gary does too".

3.5.4 Frobenius Semantics for Ellipsis with Anaphora

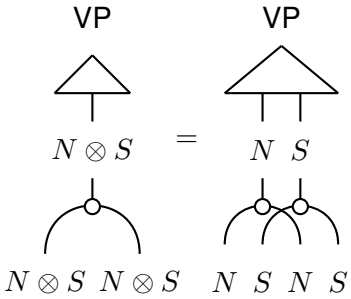
In order to give the vectorial semantics for the derivations for ellipsis with anaphora above, we proceed in the same fashion as for the case of relative pronouns and parasitic gaps, largely taking over the type and proof interpretation defined in section 3.2. The only extra step we need to take is to give an interpretation to the alternative structural rules that we proposed. Interpretation for the associativity structural rules is immediate via the standard associativity of **FVect**: $\lceil \widehat{\alpha^{-1}}(f) \rceil = \lceil f \rceil \circ \alpha^{-1}$ and $\lceil \widehat{\alpha^{-1}}(f) \rceil = \alpha \circ \lceil f \rceil$. For the other structural rules, we additionally use the symmetry maps of **FVect** as well as the diagonal Frobenius map Δ:

$$\lceil \widehat{C}(f) \rceil = \lceil A \rceil \xrightarrow{\Delta_{\lceil A \rceil}} \lceil A \rceil \otimes \lceil A \rceil \xrightarrow{\lceil f \rceil} \lceil B \rceil$$

$$\lceil \widehat{M}(f) \rceil = (\lceil B \rceil \otimes \lceil A \rceil) \otimes \lceil C \rceil \xrightarrow{\sigma_{\lceil B \rceil, \lceil A \rceil} \otimes 1_{\lceil C \rceil}} (\lceil A \rceil \otimes \lceil B \rceil) \otimes \lceil C \rceil \xrightarrow{\alpha} \lceil B \rceil \otimes (\lceil A \rceil \otimes \lceil C \rceil) \xrightarrow{\lceil f \rceil} \lceil D \rceil$$

$$\lceil \widehat{S}(f) \rceil = \lceil B \rceil \otimes (\lceil A \rceil \otimes \lceil C \rceil) \xrightarrow{\alpha^{-1}} (\lceil B \rceil \otimes \lceil A \rceil) \otimes \lceil C \rceil \xrightarrow{\sigma_{\lceil B \rceil, \lceil A \rceil} \otimes 1_{\lceil C \rceil}} (\lceil A \rceil \otimes \lceil B \rceil) \otimes \lceil C \rceil \xrightarrow{\lceil f \rceil \circ \alpha} D$$

It is important to note how the copying rule, interpreted by precomposition with the Frobenius Δ map, acts recursively on complex types, as many examples of ellipsis involve copying complex types rather than simple types. As noted in Section 2.1, the Frobenius Δ map distributes over the tensor product, as the concrete maps in **FVect** are symmetric, i.e. we have $\Delta_{A \otimes B} = \Delta_A \otimes \Delta_B$. Given that all types are interpreted as tensor products of vector spaces, the copying of resources essentially decomposes to the level of basic vector spaces. For example, copying a verb phrase of type $np\backslash s$, in the Lambek diagrammatic representation of Section 2.1.7 represented by a $\textcircled{C}$ node, will be translation to a string diagram as a decomposed application of Frobenius maps:



This gives us all the ingredients to translate the information flow diagram of Figure 3.22 to get the interpretation for "Alice drinks and Bob does too". Assigning to the coordinator 'and' an intersective meaning by means of the Frobenius combining operation μ, and translating the

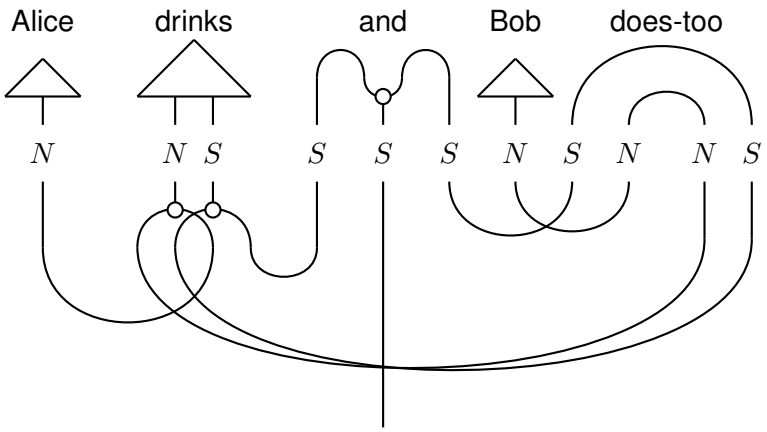


FIGURE 3.29: Semantic information flow for simple verb phrase ellipsis.

Note that we stuck to the translation from Lambek types into pregroup types, thereby introducing a number of crossing wires in the diagram. This reflects the commutativity applications in the proof system that allow for the correct interpretation to be derived.

Given the equations that hold for compact closed categories and Frobenius Algebras, which in our case are special and commutative, we can simplify the diagram to a normal form. This is presented in Figure 3.30.

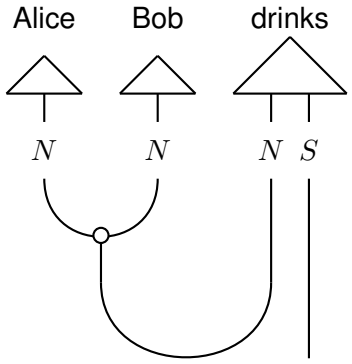


FIGURE 3.30: Information flow for strict ellipsis.

First, the Frobenius Δ map creates to objects in the space S, which then are merged together again by means of the coordinator 'and'. By speciality this then reduces to a single wire. Then, yanking the wire that connects 'Bob' to a copy of the N component of the verb phrase, we can turn around the Frobenius Δ map and get the Frobenius μ map which expresses elementwise multiplication. The final normal form expresses the linear algebraic formula

$$\overline{drink} \times (\overrightarrow{alice} \odot \overrightarrow{bob})$$

which may be seen as computing a meaning that corresponds to "Alice and Bob both

drink".

Structural Ambiguities and Meaning Collapse To interpret the derivations for the strict and sloppy readings of "Gary likes his code and Bill does too" in Figures 3.25 and 3.26, we keep the lexical meaning of the coordinator 'and' to be expressed by element wise multiplication, using the Frobenius map μ, as the vector analogue of the intersective formal semantics term $\lambda x^t y^t . x \wedge y$. For the anaphor 'his', a formal semantics account would assign a term $\lambda x^e P^{e \to t} . \mathbf{own}(x, P)$. Since we assign the same vector space to both common nouns and noun phrases, we can model this in a natural way using again the Frobenius μ map. Finally, the auxiliary verb functions as an identity on the main verb. Its lambda term would simply by $\lambda P^{e \to t} . P$, in vectorial terms we use a rewiring using caps. Putting this together, the two different proofs lead to the two seemingly different diagrams in Figures 3.31 and 3.32, which also seemingly diverge. However, rewriting the diagrams leads in both cases to the normal form diagram in Figure 3.33, which gives the algebraic interpretation below, where the element wise multiplication of all nouns in the sentence are passed on as an argument to both the subject and the object position of the verb, by means of the Frobenius μ map.

$$(\overline{\overline{likes}} \times \mu(\overrightarrow{gary} \odot \overrightarrow{code} \odot \overrightarrow{bill})_1) \times \mu(\overrightarrow{gary} \odot \overrightarrow{code} \odot \overrightarrow{bill})_2$$

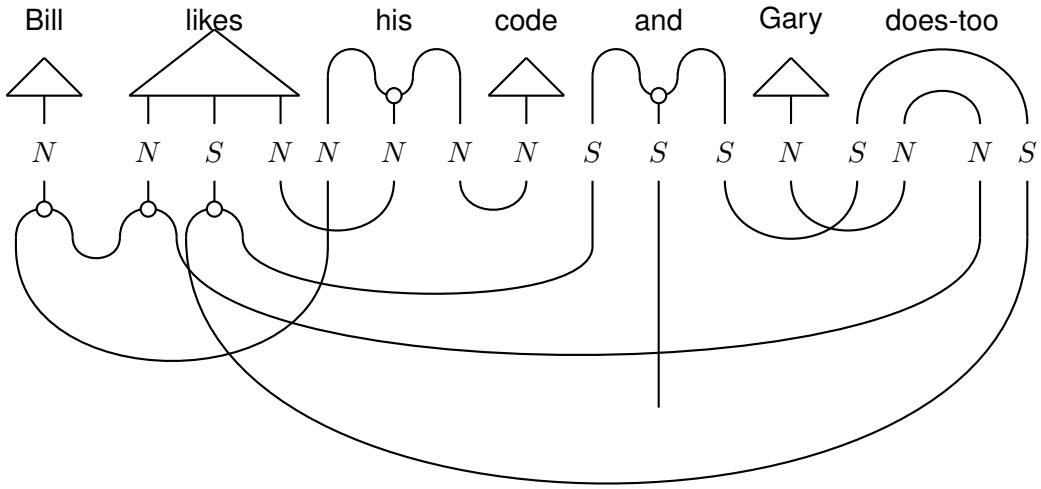


FIGURE 3.31: Semantic information flow for strict ellipsis.

This result shows that interpreting the copy rule as a Frobenius map, in combination with a direct vectorial analogue of the lexical semantics of the anaphor 'his', the coordinator 'and', and the auxiliary 'does too', not only leads to an nonsensical interpretation of complex cases of ellipsis, but moreover to the identification of strict and sloppy identities. The main problem is that the copying with Frobenius Algebras is

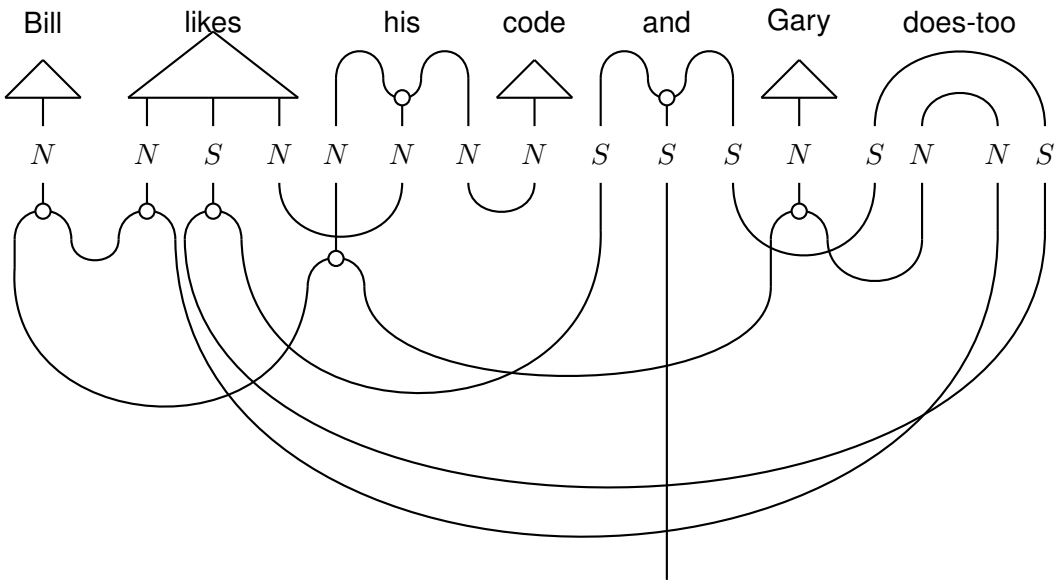


FIGURE 3.32: Semantic information flow for sloppy ellipsis.

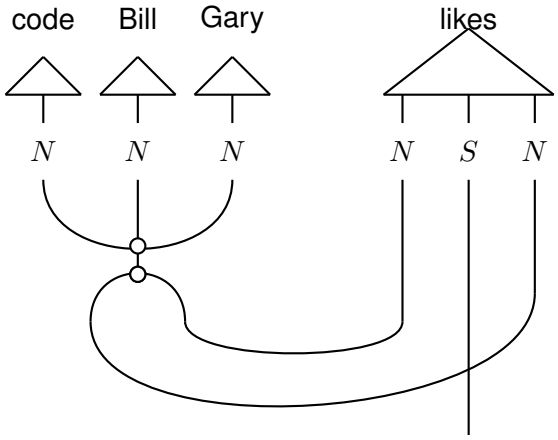


FIGURE 3.33: Normal form diagram for strict ellipsis.

effectuated by a non-cartesian linear map in which the material that was copied, is entangled.

We argue that taking a direct interpretation in the category **FVect** may not be a suitable way of representing these cases which require a cartesian way of copying material. The main Frobenius map that is used for relative pronouns by [SCC13; MW17] expresses element wise multiplication, but its dual map copies a vector by placing its values on the diagonal of a square matrix. In terms of a type signature this indeed multiplies the vector space on which the map is performed, but does not allow for the actual vector to be used in a non-entangled way. In fact, there is no linear map that can copy arbitrary vectors in the cartesian sense [Abr09; Jac11]. To see this for a concrete example, consider the phrase "Alice loves herself" with

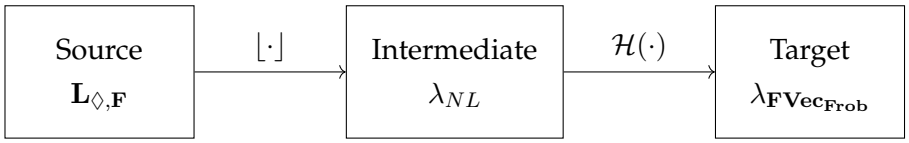
tensors $\mathbf{alice} = \sum_i a_i \vec{v}_i$, and $\mathbf{loves} = \sum_{jkl} c_{jkl}(\vec{v}_j \otimes \vec{s}_k \otimes \vec{v}_l)$. The interpretation of a classical semantics (left) differs from the result of using Frobenius algebras (right):

Classical	**Frobenius**
$\mathbf{alice}_i\mathbf{loves}_{ijk}\mathbf{alice}_k = \sum_{ijk} a_i c_{ijk} a_k \vec{s}_j$	$\mathbf{alice}_i\mathbf{loves}_{iji} = \sum_{ij} a_i a_i c_{iji} \vec{s}_j$

In order to get around the problem and be able to copy semantic material in the intended cartesian way, one has no choice but to move away from a strict linear algebraic setting. We do so in the next section, by interpreting proofs in a simply typed lambda calculus — category-theoretically speaking the internal language of a cartesian closed category [LS88] — in which terms can now model embeddings: the meaning of a sentence is a program with non-linear access to word embeddings.

3.5.5 Lambdas and Tensors for Ellipsis

In order to allow classical copying behaviour in a compositional distributional model, we decompose the categorical model of [CGS13] into a two-step architecture: derivations and are now mapped onto terms of a non-linear simply typed lambda calculus λ_{NL}. The second stage of the interpretation process replaces the assumed lexical constants for words by their lexical semantics, finally resulting in a term of a lambda calculus that models vectors and linear maps, denoted $\lambda_{\mathbf{FVec}_{\mathbf{Frob}}}$. In a picture:



We model this by moving to a setting of simply typed lambda calculus, which categorically speaking is the internal language of a *cartesian closed category*: the equivalence classes of reductions of simply typed lambda terms correspond to the morphisms of a cartesian closed category freely generated by the terms of the calculus [LS88]. Rather than mapping types to vector spaces and proofs to linear maps, we now map types to types of a typed lambda calculus system, with proofs mapped onto terms of this calculus. The terms were already defined in Section 2.1.5, but we repeat the formulation of the terms of the typed lambda calculus with products here:

Definition 4 *Given a countably infinite set of variables $V = \{x, y, z...\}$, terms of λ are as in the below grammar:*

$$M, N := V \mid \lambda x.M \mid M\ N \mid \langle M,\ N \rangle \mid \pi_1(M) \mid \pi_2(M)$$

For the basic types, we set

$$\lfloor np \rfloor = e \qquad \lfloor n \rfloor = e \to t \qquad \lfloor s \rfloor$$

We can then give extend the interpretation of types and proofs from Section 2.1.5 to include the control modalities. The interpretation of the Lambek connectives remain unchanged and the control modalities are again semantically vacuous, hence we define on complex types

$$\lfloor A \otimes B \rfloor = \lfloor A \rfloor \times \lfloor B \rfloor \quad \lfloor A/B \rfloor = \lfloor A \backslash B \rfloor = \lfloor A \rfloor \to \lfloor B \rfloor \quad \lfloor \Diamond A \rfloor = \lfloor \Box A \rfloor = \lfloor A \rfloor$$

For the proofs, we can interpret identity and composition straightforwardly by $\lambda x.x$ and $\lambda x.N\ (M\ x)$ for M and N the terms of the subproofs, respectively. The binary residuation rules correspond to application and abstraction depending on the direction in which the rule is applied, whereas unary residuation does not change the terms at all:

$$\lfloor \triangleright M \rfloor = \lambda x\ y.M\ \langle x, y \rangle \qquad \lfloor \triangleleft M \rfloor = \lambda y\ x.M\ \langle x, y \rangle \qquad \lfloor \triangledown M \rfloor = M$$
$$\lfloor \triangleright^{-1} N \rfloor = \lambda \langle x, y \rangle.(N\ x)\ y \qquad \lfloor \triangleleft^{-1} N \rfloor = \lambda \langle x, y \rangle.(N\ y)\ x \qquad \lfloor \triangledown^{-1} N \rfloor = N$$

The derived monotonicity rules get the interpretation below:

$$\lfloor M \otimes N \rfloor = \lambda \langle x, y \rangle.\langle M\ x, N\ y \rangle$$

$$\lfloor M \backslash N \rfloor = \lambda f\ x.N\ (f\ (M\ x))$$

$$\lfloor M / N \rfloor = \lambda f\ x.M\ (f\ (N\ x))$$

The associativity rules behave as an identity since associativity is implicit in lambda terms.

$$\lfloor \widehat{\alpha}^{l}_{\diamond}(M) \rfloor = \lambda \langle x, y, z \rangle.M\ \langle x, y, z \rangle \qquad \lfloor \widehat{\alpha}^{r}_{\diamond}(M) \rfloor = \lambda \langle x, y, z \rangle.M\ \langle x, y, z \rangle$$

The non-linear behaviour enters with the interpretation for the structural rules for movement and copying:

$$\lfloor \widehat{C}(M) \rfloor = \lambda x.M\ \langle x, x \rangle$$
$$\lfloor \widehat{M}(M) \rfloor = \lambda \langle y, x, z \rangle.M\ \langle x, y, z \rangle$$
$$\lfloor \widehat{S}(M) \rfloor = \lambda \langle y, x, z \rangle.M\ \langle x, y, z \rangle$$

Abstract Proof Terms for Ellipsis The interpretation above allows us to give an abstract meaning representation for the proofs involving ellipsis and anaphora, in which each word is representation by some constant, that later on can refer to a concrete meaning. For the sample derivation of Figure 3.21, the logical phase computes only reductions; the subproof of the type

$$(np \otimes np\backslash s) \otimes ((s\backslash s)/s \otimes (np \otimes (\Diamond(np\backslash s) \otimes\ \Diamond(np\backslash s)\backslash(np\backslash s)))) \longrightarrow s$$

gives an abstract term

$$\lambda\langle \texttt{subj}_1, \texttt{verb}, \texttt{coord}, \texttt{subj}_2, \texttt{verb}^*, \texttt{aux}\rangle.(\texttt{coord}\ ((\texttt{aux}\ \texttt{verb}^*)\ \texttt{subj}_2))(\texttt{verb}\ \texttt{subj}_1)$$

and the structural phase repositions the copy $\texttt{verb}^*$ next to the verb, after which the contraction rule *identifies* the variables associated with them, unifying $\texttt{verb}$ and $\texttt{verb}^*$:

$$\lambda\langle \texttt{subj}_1, \texttt{verb}, \texttt{coord}, \texttt{subj}_2, \texttt{aux}\rangle.(\texttt{coord}\ ((\texttt{aux}\ \texttt{verb})\ \texttt{subj}_2))(\texttt{verb}\ \texttt{subj}_1)$$

We get the final *abstract proof term* for the proof in Figure 3.21 by applying the term above to the constants for the words in the sentence:

$$(\texttt{and}\ ((\texttt{dt}\ \texttt{drinks})\ \texttt{bob}))(\texttt{drinks}\ \texttt{alice}) : s \tag{3.17}$$

For the cases of strict and sloppy readings of ellipsis and anaphora, the difference in order of the ellipsis versus anaphor resolution is reflected in the two different abstract meaning terms below:

$$(\texttt{and}\ ((\texttt{dt}\ (\texttt{loves}\ ((\texttt{his}\ \texttt{gary})\ \texttt{code})))\ \texttt{bill}))\ ((\texttt{loves}\ ((\texttt{his}\ \texttt{gary})\ \texttt{code}))\ \texttt{gary}) \qquad (\textit{strict})$$

$$(\texttt{and}\ ((\texttt{dt}\ (\lambda t.(\texttt{loves}\ ((\texttt{his}\ t)\ \texttt{code}))\ t))\ \texttt{bill}))\ ((\texttt{loves}\ ((\texttt{his}\ \texttt{gary})\ \texttt{code}))\ \texttt{gary}) \qquad (\textit{sloppy})$$

In order to give a concrete vector semantics for these abstract proof terms, we need to encode vectors and their operations using lambda calculus.

Modelling Vector Semantics Vectors can be seen as functions from natural numbers to the values in the underlying field, allowing us to represent them naturally as lambda terms. For any dimensionality n, we assume a basic type I_n, representing a finite index set (in concrete models the number of index types will be finite). The underlying field, in our case the real numbers $\mathbb{R}$, is given by the type R.

The type of a vector in $\mathbb{R}^n$ is now $V^n = I_n \to R$, the type of an $n \times m$ matrix is $M^{n\times m} = I_n \to I_m \to R$. In general, we may represent an arbitrary tensor with dimensions $n, m, ..., p$ by $T^{n\times m\ldots\times p} = I_n \to I_m \to ... \to I_p \to R$. We will leave out the superscripts denoting dimensionality when they are either irrelevant or understood from the context.

By reference to index notation for linear algebra, we write $v\ i$ as v_i whenever it is understood that i is of type I. We moreover assume constants for the basic operations of a vector space: $0 : R, 1 : R, + : R \to R \to R, \cdot : R \to R \to R$ with their standard interpretation. Standard operations can now be expressed:

Name	**Symbol**	Lambda term
Matrix transposition	$.^T$	$\lambda mij.m_{ji} : M \to M$
Matrix multiplication	$\times_1$	$\lambda mvi.\sum_j m_{ij} \cdot v_j : M \to V \to V$
Cube multiplication	$\times_2$	$\lambda cvij.\sum_k c_{ijk} \cdot v_k : C \to V \to M$
Element wise multiplication	$\odot$	$\lambda uvi.u_i \cdot v_i : V \to V \to V$

All of these operations, except for addition, are instances of the multilinear algebraic operation of tensor contraction applicable to any two tensors of arbitrary rank as long as they share at least one index. The tensor contraction between them is formed by applying the following formula:

$$\sum_{i_1,\dots,i_{n+k}} A_{i_1 i_2 \cdots i_n} B_{i_n i_{n+1} \cdots i_{n+k}} \in \underbrace{W \otimes \cdots \otimes W}_{n+k-1}$$

$$\text{For} \sum_{i_1,\dots,i_n} A_{i_1 i_2 \cdots i_n} \in \underbrace{W \otimes \cdots \otimes W}_{n} \quad \text{and} \quad \sum_{i_n,\dots,i_{n+k}} B_{i_n i_{n+1} \cdots i_{n+k}} \in \underbrace{W \otimes \cdots \otimes W}_{k+1}$$

Element wise multiplication between two vectors, or matrices, or tensors of the same rank is also an instance of tensor contraction, where one of the arguments of the multiplication is *raised* to a tensor of a higher rank, with the argument in its diagonal and its other entries padded with zero. For an instance of this see [Kar16] where coordination is treated in a DisCoCat model, and the where the author shows how the linear algebraic closed form of element wise multiplication arises as a result of a tensor contraction.

To obtain a concrete model for a phrase, we homomorphically replace the abstract meaning term of a proof by a concrete tensor mapping. Since we map lambda terms to lambda terms, we only need to specify how constants c are mapped to tensors. This will automatically induce a type-respecting term homomorphism $\mathcal{H}$. A general map that sends constants to a contraction friendly model is presented in Table 3.1.

w	$\sigma(w)$	$\mathcal{H}(w)$	$\mathcal{T}(w)$
`cn`	n	$\mathbf{cn}$	V
`adj`	n/n	$\lambda v.(\mathbf{adj} \times_1 v)$	VV
`adv`	$(np\backslash s)\backslash(np\backslash s)$	$\lambda m.(\mathbf{adv} \times_3 m)$	HM
`itv`	$np\backslash s$	$\lambda v.(\mathbf{itv} \times_1 v)$	VV
`tv`	$(np\backslash s)/np$	$\lambda uv.(\mathbf{tv} \times_2 u) \times_1 v$	VVV
`coord`	$(s\backslash s)/s$	$\lambda P.\lambda Q.P \odot Q$	VVV

TABLE 3.1: Translation that sends abstract terms to a tensor-based model using matrix and cube multiplication as the main operations; here an in the two other proceeding tables the atomic types are np and s.

The composition defined in Table 3.1 is in a sense external to the term calculus itself; using the abstract mechanism of function application, we can specify any linear map as an internal operation in the semantic lexicon. For the adjectival phrase we use matrix multiplication, and for an adverbial modifier we model it using a hypercube contraction, for the coordinator we can insert the element wise multiplication.

The translation of the proof term of Figure 3.21 under Table 3.1 becomes as follows:

$$(\texttt{and}\ ((\texttt{dt}\ \texttt{drinks})\ \texttt{bob}))(\texttt{drinks}\ \texttt{alice}) : s$$

Substituting the concrete terms we get the following β-reduced version:

$$\twoheadrightarrow_\beta (\mathbf{drinks} \times_1 \mathbf{alice}) \odot (\mathbf{drinks} \times_1 \mathbf{bob})$$

As another alternative, we can instantiate the proof terms in a multiplicative-additive model. This is a model where the sentences are obtained by adding their individual word embeddings and the overall result is obtained by multiplying the two sentence vectors. This model is presented in Table 3.2, according to which we obtain the following semantics for our example sentence above:

$$\twoheadrightarrow_\beta (\mathbf{drinks} + \mathbf{alice}) \odot (\mathbf{drinks} + \mathbf{bob})$$

Another alternative is Table 3.3, which provides the same terms with a Kronecker -based tensor semantics, originally used by [GS11a] to model transitive sentences.

w	$\sigma(w)$	$\mathcal{H}(w)$	$\mathcal{T}(w)$
`cn`	n	$\mathbf{cn}$	V
`adj`	np/n	$\lambda v.(\mathbf{adj} + v)$	VV
`adv`	$(np\backslash s)\backslash(np\backslash s)$	$\lambda m.(\mathbf{adv} + m)$	VV
`itv`	$np\backslash s$	$\lambda v.(\mathbf{itv} + v)$	VV
`tv`	$(np\backslash s)/np$	$\lambda uv.(\mathbf{tv} + u + v)$	VVV
`coord`	$(s\backslash s)/s$	$\lambda P.Q.(P \odot Q)$	VVV

TABLE 3.2: Translation that sends abstract terms to a multiplicative-additive model.

We symbolise the semantics of the basic elliptical phrase that comes out of any of these models for our example sentence as follows:

$$M(\mathbf{sub}_1, \mathbf{verb}) \star M(\mathbf{sub}_2, N(\mathbf{verb}))$$

where M is a general term for an intransitive sentence, N is a term that modifies the verb tensor through the auxiliary verb, and $\star$ is an operation that expresses the coordination of the two subclauses. For a transitive sentence version, the above changes to the following:

$$M(\mathbf{subj}_1, \mathbf{verb}, \mathbf{obj}_1) \star M(\mathbf{subj}_2, N(\mathbf{verb}), \mathbf{obj}_1)$$

Such a description is very general, and in fact allows us to derive almost all compositional vector models that have been empirically evaluated in the literature (see e.g., [Mil+14]). This flexibility is necessary for ellipsis because it can model the Cartesian behaviour that is unavailable in a categorical modelling of vectors and linear maps. Some models can, however, only be incorporated by changing the lexical formulas associated to the individual words. The proposal of Kartsaklis et al [KPS16] is one such example. They use the coordinator to a heavy extent and their typing and vector/tensor assignments result in the following lambda

semantics for the phrase "Alice drinks and Bob does too":

$$\mathbf{drinks} \times_1 (\mathbf{alice} \odot \mathbf{bob})$$

The above is obtained by assigning an identity linear map to the auxiliary phrase 'does too' and then assigning a complex linear map to the coordinator 'and' tailored in a way that it guarantees the derivation of the final meaning. In our framework, we would need to take a similar approach, and we need to modify M to essentially return the verb-subject pair, N would be the identity, and `and` has to be defined with the tailored to purpose term below, which takes two pairs of subjects and verbs, but discards one copy of the verb to mimic the model of Kartsaklis et al [KPS16]:

$$\texttt{and} \qquad \lambda\langle s, v\rangle.\lambda\langle t, w\rangle.v \times_1 (s \odot t)$$

w	$\sigma(w)$	$\mathcal{H}(w)$	$\mathcal{T}(w)$
`cn`	n	$\mathbf{cn}$	V
`adj`	np/n	$\lambda v.(\mathbf{adj} \odot v)$	VV
`adv`	$(np\backslash s)\backslash(np\backslash s)$	$\lambda m.(\mathbf{adv} \odot m)$	VV
`itv`	$np\backslash s$	$\lambda v.(\mathbf{itv} \odot v)$	VV
`tv`	$(np\backslash s)/np$	$\lambda uv.(\mathbf{tv} \odot (v \otimes u))$	VVM
`coord`	$(s\backslash s)/s$	$\lambda P.\lambda Q.(P \odot Q)$	VVV

TABLE 3.3: Translation that sends abstract terms to a Kronecker model. We abuse the notation to denote the element wise multiplication of two matrices with the same symbol, i.e. $\odot$, as the element wise multiplication of two vectors.

All in all, we can reasonably derive a large class of compositional functions that can be experimented with in a variety of tasks. If we assume a tensor-based compositional model that uses tensor contraction to obtain the meaning of a sentence, we get the two different meanings for the strict and sloppy readings as follows:

1. $((\mathbf{loves} \times_2 (\mathbf{gary} \odot \mathbf{code})) \times_1 \mathbf{gary}) \odot ((\mathbf{loves} \times_2 (\mathbf{gary} \odot \mathbf{code})) \times_1 \mathbf{bob})$ (strict)

2. $((\mathbf{loves} \times_2 (\mathbf{gary} \odot \mathbf{code})) \times_1 \mathbf{gary}) \odot ((\mathbf{loves} \times_2 (\mathbf{bob} \odot \mathbf{code})) \times_1 \mathbf{bob})$ (sloppy)

3.6 Conclusion

This chapter presented the first main line of contributions of this thesis. We extended the categorical compositional framework of Coecke, Sadrzadeh, and Clark [CSC10] and Coecke, Grefenstette, and Sadrzadeh [CGS13] and introduced a vector semantic interpretation for an extension of the Lambek Calculus with unary modalities. We then reviewed how existing such systems can be used to model pronoun relativisation in English and in Dutch. Then, we extended the previous setting where all the copying behaviour is encoded in the lexicon, to the case of parasitic gaps.

Finally, we adapted the existing system of Moortgat [Moo96] with a different set of structural rules to allow for a controlled form of copying and movement of linguistic material, and gave two different semantics for this proof system. The first strictly follows the categorical compositional framework of Coecke, Sadrzadeh, and Clark [CSC10] and Coecke, Grefenstette, and Sadrzadeh [CGS13] but gave undesired results in the case of ellipsis combined with anaphora, as strict and sloppy readings of the same sentence would generate the exact same vector semantics. The second semantics was based on the work of Muskens and Sadrzadeh [MS17], and used the simply typed lambda calculus, allowing for cartesian copying and thereby restoring the desired meaning representations for ambiguous elliptical phrases.

The developed theory of this chapter gives two different accounts for different types of ellipsis: the approach in which the proof system is kept linear and all instances of copying need to be lexically encoded preserves a decidable proof system, but has the downside of overloading the lexicon and the need to annotate parts of a phrase with modalities in order before the proof theoretic analysis proceeds. The other approach, in which a limited form of copying is allowed in the proof system avoids lexical polymorphism and extra parsing requirements, but does lead to a potentially undecidable system, as a number of relevant and recent results suggest [KKS16; KKS17; KKS19].

Given that we can now instantiate the developed vector models, there is the opportunity to validate different theoretical approaches on concrete experimental tasks. This is the approach of the next chapter of this thesis, where we continue with ellipsis from a practical viewpoint: we evaluate the models developed thus far, introducing three novel datasets to do so.

Part III

Practice

Chapter 4

Evaluation: Composition Models for Verb Phrase Ellipsis

Chapter Abstract

In this chapter we discuss the results of empirically validating compositional distributional models of meaning on verb phrase ellipsis. We start by introducing the general approach taken before in evaluating word and sentence similarity, and then introduce three new datasets and carry out a number of experiments. The core contributing material is mainly based on [WS19b] but some of the evaluation results and background come from [WS19a] and [WS18] (all in collaboration with Mehrnoosh Sadrzadeh).

There is a long standing tradition of evaluating the similarity between words and sentences, as we already hinted at in Chapter 1. On the word level, datasets that allow one to compare a model's performance against human similarity judgments have been developed as far back as the 1960's with Rubenstein and Goodenough's dataset [RG65], and larger and more complex datasets followed [MC91; Fin+01; YP06; Bru+12; HRK15; Ger+16].

On the sentence level, two important datasets arose from the work of Mitchell and Lapata [ML08; ML10], who were one of the first to systematically experiment with composing word embeddings. This was followed by the work of Sadrzadeh and colleagues to develop datasets for more complex sentences [GS11a; KSP13; KS13]. Later, with the introduction of advanced neural network models for sentence embeddings, large-scale datasets were introduced that measure not just similarity between sentences, but textual entailment, and natural language inference. Prime examples of these arbitrary length sentence comparison datasets are the SICK dataset (Sentence Involving Compositional Knowledge) introduced by Marelli et al. [Mar+14], the Stanford Natural Language Inference dataset (SNLI) introduced by Bowman et al. [Bow+15] and its follow up datasets MNLI (multigenre) [WNB18] and XNLI (crosslinguistic) [Con+18].

The phenomena of ellipsis, pronoun relativisation and parasitic gaps all fall in between these datasets: the datasets of Mitchell and Lapata, as well as the datasets introduced by Grefenstette, Kartsaklis and Sadrzadeh cover intransitive and transitive sentences without complex constructions such as verb phrase ellipsis or parasitic gapping. On the other hand, the large-scale inference datasets that contain arbitrary length sentence pairs are too general to properly evaluate models for the phenomena of interest to this thesis, and barely contain examples of ellipsis and anaphora in the first place. For the case of relative pronouns there is the RELPRON dataset, introduced by Rimell and colleagues [Rim+16]. The authors show there that the tensor-based modelling for pronoun relativisation in English is not able to

outperform a simple additive model. Models for pronoun relativisation and ellipsis as we outlined them in Chapter 3 both involve the movement of semantic information, but the main difference with ellipsis is that in the latter case, there is semantic information that is not explicitly available in a phrase. We modelled ellipsis by means of copying semantic information, but evaluating these models poses a different challenge than relative pronouns: not only do we need to test which way of modelling the ellipsis is the most suitable in an experimental setting, but we also need to design a study in such a way that it allows one to test whether resolving ellipsis is useful in the first place.

So the core problem that this chapter addresses, is the empirical evaluation of the models that we developed in Chapter 3 to deal with *ellipsis*. In such sentences, there is a part of a sentence missing, that often can be recovered from context. In the case of verb phrase ellipsis, this context is a verb phrase, usually marked by an auxiliary verb. Taking inspiration from such phenomena, and building on previous datasets that task disambiguation in context and sentence similarity, we introduce three novel datasets for verb phrase elliptical constructions. These datasets now serve a dual purpose: on the one hand they provide larger datasets than were available before to compare concrete models on general disambiguation and similarity tasks, but in addition they allow us to compare models that perform linguistic analysis to resolve ellipsis with naive models that linearly encode the surface form sentence that is given. Because these datasets are based on those constructed by Mitchell and Lapata, and by Sadrzadeh and colleagues, we first discuss these tasks in this chapter, recomputing performance based on the concrete models that we use on the ellipsis datasets.

The models we developed in Chapter 3 for ellipsis come in two flavours: first, there is the purely categorical approach, which we referred to as giving the *Frobenius semantics* of ellipsis as the syntactic copying mechanism is interpreted by a Frobenius Algebra. Secondly, we defined the looser lambda-based modelling, that allows in fact to derive any of the concrete compositional models that were evaluated before, e.g. by Milajevs et al. [Mil+14]. We give an overview of those models and show how we can connect the theoretical models to concrete composition operations for evaluation.

This chapter is structured as follows: first, we discuss existing datasets for similarity on the word level, and move to resources for disambiguation and similarity of sentences. We then illustrate how we can concretely construct sentence embeddings for verb phrase elliptical sentences. Then, we introduce four new datasets, two of which are given new annotations using the Amazon Mechanical Turk crowdsourcing platform. We compare the interannotator agreement for these datasets to argue that introducing verb phrase ellipsis should help a human to more easily judge verb disambiguation in context. In the next section we discuss the concrete vector spaces that we create, and concrete encoding models, based either on the tensor-based models that we developed in the previous chapter or on state of the art sentence encoder models from the NLP literature. We give results on all the datasets discussed, and proceed to investigate what the results tell us about our embedding models and different choices of embedding verb phrase elliptical sentences. Finally, we conclude with some directions for future research.

4.1 Evaluating Composition Models

We start our discussion of related work on evaluating embeddings on the sentence level, specifically in the setting of tensor-based models, which is the focus of this thesis.

Intransitive sentences The first datasets that investigated compositionality of distributional models were introduced by Mitchell and Lapata [ML08; ML10]. The first dataset, which we call **ML2008**, is a verb disambiguation dataset. It contains pairs where the subject of both sentences is the same but the verb differs:

export booms vs. *export prospers*
export booms vs. *export thunders*

The idea is that a verb like 'boom' could be ambiguous between 'prosper' and 'thunder', and that the subject of the verb ought to be relevant for an unambiguous interpretation of the verb. Here, having 'export' as a subject, the verb is more likely to mean 'prosper', but if it the subject is replaced by 'gun', the more likely interpretation changes to 'thunder'. The dataset contains a total of 120 intransitive sentences, generated by considering 15 verbs and two landmark interpretations for each of them, and combining those verb triples with four different nouns, two for each landmark interpretation. The authors marked in each case the more likely interpretation with HIGH, and the less likely interpretation with LOW, then had humans annotate all the sentence pairs for similarity ratings. The HIGH and LOW marks then served as a check to see whether human annotators would indeed assign higher similarity to HIGH cases, and lower similarity to LOW cases.

The second dataset that Mitchell and Lapata introduced, was also a dataset containing pairs of two word phrases, with a mixture of *noun-noun* compounds, *adjective-noun* combinations, and *verb-object* sentences. Here we refer to this dataset as **ML2010**, and work with the part of the dataset that contains *verb-object* sentences, as we are evaluating verb phrase ellipsis. In contrast to the **ML2008** dataset, these intransitive sentence pairs now share no lexical overlap because the dataset is meant to test for similarity. For example, the dataset contains sentence pairs like

consider matter vs. *discuss issue*
like people vs. *increase number*

The **ML2010** contains 108 distinct sentence pairs. These were now unmarked, but annotated by humans with similarity judgments. The aggregated average judgments then form the basis for evaluation of this dataset.

Transitive sentences Grefenstette and Sadrzadeh [GS11a] introduced a dataset that followed the same approach as Mitchell and Lapata for the **ML2008** dataset (verb disambiguation), but considered transitive verbs rather than intransitive ones, i.e. verbs that go with both a subject and an object. This dataset we call **GS2011**. Similar to **ML2008**, the **GS2011** verb disambiguation dataset contains 10 verbs, each with two possible interpretations. For each verb v and its two interpretations v_1 and v_2, the dataset contains human similarity judgments for 10 subject-object combinations. For instance, for the verb *meet* – ambiguous between *visit* and *satisfy* – the dataset contains the pairs below:

system meet requirements vs. *system satisfy requirements*
system meet requirements vs. *system visit requirements*

As with the **ML2008** dataset, the more likely interpretation is marked as HIGH whereas the unlikely interpretation is marked LOW, in order to verify the authors' intuition against the annotators' judgments.

Kartsaklis, Sadrzadeh, and Pulman [KSP13] introduce another verb disambiguation dataset, which we refer to as **KS2013** (though it was actually produced by Grefenstette and Sadrzadeh).

The **KS2013** dataset is a modified version of the **GS2011** dataset: first, the **GS2011** was extended with adjectives to go with the subject and object of each sentence in previous work [GS15]. Then, Kartsaklis, Sadrzadeh, and Pulman [KSP13] took out ten verbs and replaced them with genuinely ambiguous verbs, based on the study of Pickering and Frisson [PF01]. This dataset contains a total of 194 transitive sentence pairs. A different dataset was constructed by Kartsaklis and Sadrzadeh [KS13] as an extension of the **ML2010** dataset. Because this dataset was re-annotated by crowdsourced annotators for a later publication [KS14], we refer to this dataset as **KS2014**. It contains 108 transitive sentence pairs annotated with human similarity judgments. We show two such examples below:

government use power vs. *authority exercise influence*
pupil achieve end vs. *gentleman close eye*

As opposed to the **GS2011** dataset, and similar to the **ML2010** dataset, subjects and objects of each sentence pair are not the same, so several different contexts get compared to one another. In this sense, the **KS2014** dataset aims to investigate the role of content of individual words versus the role of composition, as the similarity of sentences might be predictable from the contribution of individual words rather than the specific way of composing them.

Inter-annotator agreement An important aspect of any similarity dataset is the *inter-annotator agreement*: the ratings given by individual annotators are necessarily subjective, but most importantly, multiple annotators may not agree in their judgment. An often taken approach is to produce a final dataset that contains the average human judgments, but also supplying the inter-annotator agreement as the average pairwise correlation between two annotators, or even the average correlation of a single annotator against the average judgment of all other annotators [HRK15]. The inter-annotator agreement gives an upper bound on the performance of a model: it should perform at least as good as the average of the annotators, but when it outperforms the inter-annotator agreement by a large margin, it may indicate that either the model is overfitting on the dataset, or that the dataset may be too simplistic.

Here, we choose to report the average correlation of a single annotator against the average of all judgments, for two reasons. As Hill, Reichart, and Korhonen [HRK15] note, this is more fair since the average of all annotators is used as the gold standard against which a model is compared. Moreover, it is more robust against missing data points: in large datasets, a given annotator may have judged a disjoint part of the total dataset with respect to some other annotator. However, taking subsets of the dataset to compare individual annotators does not reflect well the inter-annotator agreement as Spearman's ρ is not distributive, i.e. the average between the Spearman's ρ of two subsets is not the same as the Spearman's ρ on the full dataset. So, we opt for comparing an individual annotator with the average of all other judgments on those datapoints that the annotator has judged.

Table 4.1 shows the inter-annotator agreement for the sentence pair datasets discussed above, when computed by the mean spearman correlation between an individual annotator and the average of all other annotators.

Modelling Intransitive and Transitive Sentences To concretely model sentence embeddings for the verb phrase elliptical sentence datasets we described above, we draw inspiration from the literature on formally modelling ellipsis, the work described in Chapter 3, and previous evaluation studies in the context of tensor-based models [ML08; ML10; GS11a; GS11b; KS13; KSP13; Mil+14].

Name	IAA 1 vs. all	Source
ML2008	0.66	recomputed
ML2010	0.711	recomputed (only *verb-object* sentences)
GS2011	0.739	recomputed
KS2013	0.575	recomputed (forgetting the adjectives)
KS2014	0.754	recomputed

TABLE 4.1: Inter-annotator agreement scores, by taking the average correlation between single annotators versus the average score of all other annotators.

On the level of intransitive sentences, the overview article of Grefenstette and Sadrzadeh [GS15] lists a number of concrete composition models that were tested by Mitchell and Lapata [ML08] and Grefenstette and Sadrzadeh [GS11a]. Grefenstette and Sadrzadeh experiment with the unsupervised methods of Mitchell and Lapata [ML08], that is, these models perform an unweighted addition or multiplication. Moreover, they compare these with a verb only baseline, in which the vector for an intransitive sentence is given by the vector for the verb in that sentence, and with a categorical model, in which the verb is represented as a matrix, that applies to its subject (resp. object) vector. Although the datasets contain both subject-verb and verb-object combinations, for the sake of simple presentation we present the composition models for subject-verb combinations in Table 4.2.

Name	Sentence	Formula
Verb only	*subj verb*	$\overrightarrow{verb}$
Additive	*subj verb*	$\overrightarrow{subj} + \overrightarrow{verb}$
Multiplicative	*subj verb*	$\overrightarrow{subj} \odot \overrightarrow{verb}$
Categorical	*subj verb*	$\overrightarrow{subj}^{\top} \times \overline{verb}$

TABLE 4.2: Table of compositional models that were evaluated for intransitive sentences by Mitchell and Lapata [ML08] and Grefenstette and Sadrzadeh [GS11a].

The first baseline model takes just the verb vector and 'forgets' the subject. The two other baseline models either add ($+$) or point-wise multiply ($\odot$) the vectors for the subject and verb. In the categorical model, we use a verb matrix, denoted by $\overline{verb}$, which may take several forms as outlined below. This matrix is backward applied to the subject vector:

$$\overrightarrow{subj}^{\top} \times \overline{verb} = \begin{pmatrix} s_1 & s_2 \ldots & s_n \end{pmatrix} \times \begin{pmatrix} v_{11} & \ldots & v_{1n} \\ v_{21} & \ldots & v_{2n} \\ \vdots & \ddots & \vdots \\ v_{n1} & \ldots & v_{nn} \end{pmatrix} = \begin{pmatrix} v_{11}s_1 + \ldots + v_{n1}s_n \\ v_{12}s_1 + \ldots + v_{n2}s_n \\ \vdots \\ v_{1n}s_1 + \ldots + v_{nn}s_n \end{pmatrix}$$

For transitive sentences, several models were tested throughout the literature. Grefenstette and Sadrzadeh [GS11b] experiment — besides the additive and multiplicative models — with a categorical and a Kronecker model, which respectively multiply two different matrices with a matrix given by the subject and object vectors in a transitive sentence. In the

case of the Relational model, the verb matrix represents an average of the relations between subject and object vectors, their relation being expressed by the outer product $\otimes$. This relational matrix is obtained by taking the sum of the outer product of all subject and object vectors that occurred with the verb in a corpus:

$$\overline{verb} = \sum_i \overrightarrow{subj_i} \otimes \overrightarrow{obj_i}$$

Consequently, the relational models how well this verb approximation fits with the given subject and object in a transitive sentence by point-wise multiplying their features. The Kronecker model swaps the verb representation for one that is uniquely derived from the verb vector by means of the outer product:

$$\widetilde{verb} = \overrightarrow{verb} \otimes \overrightarrow{verb}$$

Besides the fact that the Kronecker model is easy to implement, it also has a theoretical motivation (as discussed in Grefenstette and Sadrzadeh [GS15]). The expressed model is equal to taking the outer product of two separate applications of the verb vector:

$$(\overrightarrow{verb} \otimes \overrightarrow{verb}) \odot (\overrightarrow{subj} \otimes \overrightarrow{obj}) = (\overrightarrow{verb} \odot \overrightarrow{subj}) \otimes (\overrightarrow{verb} \odot \overrightarrow{obj})$$

The fact that the dot product between two outer products is the multiplication of the dot products between operands

$$(\overrightarrow{a} \otimes \overrightarrow{b}) \cdot (\overrightarrow{c} \otimes \overrightarrow{d}) = (\overrightarrow{a} \cdot \overrightarrow{c})(\overrightarrow{b} \cdot \overrightarrow{d})$$

then motivates the Kronecker model as a test between the verb representations ability to combine with the subject and object separately, then merging the result.

Kartsaklis, Sadrzadeh, and Pulman [KSP12] and Kartsaklis and Sadrzadeh [KS14] add to this a number of models based on the use of the Frobenius algebra operation Δ that expands information. The first two such models use the Frobenius Δ map to expand one dimension of the verb matrix to embed the verb as a cube, that can then be contracted with the subject and object vector to produce a sentence vector. Because the Δ essentially places a vector on the diagonal of a matrix, the concrete models that come out multiply the subject (resp. object) vector with the result of applying the verb matrix to the object (resp. subject) vector, hence the appropriate names Copy Subject and Copy Object. These two models can then be combined using addition, multiplication, or outer product to give the Frobenius Additive, Frobenius Multiplicative and Frobenius Outer models. All of these models were evaluated by Milajevs et al. [Mil+14], which focussed on the use of different word embedding techniques with which to combine these compositional models. We summarise the models in Table 4.3.

4.2 Evaluation Datasets for Verb Phrase Elliptical Sentences

Ellipsis poses an interesting test bed for evaluating compositional distributional models, since it is a phenomenon in which the semantic representation does not directly correspond to the textual form of the phrase we are trying to analyse. In terms of concrete evaluation, our main aim therefore is to create a task that allows us to test whether models that resolve the ellipsis — i.e. which recover the implicit semantic content of the elliptical site — will

Name	Abbreviation	Formula
Verb Only	**VO**	$\overrightarrow{verb}/\overline{verb}$
Additive	**ADD**	$\overrightarrow{subj} + \overrightarrow{verb} + \overrightarrow{obj}$
Multiplicative	**MULT**	$\overrightarrow{subj} + \overrightarrow{verb} + \overrightarrow{obj}$
Relational	**RE** $^{-}$	$\overline{verb} \odot (\overrightarrow{subj} \otimes \overrightarrow{obj})$
Kronecker	**RE** $^{\sim}$	$(\overrightarrow{verb} \otimes \overrightarrow{verb}) \odot (\overrightarrow{subj} \otimes \overrightarrow{obj})$
Copy Subject	**CS**	$\overrightarrow{subj} \odot (\overline{verb} \times \overrightarrow{obj})$
Copy Object	**CO**	$\overrightarrow{obj} \odot (\overline{verb}^T \times \overrightarrow{subj})$
Frobenius Additive	**FA**	$\mathbf{CS}(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj}) + \mathbf{CO}(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj})$
Frobenius Multiplicative	**FM**	$\mathbf{CS}(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj}) \odot \mathbf{CO}(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj})$
Frobenius Outer	**FO**	$\mathbf{CS}(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj}) \otimes \mathbf{CO}(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj})$

TABLE 4.3: Table of all compositional models that were evaluated by Milajevs et al. [Mil+14].

perform higher on said task than models that have an ellipsis resolution step. Besides from this main motivation, we choose verb phrase elliptical sentences that extend the intransitive and transitive sentences from the **ML2008**, **GS2011**, and **KS2014** datasets, in order to create a structured dataset that can be related to those originating datasets.

To demonstrate the intuition behind our dataset design, consider the sentence "the man runs", which is ambiguous between "the man races" and "the man stands (for election)". The sentence itself does not have enough context to help disambiguate the verb, but after adding a case of ellipsis such as "the man runs, the dog too", the ambiguity will be greatly decreased. An example of a transitive sentence is "the man draws the sword", which is ambiguous between "the man pulls the sword" and "the man depicts the sword". Again, the current sentential context in which the ambiguous verb occurs may not easily disambiguate its meaning, but after adding the extra context "the soldier does too", the disambiguating effect of the context is much stronger.

We can make this intuition data-driven by considering the following vector space built from (hypothetical) raw co-occurrence counts of several nouns and verbs with respect to a set of context words. The co-occurrence matrix is given in Table 4.4; each row of the table represents a word embedding in the vector space spanned by the column words.
By computing vectors for the sentences mentioned above we can then work out the cosine similarity scores between their vector representations. For sentence triples generated by an ambiguous verb and its two landmark interpretations, we can then see how well different models disambiguate the verb. Following the models in Table 4.5 above, we show how this works for the concrete sentence "the man runs" with the extension of 'governer' and 'athlete' respectively. The idea is that the representation of "the man runs and the governor does too" will be closer to that of "the man stands and the governor does too", whereas the representation of "the man runs and the athlete does too" will be closer to that of "the man races and the athlete does too". The cosine similarity scores for each model are presented in Table 4.6. The original representation of "the man runs" is more similar to "the man races"

	human	painting	army	weapon	marathon	election
man	2	3	4	2	4	4
painter	3	8	1	3	1	1
warrior	4	1	2	9	1	0
sword	2	3	9	2	0	0
picture	1	20	0	1	1	1
governor	7	1	1	3	1	9
athlete	6	2	0	1	9	1
draw	4	10	9	11	2	3
pull	7	2	10	15	1	1
depict	3	15	2	2	1	2
run	4	0	2	1	8	7
race	8	0	0	3	10	3
stand	5	1	0	1	2	11

TABLE 4.4: (Hypothetical) co-occurrence counts for several nouns and verbs.

Model Name	**Formula**
Multiplicative $\odot$	$\overrightarrow{subj} \odot \overrightarrow{verb} \odot \overrightarrow{subj^*} \odot \overrightarrow{verb}$
Multiplicative $+$	$(\overrightarrow{subj} \odot \overrightarrow{verb}) + (\overrightarrow{subj^*} \odot \overrightarrow{verb})$
Additive $\odot$	$(\overrightarrow{subj} + \overrightarrow{verb}) \odot (\overrightarrow{subj^*} + \overrightarrow{verb})$
Additive $+$	$\overrightarrow{subj} + \overrightarrow{verb} + \overrightarrow{subj^*} + \overrightarrow{verb}$
Kronecker $\odot$	$\overrightarrow{subj}^\top \times (\overrightarrow{verb} \otimes \overrightarrow{verb}) \odot \overrightarrow{subj^*}^\top \times (\overrightarrow{verb} \otimes \overrightarrow{verb})$
Kronecker $+$	$\overrightarrow{subj}^\top \times (\overrightarrow{verb} \otimes \overrightarrow{verb}) + \overrightarrow{subj^*}^\top \times (\overrightarrow{verb} \otimes \overrightarrow{verb})$
Categorical $\odot$	$\overrightarrow{subj}^\top \times \overline{verb} \odot \overrightarrow{subj^*}^\top \times \overline{verb}$
Categorical $+$	$\overrightarrow{subj}^\top \times \overline{verb} + \overrightarrow{subj^*}^\top \times \overline{verb}$

TABLE 4.5: Composition models for intransitive elliptical sentences.

by a difference of 0.10. However, for all models except the fully additive model, we see that adding the extra subject increases the difference between similarity scores, thereby making it easier to distinguish the correct interpretation. The most discriminative model is the fully multiplicative one, which treats the conjunctive coordinator as multiplication.

For the transitive case, we compare the sentences "the man draws the sword" and "the man draws the picture" with their landmark interpretations in which the verb 'draw' is replaced by either 'pull' or 'depict'. All of these are extended with the contexts 'warrior' and 'painter', and we compute the result of four of the mixed transitive models outlined above for the elliptical case: two are the same additive models that just sum all the vectors in a (sub)clause and either sum or multiply vectors for the subclauses for the elliptical variant, and two models use the Kronecker representation detailed above. The concrete cosine similarity scores are displayed in Table 4.7.

In this case, neither of the additive models seem to be effective: for the original phrases they

	Multiplicative $\odot$		Multiplicative $+$		Additive $\odot$		Additive $+$	
	race	**stand**	**race**	**stand**	**race**	**stand**	**race**	**stand**
man run	.88	.78	.88	.78	.94	.92	.94	.92
man run, governor does too	**.47**	**.99**	.80	.94	.82	.89	.95	.93
man run, athlete does too	**.99**	**.36**	.96	.71	.94	.71	.95	.92

TABLE 4.6: Cosine similarity scores between representations involving the intransitive verb 'run'. Column **race**: the representation of the corresponding row sentence but with 'race' instead of 'run', similarly for **stand**.

	Kronecker $\odot$		Kronecker $+$		Additive $\odot$		Additive $+$	
	pull	**depict**	**pull**	**depict**	**pull**	**depict**	**pull**	**depict**
man draw sword	.83	.50	.83	.50	.96	.93	.96	.93
man draw sword, warrior does too	**.98**	**.07**	.92	.44	.94	.76	.96	.93
man draw sword, painter does too	**.37**	**.28**	.69	.59	.89	.80	.96	.93
man draw picture	.82	.74	.82	.74	.97	.95	.97	.95
man draw picture, warrior does too	**.98**	**.25**	.91	.65	.92	.97	.97	.95
man draw picture, painter does too	**.37**	**.95**	.68	.88	.96	.98	.97	.96

TABLE 4.7: Cosine similarity scores between sentence representations using several models. Column **pull**: the representation of the corresponding row sentence but with 'pull' instead of 'draw', similarly for **depict**.

already give very high similarity scores, and those do not change greatly after adding the extra context. For the Kronecker models, we see that the best discriminatory model is the one that multiplies the vectors for the subclauses. For both the examples with "draw picture" and "draw sword", the interpretation of the verb as 'pull' is deemed more likely than 'depict' under the Kronecker model on the original transitive sentences. After adding the extra subject, the Kronecker model clearly improves the disambiguation result. For the first phrase, where a sword is drawn, the addition of 'warrior' greatly improves the similarity with 'pull' and accordingly decreases the similarity with 'depict', though for the addition of 'painter' this is not the case. The representation for 'painter' is in itself already closer to that of 'depict' (cosine similarity of 0.97) than it is to that of 'pull' (cosine similarity of 0.52), so adding 'painter' to the sentence makes it harder to be certain about 'pull' as a likely interpretation of 'draw'. We see in fact that the difference between the two sentence interpretations has become smaller.

For the second phrase, in which a picture is drawn, the original ambiguity is bigger, but adding the context provides us with the appropriate disambiguating scores. As with the first phrase, we also experience the difficulty in disambiguation: a human may deem "man draw picture, warrior does too" to be more similar to "man depict picture, warrior does too" since pulling a picture is not a very sensible action. However, because the vector for 'warrior' is closer to that for 'pull' (cosine similarity of 0.94) than it is to 'depict' (cosine similarity of 0.31) the model will favour the interpretation in which the picture is pulled.

4.2.1 Dataset descriptions

In order to create large scale datasets, we concretely extend the verb disambiguation datasets **ML2008** and **GS2011**, as well as the sentence similarity dataset **KS2014**.

First dataset: MLELLDIS

For the first dataset, we took the **ML2008** verb disambiguation dataset, and extended it as follows: for a given subject/verb combination and its two interpretations, we chose a new subject among the 100 most frequently occurring verb subjects in a corpus[1], but picked one that was significantly more frequent with the more likely unambiguous verb (the one marked HIGH). For example, the word "economy" occurs with "boom" but it occurs significantly more often with "prosper" then it does with "thunder". And similarly, "cannon" occurs with "boom" and "thunder" but not so often with "prosper". We then format the pairs from the **ML2008** dataset using the new subject and the elliptical setting. For the examples above, we then got

Landmark	export boom and economy does too	gun boom and cannon does too
HIGH	export prosper and economy does too	gun thunder and cannon does too
LOW	export thunder and economy does too	gun prosper and cannon does too

In total, we added two new subjects to each sentence pair, producing a dataset of 240 entries. We used the human similarity judgments of the original **ML2008** dataset to see whether the addition of disambiguating context, combined with our ellipsis model, will be able to better distinguish verb meaning. We did not re-annotate this dataset with new similarity judgments, but rather provide the dataset as an initial study for investigating verb phrase elliptical sentence embeddings. Since we reuse the original annotations of **ML2008**, computing the inter-annotator agreement for this dataset is not very sensible: it would simply be the same as for the original dataset since each original sentence pair will now have two elliptical entries with the same annotation.

Second dataset: ELLDIS

The second dataset is based on the transitive verb disambiguation dataset **GS2011**, but follows the same process as for the intransitive case above. For a given subject-verb-object combination and two landmark verbs that give a possible interpretation of the original verb, we selected a new subject from the list of most frequent subjects of the landmark verb[2] that was the more likely interpretation (HIGH) of the original, ensuring that it is was significantly more frequent than for the alternative, unlikely (LOW) landmark verb. By doing so we strengthened the disambiguating effect of the context for each verb. The final check was for the selected subject to make sense in the resulting elliptical phrase. For each combination and new subject considered, we added two elliptical sentence pairs, one for each of the two landmark verbs. For example, for the verb triple (*draw, depict, attract*), and original sentence pairs

man draw sword vs. *man depict sword*
man draw sword vs. *man attract sword*

we selected the new subject *artist* and added the two pairs below:

man draw sword and artist does too vs. *man depict sword and artist does too*
man draw sword and artist does too vs. *man attract sword and artist does too*

[1] In our case, this was the combined UKWaC and Wackypedia corpus, availabe at wacky.sslmit.unibo.it

[2] As found in the combined ukWaC+WackyPedia corpus.

We selected two new subjects for each combination, and in this way we obtained a dataset of roughly 400 entries. New human judgments were collected through Amazon Mechanical Turk, by prepending *the* to each noun and putting the phrase in the past tense. As with the original dataset, participants were asked to judge the similarity between sentence pairs using a discrete number between 1 and 7; 1 for highly dissimilar, 7 for highly similar. By inserting gold standard pairs of identical sentences we checked if participants were trustworthy. We collected 25 judgments per sentence pair but excluded participants that annotated less than 20 entries of the total dataset. We ended up with 55 different participants who ranked more than 20 entries of the total dataset, to give a final amount of ca. 9200 annotations. As an example, the verb *show* was a very hard case to disambiguate in the **GS2011** dataset: *child show sign* had an average score of 2.5 with both *child picture sign* and *child express sign*. In the new dataset, with the extra subject *patient*, it got much clearer that the verb had to be interpreted as *express* with an average score of 5.869, versus 4.875 for *picture*. The inter-annotator agreement in 1 vs. all format is 0.584 for the **ELLDIS** dataset. This is significantly lower than for the **GS2011** dataset, showing that there is in fact less agreement than for **GS2011**. This would intuitively contradict our intuition that adding more context makes it easier to disambiguate the verb, but a proper comparison between the two datasets is impossible in the current situation as the annotators for the **GS2011** dataset were most likely not the same as for the **ELLDIS** dataset. Moreover, due to the size of our dataset we could not present the whole dataset in one go to the annotators, so we rather split it into batches of ca. 100 entries. It was then up to the annotators to choose how many items they would annotate, effectively making any inter-annotator agreement score be more unreliable than for smaller datasets.

Third dataset: ELLSIM

Our third dataset differs from the two above, in that it does not test verb disambiguation, but general sentence similarity. It is an extension of the **KS2014** dataset described above. We extend this dataset to cover verb phrase ellipsis by following a similar procedure as for **GS2011**. For each subject-verb-object transitive sentence in the dataset, we selected a new subject s^* from a list of most frequent subjects of the verb[3] and built elliptical entries in such a way that the meaning of the original transitive sentence got changed as little as possible and that the resulting elliptical phrase made sense. We then considered every transitive sentence pair in the dataset and added the new respective subjects to both sentences. For example, for the pair

school encourage child vs. *employee leave company*

we selected *parent* and *student* to get the new pair

school encourage child and parent does too
vs.
employee leave company and student does too

We chose two subjects for every original sentence, generating four possibilities for each sentence pair, and a new dataset of 432 entries. This dataset was also annotated using Amazon Mechanical Turk, after putting each verb in the past tense and prepending *the* to each noun in the dataset. Gold standard pairs of identical sentences were inserted to validate trustworthiness of participants. The final dataset contains ca. 9800 annotations by 42 different

[3] Again taken from the UKWaC+Wackypedia corpus.

participants. Interestingly, we obtain a very low inter-annotator agreement using the 1 vs. all scheme discussed above: for **ELLSIM** this score is 0.425. Again, we can't truly compare this score to the IAA score for **KS2014**, as the annotators were different, and the size of the dataset forces us to present the dataset in batches to the annotators. As an effect of the batching we obtain less overlap between the judgments given by any two individual annotators.

4.3 Composition Models for Verb Phrase Elliptical Sentence Embeddings

We now turn to describe concrete composition models for the verb phrase ellipsis tasks that we described above, as well as detailing the vector spaces that we built to experiment with them.

4.3.1 Embedding Verb Phrase Elliptical Sentences

In order to experiment with verb phrase elliptical sentences, we adhere to a three-way classification of concrete composition mechanism for creating sentence embeddings. First, there are the simple algebraic models, which take all word representations to be vectors and add or multiply these word vectors to get a single sentence vector. The work of Mitchell and Lapata [ML08; ML10] experiments with these models. Second, there are tensor-based models, which differ in that they represent complex words as tensor of different orders: for example, Baroni and Zamparelli [BZ10] represent adjectives as matrices which, applied to a word vector produce a vector representation of the compound adjective-noun combination. The account of Coecke, Sadrzadeh, and Clark [CSC10] and others [CGS13; Cla15] generalises this to higher-order tensors, e.g. cubes for transitive verbs and hypercubes for ditransitive verbs. The benefit of a type-driven approach over the simple models is that they respect the grammatical structure of sentences: the meaning of "man bites dog" is distinct from that of "dog bites man" whereas in an additive/multiplicative model they would be identical. Moreover, the use of higher order tensors allows one to encapsulate the multifaceted information that complex words such as verbs contain, in a single representation. That is, a verb representation now allows the vectors for its associated subjects and/or objects to be *updated in context*. The trade-off is that the tensors themselves have to be learnt; where Baroni and Zamparelli [BZ10] apply regression learning to learn the content of adjective matrices, for transitive verbs there have been several approaches using multi-step regression learning [Gre+13], relational learning [GS11a], or a combination of co-occurrence information with machine learning techniques [PFC14; PRC14; FPC15]. In the next chapter we dive deeper into the problem of representation learning in a tensor-based setting. The third class of compositional models, which we refer to as *neural composition*, again takes the individual word meanings to be represented as vectors, but instead of presupposing an algebraic formula that ought to capture the sentence meaning as a vector, this operation is trained by passing through the word vectors through a neural network that optimised for a particular task. Examples are Skip-Thought Vectors [Kir+15], the Distributed Bag of Words model [LM14], InferSent [Con+17], and Universal Sentence Encoder [Cer+18], all discussed in Chapter 1 of this thesis. Where Skip-Thoughts lift the skipgram model for word vectors to the level of sentences, where the optimisation goal is to predict what is the correct next sentence embedding, given a constructed embedding for a current sentence, InferSent and Universal

Sentence Encoder are rather trained by fine-tuning a constructed sentence embedding such that it performs well on a number of natural language inference tasks.

To model verb phrase elliptical sentences, we have a number of additional composition choices at our disposal. For the experiments with the three ellipsis datasets MLELLDIS, ELLDIS, and ELLSIM we have two main goals in mind: primarily, we want to verify that resolving ellipsis contributes to the performance of a compositional model. For this purpose we contrast non-linear models, i.e. models that resolve the ellipsis (and thus use the verb and object resources twice) with linear models, which do not resolve the ellipsis (and thus only use the verb and object once). Our second goal is to investigate whether amongst the models that resolve the ellipsis, the ones that did so in a tensor-based way, i.e. using tensors instead of vectors to represent the verbs, perform better than additive and multiplicative models, and how these compare to neural composition models. Hence, for the non-neural methods we consider four classes of models: first we have linear vector models and non-linear vector models, which constitute two classes of baseline models that do or do not resolve ellipsis as part of their modelling. Second, we have tensor-based models, which derive from the lambda-based modelling of section 3.5.5, and linear Frobenius models, given by the Frobenius style modelling of section 3.5.4. Although the latter Frobenius style models of ellipsis produce theoretically unwanted results (as discussed in Chapter 3), this is in the case of complex ambiguous sentences, whereas our datasets don't contain such cases. Therefore, we include the results of such models too.

Linear Vector Models: In the first class of composition models every resource is used exactly once, following the pattern $\overrightarrow{w_1} \star \overrightarrow{w_2} ... \star \overrightarrow{w_n}$ for any sequence of words $w_1 w_2 ... w_n$. For an elliptical phrase "*subj verb obj and subj* does too*" it will thus compute the vector

$$\overrightarrow{subj} \star \overrightarrow{verb} \star \overrightarrow{obj} \star \overrightarrow{and} \star \overrightarrow{subj^*} \star \overrightarrow{does} \star \overrightarrow{too}$$

where $\star$ denotes either addition or multiplication. This is a natural extension of the models **ADD** and **MULT** above.

Non-Linear Vector Models: We contrast the linear vector models with their non-linear counterpart. Here, the assumption is that ellipsis is resolved but models do not respect word order, i.e. the models for the meaning of the subclause is still additive or multiplicative. The meaning of "*subj verb obj and subj* does too*" now is

$$\overrightarrow{subj} \star \overrightarrow{verb} \star \overrightarrow{obj} \nabla \overrightarrow{subj^*} \star \overrightarrow{verb} \star \overrightarrow{obj}$$

where both $\star$ and ∇ are placeholders that denote either addition of multiplication. We differentiate between $\star$ and ∇ because we assume that the combination of the two subclauses may be modelled differently from the composition inside the subclauses themselves. Hence, this leads to four different models, given by the choice of addition or multiplication for $\star$ and ∇. For example, we may choose $\star = +, \nabla = \odot$ to get the concrete sentence meaning below:

$$\overrightarrow{subj} + \overrightarrow{verb} + \overrightarrow{obj} \odot \overrightarrow{subj^*} + \overrightarrow{verb} + \overrightarrow{obj}$$

Tensor-Based Models: As shown in Chapter 3, vectors, tensors, and their basic operations can be *emulated* using a lambda calculus with constants for the relevant operations, following Muskens and Sadrzadeh [MS16]. The derivational mechanism produces an abstract lambda

term for a verb phrase elliptical sentence, which then is substituted with a composition operation of choice. In Chapter 3 we already gave different instantiations of such models to get a tensor-based, a multiplicative-additive, and a Kronecker model.

For a verb phrase elliptical phrase like "Dogs chase cats and children do too" we obtain the following abstract lambda term:

$$\mathit{chase}(\mathit{cats}, \mathit{dogs}) \wedge \mathit{chase}(\mathit{children}, \mathit{dogs})$$

and we will generally replace the nouns by simple vectors, leaving only the meaning of the verb and the coordination ($\wedge$) variable. The concrete models for transitive sentences that were evaluated by Milajevs et al. [Mil+14] can all be derived by varying the verb instantiation. For example, we can obtain the Copy Object (**CO**), Frobenius Additive (**FA**), Frobenius Multiplicative (**FM**) and Frobenius Outer (**FO**) instantiations of the verb:

$$\begin{aligned}
\textbf{CO}: &\quad \lambda os.o \odot (\textbf{verb} \times^{\top} s) \\
\textbf{FA}: &\quad \lambda os.s \odot (\textbf{verb} \times o) + o \odot (\textbf{verb} \times^{\top} s) \\
\textbf{FM}: &\quad \lambda os.s \odot (\textbf{verb} \times o) \odot o \odot (\textbf{verb} \times^{\top} s) \\
\textbf{FO}: &\quad \lambda os.s \odot (\textbf{verb} \times o) \otimes o \odot (\textbf{verb} \times^{\top} s)
\end{aligned}$$

The vector semantics of the extensions of transitive sentences with verb phrase ellipsis are obtained by taking each of the above as the semantics of each conjunct of the lambda logical form and interpreting the conjunction constant $\wedge$ as either sum or multiplication. We denote the effect of a transitive models from Table 4.8 with $T(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj})$ for some verb representation of the verb and vector representations of the subject and object. For example, for some verb representation $\widetilde{verb}$ (this will refer to the Kronecker model in the section below), we will write

$$\textbf{CO} \stackrel{\sim}{\cdot} (\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj}) = \overrightarrow{obj} \odot (\widetilde{verb} \times^{\top} \overrightarrow{subj})$$

The tensor-based models are assumed to resolve ellipsis and, given a transitive sentence model T from Table 4.8, the tensor-based meaning of "*subj verb obj and subj* does too*" is

$$T(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj}) \nabla T(\overrightarrow{subj^*}, \overline{verb}, \overrightarrow{obj})$$

where ∇ interprets the conjunction of the two subclauses. For the verb matrix we used either the relational verb or the Kronecker verb, and for ∇ we tried both addition and multiplication. As an ablation study, we also include results for a model which simply adds or multiplies the second subject, as it may be that just this new subject is informative enough to model the full sentence properly.

Linear Frobenius Models: We use the linear Frobenius models, that are tensor-based but follow the theoretical categorical model that we developed in Chapter 3 (and that are also discussed by Kartsaklis, Purver, and Sadrzadeh [KPS16]). In such models, the verb phrase is not copied but is rather applied to a mixture of the two subjects in the sentence. Concretely, for a sentence "*subj verb obj and subj* does too*", rather than having two complete subclauses that we mix in together using a transitive model T, we instead have a general formula

$$T(\overrightarrow{subj} \odot \overrightarrow{subj^*}, \overline{verb}, \overrightarrow{obj})$$

Name	Abbreviation	Formula
Verb Only	**VO**	$\overrightarrow{verb}/\overline{verb}$
Additive	**ADD**	$\overrightarrow{subj} + \overrightarrow{verb} + \overrightarrow{obj}$
Multiplicative	**MULT**	$\overrightarrow{subj} + \overrightarrow{verb} + \overrightarrow{obj}$
Relational	**RE** $^{-}$	$\overline{verb} \odot (\overrightarrow{subj} \otimes \overrightarrow{obj})$
Kronecker	**RE** $^{\sim}$	$(\overrightarrow{verb} \otimes \overrightarrow{verb}) \odot (\overrightarrow{subj} \otimes \overrightarrow{obj})$
Copy Subject	**CS**	$\overrightarrow{subj} \odot (\overline{verb} \times \overrightarrow{obj})$
Copy Object	**CO**	$\overrightarrow{obj} \odot (\overline{verb}^T \times \overrightarrow{subj})$
Frobenius Additive	**FA**	$\mathbf{CS}(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj}) + \mathbf{CO}(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj})$
Frobenius Multiplicative	**FM**	$\mathbf{CS}(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj}) \odot \mathbf{CO}(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj})$
Frobenius Outer	**FO**	$\mathbf{CS}(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj}) \otimes \mathbf{CO}(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj})$

TABLE 4.8: Table of all compositional models that were evaluated by Milajevs et al. [Mil+14].

Sentence Encoder Models: To compare the mentioned compositional models with state of the art neural baselines, we carried out our experiments with a four types of holistic sentence encoders, that take arbitrary text as input and produce an embedding. To properly compare with the compositional models above, we gave three different inputs to the encoders: a baseline encoding (Base), a resolved encoding (Res), and an encoding without functional words (Abl), all as below:

Base: "*subj verb obj and subj* does too*"
Res: "*subj verb obj and subj* verb obj*"
Abl: "*subj verb obj subj**"

We used six concrete pretrained encoders, available online: 4800-dimensional embeddings from the Skip-Thought model[4], 300-dimensional embeddings from two Doc2Vec implementations [LB16][5], 4096-dimensional embeddings from two InferSent encoders[6], and 512-dimensional embeddings from Universal Sentence Encoder[7].

Contextualised Embeddings: We also include results of a comparison with contextualised representation models ELMo and BERT. We classify them separately from the sentence encoders above, as they are in between word and sentence embeddings. Rather than a single sentence vector, ELMo and BERT models return vectors for each word in a sentence whose values *depend* on all words occurring in the sentence. For this, we use a pretrained ELMo model provided by Google[8], and two pretrained BERT models available on Github[9]. Rather than getting a single vector for a sentence, these models return a vector for each word in the

[4] github.com/ryankiros/skip-thoughts
[5] github.com/jhlau/doc2vec
[6] github.com/facebookresearch/InferSent
[7] tfhub.dev/google/universal-sentence-encoder
[8] tfhub.dev/google/elmo/2
[9] github.com/imgarylai/bert-embedding

sentence. We use the same models as for the sentence encoders above (**Base**, **Res**, **Abl**) and take the mean of the word vectors to compute a sentence vector.

Name	Abbreviation	Formula
Verb Only	**VO**	$\overrightarrow{verb}/\overline{verb}$
Additive	**ADD**	$\overrightarrow{subj} + \overrightarrow{verb} + \overrightarrow{obj} + \overrightarrow{and} + \overrightarrow{subj^*} + \overrightarrow{does} + \overrightarrow{too}$
Multiplicative	**MULT**	$\overrightarrow{subj} \odot \overrightarrow{verb} \odot \overrightarrow{obj} \odot \overrightarrow{and} \odot \overrightarrow{subj^*} \odot \overrightarrow{does} \odot \overrightarrow{too}$
Additive Non-Linear	**ADDNL**	$\overrightarrow{subj} + \overrightarrow{verb} + \overrightarrow{obj} + \overrightarrow{subj^*} + \overrightarrow{verb} + \overrightarrow{obj}$
Multiplicative Non-Linear	**MULTNL**	$\overrightarrow{subj} \odot \overrightarrow{verb} \odot \overrightarrow{obj} \odot \overrightarrow{subj^*} \odot \overrightarrow{verb} \odot \overrightarrow{obj}$
Tensor-Based	**TEN**	$T(\overrightarrow{subj}, \overline{verb}, \overrightarrow{obj}) \star T(\overrightarrow{subj^*}, \overline{verb}, \overrightarrow{obj})$
Frobenius	**FROB**	$T(\overrightarrow{subj} \star \overrightarrow{subj^*}, \overline{verb}, \overrightarrow{obj})$
Sentence Encoder Base	**SEB**	$E(subj\ verb\ obj\ and\ subj^*\ does\ too)$
Sentence Encoder Resolved	**SER**	$E(subj\ verb\ obj\ and\ subj^*\ verb\ obj)$
Sentence Encoder Ablate	**SEA**	$E(subj\ verb\ obj\ subj^*)$
Contextualised Base	**CEB**	$\texttt{mean}(E(subj\ verb\ obj\ and\ subj^*\ does\ too))$
Contextualised Resolved	**CER**	$\texttt{mean}(E(subj\ verb\ obj\ and\ subj^*\ verb\ obj))$
Contextualised Ablate	**CEA**	$\texttt{mean}(E(subj\ verb\ obj\ subj^*))$

TABLE 4.9: Composition models for transitive elliptical sentences that we evaluated.

All the concrete models that we are evaluate are summarised in Table 4.9.

4.3.2 Training Vectors and Tensors

As an input to the composition models, we train multiple vector spaces and verb matrices, where the choice of learning technique drastically changes the word representation and the way they ought to be composed into a sentence vector. Thus, in order to provide a comprehensive study with robust results, we use four different vector spaces: a count based vector space, and newly trained Word2Vec, GloVe, and FastText spaces, as detailed below. None of the spaces is taken off the shelf, but rather they are all trained on the same corpus, which is the combined UKWaC and Wackypedia corpus[10].

Count-Based: We used the combined ukWaC and Wackypedia corpora[11] to extract raw co-occurrence counts, using as a basis the 2000 most frequently occurring tokens (after excluding the 50 most frequent ones). When extracting counts, we disregarded a list of stopwords that do not contribute to the content of the vectors. We used a context window of 5 around the focus word, and PPMI as weighting scheme. These settings were use in the original **KS2013** dataset [KS13].

[10] wacky.sslmit.unibo.it
[11] wacky.sslmit.unibo.it

Word2Vec: The Word2Vec embeddings we used were trained with the continuous bag of words model of [Mik+13] (CBOW). We trained this model on the combined and lemmatised ukWaC and Wackypedia corpora, using the implementation for Python available in the `gensim` package[12], with a minimum word frequency of 50, a window of 5, dimensionality 300, and 5 training iterations.

GloVe: The GloVe model [PSM14] considers the *ratio* of co-occurrence probabilities by minimising the least-squares objective between the dot product of two word embeddings and the log-probability of the words' co-occurrence. We trained a GloVe space on the combined and lemmatised ukWaC and Wackypedia corpora, using the code provided by the original authors[13]. Similar to the Word2Vec settings above, we trained 300 dimensional vectors with a minimum word frequency of 50 and a window of 5, but we trained with 15 iterations.

FastText: The FastText vectors are like Word2Vec, except the word vector takes into account subword information: words are represented as n-grams, for which vectors are trained. The final word vector will then be the sum of its constituent n-gram vectors [Boj+17]. We trained a FastText space with the same settings as the Word2Vec space (CBOW, minimum word frequency 50, dimensions 300, window 5, with 5 iterations), again using `gensim`.

Verb Matrices In order to work with tensor-based models we had to represent verbs as matrices rather than as vectors. We generated verb tensors using two methods that have been used previously in the literature [GS11a; KS14].

Relational: For each verb, its corresponding matrix is obtained by summing over the tensor product of the respective subject and object vectors of the verb (subjects and objects collected from the corpus):

$$\overline{verb} = \sum_i subj_i \otimes obj_i$$

Kronecker: For each verb, its corresponding matrix is obtained by taking the tensor product of the verb vector with itself:

$$\widetilde{verb} = \overrightarrow{verb} \otimes \overrightarrow{verb}$$

In the case of the count based space, we trained verb matrices of dimensions 2000×2000, for the neural word embeddings the matrices had dimensions 300×300. The reason to choose a higher dimension for the count based embeddings is that they are generally sparser than neural word embeddings, hence it is fairer to compare higher-dimensional count based vectors and tensors with the denser neural embeddings.

4.4 Evaluation Results and Analysis

We now discuss the results of our embeddings for verb phrase elliptical sentences as per the datasets that we introduced. First, we validate the models that we described above on word and sentence tasks, before moving on to the results for the ellipsis datasets.

Results for word similarity First of all we validate the quality of the trained word spaces by evaluating on a number of the word similarity tasks described in Section 4.1. The results

[12] `radimrehurek.com/gensim`
[13] `nlp.stanford.edu/projects/glove`

are displayed in Table 4.10, for the spaces described in the previous section. These serve as a minimal baseline: the word embeddings we use ought to obtain a reasonably high performance on these datasets at the very latest, since we are interested mainly in the composition of high quality word embeddings. The results in Table 4.10 show that indeed the spaces that

	RG	**WS353**	**MC**	**SL999**	**MEN**
Count	0.608	0.358	0.546	0.259	0.553
Word2Vec	0.823	**0.698**	**0.768**	**0.403**	**0.781**
GloVe	**0.831**	0.618	0.738	0.390	0.773
FastText	0.772	0.546	0.696	0.402	0.768
SoTA	0.769	0.706	0.832	0.406	0.798

TABLE 4.10: Spearman ρ scores on word similarity tasks. SoTA: State of the art scores using GloVe vectors as reported by Dobó and Csirik [DC19].

we train are of high quality compared to reported state of the art scores with similar vector space models. Here, we give the state of the art score for the GloVe vectors of Pennington, Socher, and Manning [PSM14] as reported by Dobó and Csirik [DC19]. at The word2vec space achieves the highest performance on all datasets except the **RG** dataset. Note that the lowest performance is on the SimLex-999 dataset, which is also the most challenging word embedding dataset discussed here, with a reported state of the art correlation score of 0.406 for GloVe vectors (though other, more complex models are able to achieve a correlation as high as 0.76)[14].

Results for intransitive sentence tasks Next, we look at how well the trained models perform on the **ML2008** and **ML2010** datasets, containing intransitive sentences of the form *subj verb* and *verb object*. Table 4.11 shows the Spearman ρ correlation scores on the verb disambiguation dataset **ML2008**, whereas the results for the sentence similarity task **ML2010** are presented in Table 4.12. In each table, the top two rows show the results for a non-compositional verb only baseline, either by considering the verb vector or the verb tensor, followed by arithmetic composition and order-sensitive composition models. For each different vector training method, the highest score is highlighted in bold.

ML2008	Count Based	Word2Vec	GloVe	FastText
Verb Only Vector	0.078	0.153	0.329	0.095
Verb Only Tensor	0.067	-0.035	0.033	0.036
Additive	0.081	**0.220**	0.293	0.116
Multiplicative	**0.177**	0.199	0.100	**0.189**
Kronecker	0.076	0.165	**0.379**	0.067
Categorical	0.045	0.121	0.103	0.163

TABLE 4.11: Spearman ρ correlation scores on the **ML2008** dataset.

[14] See fh295.github.io/simlex.html

ML2010	Count Based	Word2Vec	GloVe	FastText
Verb Only Vector	0.498	0.585	0.448	0.653
Verb Only Tensor	0.411	0.605	0.506	0.564
Additive	0.613	**0.646**	**0.597**	**0.670**
Multiplicative	**0.645**	0.516	0.344	0.643
Kronecker	0.498	0.589	0.449	0.630
Categorical	0.369	0.488	0.422	0.493

TABLE 4.12: Spearman ρ correlation scores on the **ML2010** dataset.

In the case of the **ML2008** and **ML2010** datasets, the results show that moving from an order-insensitive arithmetic model to a grammatically informed tensor-based model generally decreases performance. For **ML2008**, we attribute this to the fact that the tensor representations in themselves (**Verb Only Tensor**) already show a lower correlation compared to the verb vectors on the verb-only part of the dataset, which will then permeates the results for compositional models. Moreover, there is only one word available as the context of the verb, making it harder for compositional models to beat the simpler models. For **ML2010**, the tensors in themselves work quite well, but categorical composition underperforms the additive and multiplicative baseline. Again, here there is only one word available as context to the verb, possibly defeating the purpose of using a complex composition model.

Results for transitive sentence tasks The scores of our models on the **GS2011**, **KS2013**, and **KS2014** datasets are in Tables 4.13, 4.14, and 4.15 respectively. In each table, the highest score per vector space is highlighted in bold, and to be succinct we report only the two highest scoring tensor-based models, with the model specification beside the correlation score as per the model descriptions in the previous section. The full results for all composition models are included in Appendix A.

GS2011	Count Based		Word2Vec		GloVe		FastText	
Verb Only Vector	0.273		0.210		**0.302**		0.212	
Verb Only Tensor	0.167		0.342		0.241		0.292	
Additive	0.291		0.255		0.248		0.140	
Multiplicative	0.431		0.204		0.217		0.193	
Best Tensor	0.454	**CO** $\widetilde{\cdot}$	**0.353**	**FA** $\bar{\cdot}$	0.252	**CO** $\bar{\cdot}$	0.353	**FO** $\bar{\cdot}$
2nd Best Tensor	0.447	**FM** $\widetilde{\cdot}$	0.343	**RE** $\bar{\cdot}$	0.295	**FA** $\bar{\cdot}$	0.339	**CO** $\bar{\cdot}$

TABLE 4.13: Spearman ρ correlation scores on the **GS2011** dataset, with the highest score for each space in bold and in a box, the second highest result in bold. Results are against an inter-annotator agreement score of 0.739.

For the **GS2011** dataset, the highest result is achieved for the **Frobenius Additive** model on the word2vec vectors, using the relational matrix. Across the board, here the tensor-based models outperform arithmetic baselines for all of the spaces. We argue that now, since there is more context available in each sentence, the use of tensor-based composition models starts to pay off, especially since the verb tensor representations were built by considering

KS2013	Count Based		Word2Vec		GloVe		FastText	
Verb Only Vector	0.108		0.199		0.132		0.112	
Verb Only Tensor	0.093		0.100		0.065		0.040	
Additive	0.104		0.210		0.174		0.117	
Multiplicative	0.279		0.334		0.110		0.302	
Best Tensor	0.322	**FM** $\widetilde{\cdot}$	**0.415**	**FA** $\widetilde{\cdot}$	0.140	**FA** $\widetilde{\cdot}$	0.368	**FA** $\widetilde{\cdot}$
2nd Best Tensor	0.258	**CO** $\widetilde{\cdot}$	**0.408**	**CS** $\widetilde{\cdot}$	0.123	**FO** $\widetilde{\cdot}$	0.334	**CS** $\widetilde{\cdot}$

TABLE 4.14: Spearman ρ correlation scores on the **KS2013** dataset, with the highest score for each space in bold and in a box, the second highest result in bold. Results are against an inter-annotator agreement score of 0.575.

both subjects and objects of the verb. In the case of **KS2013** we see a similar pattern for all spaces, except for GloVe vectors, where the additive baseline gives the highest result. On the last dataset we experiment with, **KS2014**, the conclusion is converse: the additive models give the highest result on all spaces but the count based space. That the neural word embedding models all show a similar pattern, opposing the count based model, is due to sparsity of count based vectors versus density of neural word embeddings: once we add together sparse vectors we end up with a more general vector that therefore is less specific, whereas multiplying sparse vectors actually produces a more specific vector. It must be noted, though, that the additive model score is above the interannotator agreement score, thereby exceeding the supposed upper bound for the dataset. On the other hand, the best tensor model is equidistant from the interannotator agreement, but from below.

KS2014	Count Based		Word2Vec		GloVe		FastText	
Verb Only Vector	0.521		0.665		0.535		0.705	
Verb Only Tensor	0.456		0.617		0.504		0.563	
Additive	0.677		**0.763**		0.719		0.764	
Multiplicative	0.719		0.528		0.283		0.587	
Best Tensor	**0.745**	**FM** $\widetilde{\cdot}$	0.623	**RE** $\widetilde{\cdot}$	0.427	**FO** $\widetilde{\cdot}$	0.641	**FO** $\widetilde{\cdot}$
2nd Best Tensor	0.739	**FO** $\widetilde{\cdot}$	0.578	**RE** $\widetilde{\cdot}$	0.418	**RE** $\widetilde{\cdot}$	0.637	**RE** $\widetilde{\cdot}$

TABLE 4.15: Spearman ρ correlation scores on the **KS2014** dataset, with the highest score for each space in bold and in a box, the second highest result in bold. Results are against an inter-annotator agreement score of 0.754.

Results for ellipsis datasets We now move on the present new results for the datasets we introduced for modelling verb phrase ellipsis. We would expect in the results that models that resolve ellipsis perform better at our tasks than models which don't, i.e. linear vector models are expected to have a lower score than non-linear vector models and the grammar-sensitive tensor-based models.

Verb Disambiguation 1 (MLELLDIS): For the elliptical extension of the **ML2008** verb disambiguation dataset, we find the correlation scores in Table 4.16. These results validate

MLELLDIS	Count Based	Word2Vec	GloVe	FastText
Verb Only Vector	0.078	0.153	0.329	0.095
Verb Only Tensor	0.067	-0.035	0.033	0.036
Additive	0.040	0.179	0.249	0.142
Multiplicative	**0.206**	0.081	0.003	0.103
Multiplicative $\odot$	**0.391**	0.085	0.017	0.124
Multiplicative $+$	0.179	**0.236**	0.055	0.156
Additive $\odot$	0.172	0.195	0.078	**0.201**
Additive $+$	0.078	**0.229**	**0.336**	0.170
Kronecker $\odot$	0.128	0.136	0.294	0.133
Kronecker $+$	0.076	0.207	**0.366**	0.085
Categorical $\odot$	0.060	0.143	0.008	0.190
Categorical $+$	0.047	0.177	0.092	**0.211**

TABLE 4.16: Spearman ρ correlation scores on the MLELLDS dataset, with the highest score for each space in bold and in a box, the second highest result in bold.

our intuition that models that resolve ellipsis (from **MultMult** to **CatSum**) can outperform the models that don't resolve it. As the highest results for the count based space and the word2vec space are for arithmetic models, but for GloVe and FastText they are obtained by a tensor-based model, this shows that at least resolving ellipsis is important for distributional semantics, but that order sensitivity is not a necessary ingredient to achieve high performance. Although the correlation score on MLELLDIS is improved for the count based space comparatively to the results of Table 4.11, this isn't true for all of the spaces. Moreover, since we reused the annotations of the **ML2008** dataset, these results may not be grounded in proper human similarity judgment as the original participants were not confronted with the verb phrase elliptical variations of the sentence pairs in the **ML2008** dataset.

Verb Disambiguation 2 (ELLDIS): Table 4.17 shows the results of the linear, non-linear and tensor-based models for this task, compared against a baseline in which only the verb vector or verb matrix is compared.

Our first observation is that generally, the highest performing models were tensor-based. The highest found correlation score was 0.539 in the count based space for a tensor-based model (**CO** $\widetilde{+}$: the Copy Object model, with addition for the coordinator 'and', using the Kronecker matrix), with the Frobenius Additive model giving the second best result of 0.526 (**FA** $\widetilde{+}$: the Frobenius Additive model, with addition for the coordinator 'and', using the Kronecker matrix). For the neural spaces, the highest performing models were mostly tensor-based as well; they were always the Frobenius Additive (**FA** $\overline{+}$) model and the Frobenius Outer (**FO** $\overline{+}$) model, using the relational tensor and addition for the coordinator, except in the case of GloVe, where the Copy Object Sum (**COS** $\overline{+}$: the additive variation on the Copy Object model, using addition for coordination, and the relational verb matrix) model was the second best. The only exception to this observation is the GloVe space, for which the baseline Vector Only model in fact has a higher correlation than any other model on that space.

Our second observation is that the non-linear variants of the additive and multiplicative

ELLDIS	Count Based		Word2Vec		GloVe		FastText	
Verb Only Vector	0.436		0.241		0.445		0.229	
Verb Only Tensor	0.330		0.438		0.394		0.388	
Add. Linear	0.442		0.273		0.305		0.141	
Mult. Linear	0.325		-0.012		0.182		0.293	
Add. Non-Linear	0.445		0.328		0.326		0.140	
Mult. Non-Linear	0.503		0.209		0.245		0.044	
Best Tensor	**0.539**	**CO** $\widetilde{+}$	0.462	**FA** $\overline{+}$	0.373	**COS** $\overline{+}$	0.494	**FO** $\overline{+}$
2nd Best Tensor	0.526	**FA** $\widetilde{+}$	0.454	**FO** $\overline{+}$	0.369	**FA** $\overline{+}$	0.465	**FA** $\overline{+}$
Best Frobenius	**0.539**	**CO** $\widetilde{\odot}$	0.462	**FA** $\overline{+}$	0.397	**FA** $\overline{\odot}$	0.497	**FO** $\overline{+}$
2nd Best Frobenius	**0.527**	**FO** $\widetilde{\odot}$	0.460	**FO** $\overline{+}$	0.369	**FA** $\overline{+}$	0.465	**FA** $\overline{+}$

TABLE 4.17: Spearman ρ scores for the ellipsis disambiguation experiment ELLDIS. We contrast verb only, additive and multiplicative baselines with the two best tensor-based models per space, as well as the two best linear frobenius models.

models (which resolve ellipsis but in a naive way) show an increased performance over the linear models (which do not resolve ellipsis). All of this holds for all the four vector spaces, except for the FastText space where the linear multiplicative model achieves significantly higher correlation (0.293) than its non-linear counterpart (0.044).

Overall, these results suggests that a logical resolving of ellipsis and further grammatical sensitivity benefits the performance of composition.

One interesting fact about our results is that the best compositional methods across the board were those that interpret the coordinator 'and' as addition; in set-theoretic semantics one interprets this coordinator as set intersection, which corresponds to multiplication rather than addition in a vectorial setting. We suggest that the feature intersection approach using multiplication leads to sparsity in the resulting vectorial representation, which then has a negative effect on the overall result. This would explain the case of FastText, since those vectors take into account subword information one would expect them to be more fine-grained and therefore conflate more of their features under multiplication. The choice of verb matrix was however mixed: for the count-based models the Kronecker matrix worked best, for the neural embeddings it was best to use the relational matrix.

Sentence Similarity (ELLSIM): For the extension of the **KS2014** sentence similarity dataset, the results are shown in Table 4.18. We again wanted to see if resolving ellipsis benefits the compositional process. This was in general true, although we observed a different pattern to the previous experiment.

In all cases, except for the FastText space, we saw that non-linear models in fact perform better than their linear counterparts. But this time the best tensor-based models only outperformed addition for the count-based space: the best models scored 0.7410 and 0.7370 (respectively for the **FO** and **FA** models above, Kronecker matrix, $\nabla = \odot$). Both Word2Vec and GloVe worked best with a non-linear additive model, with Word2Vec achieving the overall highest correlation score of 0.7617, and GloVe achieving 0.7103. For FastText, the highest

ELLSIM	Count Based		Word2Vec		GloVe		FastText	
Verb Only Vector	0.457		0.583		0.435		0.647	
Verb Only Tensor	0.395		0.566		0.443		0.534	
Add. Linear	0.700		0.726		0.696		0.741	
Mult. Linear	0.633		0.130		0.367		0.199	
Add. Non-Linear	0.681		**0.762**		0.710		0.739	
Mult. Non-Linear	0.723		0.355		0.244		0.450	
Best Tensor	**0.741**	**FO** $\widetilde{\odot}$	0.706	**RE** $\overline{+}$	0.491	**FA** $\overline{+}$	0.699	**FO** $\overline{+}$
2nd Best Tensor	0.737	**FA** $\widetilde{\odot}$	0.671	**FO** $\overline{+}$	0.482	**FO** $\overline{+}$	0.688	**CO** $\overline{\odot}$
Best Frobenius	0.732	**FM** $\widetilde{\odot}$	0.706	**RE** $\overline{+}$	0.491	**FA** $\overline{+}$	0.702	**FO** $\overline{+}$
2nd Best Frobenius	0.730	**FA** $\widetilde{\odot}$	0.668	**FO** $\overline{+}$	0.486	**FO** $\overline{+}$	0.682	**RE** $\overline{+}$

TABLE 4.18: Spearman ρ scores for the ellipsis similarity experiment. We contrast verb only, additive and multiplicative baselines with the two best tensor-based models per space, as well as the two best linear frobenius models.

score of 0.7408 was achieved by linear addition. What is more, the multiplicative model did not benefit from a non-linear approach in the case of GloVe (from 0.3666 to 0.2439), and the additive model had a similar decline in performance for the count-based space (from 0.7000 to 0.6808) and FastText (0.7408 to 0.7387). We can see that for the neural word embeddings the additive models work best, with all of them seeing a drop in performance for the tensor-based models. Again, the best count-based models use the Kronecker matrix whereas the neural models benefit the most from using the relational matrix. However, this time the best count-based models used multiplication for coordination, the neural models preferring addition.

Sentence Encoders and Contextualised Embeddings: We compare the results of the above compositional models with the results of modelling the datasets with sentence encoders and contextualised embeddings. In the former case, a sentence is simply passed through an encoder which returns a single vector. In the case of contextualised embeddings the system returns vectors for each word in a sentence; these are averaged to produce a final sentence vector. We give the results on all datasets the sentence encoders discussed above in Table 4.19, whereas the results for the contextualised embeddings from ELMo and BERT are in Table 4.20.

Comparison to ML2008-KS2014 Our first observation, comparing the results of sentence encoders and contextualised embeddings with our sentence experiment baselines (ML2008 through **KS2014**), is that none of the results outperform the best scoring compositional methods: on the **ML2008** dataset, the highest score of 0.220 is obtained by the 4096-dimensional InferSent encodings (**IS2**), whereas the mean ELMo vectors achieve a 0.166 correlation, compared to a correlation of 0.379 for the Kronecker model using the GloVe vectors and tensors in Table 4.11. For **ML2010**, the highest compositional score was 0.670, obtained by a purely additive model on the FastText vectors, which is not matched by the highest encoder score

	D2V1	D2V2	ST	IS1	IS2	IS3	IS4	USE
ML2008	0.139	0.192	0.078	0.181	**0.220**	0.149	0.169	0.039
ML2010	0.512	0.447	0.494	0.631	0.492	**0.636**	0.405	0.325
GS2011	0.098	0.102	-0.157	0.297	0.320	**0.324**	0.213	0.094
KS2013	0.193	**0.212**	0.051	0.172	0.032	0.176	-0.021	0.210
KS2014	0.692	0.705	0.546	**0.784**	0.676	0.720	0.586	0.539
MLELLDIS$_{lin}$	0.090	0.233	0.232	0.199	0.224	0.108	0.135	0.105
MLELLDIS$_{res}$	0.089	0.216	0.167	0.228	**0.269**	0.144	0.156	0.154
MLELLDIS$_{abl}$	0.095	0.242	0.159	0.215	0.226	0.141	0.169	0.109
ELLDIS$_{lin}$	0.199	0.227	-0.193	0.347	**0.384**	0.330	0.344	0.269
ELLDIS$_{res}$	0.231	0.253	-0.172	0.344	0.337	0.293	0.248	0.277
ELLDIS$_{abl}$	0.195	0.259	-0.130	0.353	0.357	0.300	0.291	0.240
ELLSIM$_{lin}$	0.593	0.622	0.585	0.779	0.701	0.748	0.641	0.647
ELLSIM$_{res}$	0.698	0.692	0.604	**0.803**	0.749	0.768	0.687	0.680
ELLSIM$_{abl}$	0.652	0.655	0.471	0.782	0.730	0.749	0.682	0.640

TABLE 4.19: Spearman ρ scores for sentence encoders on all datasets. D2V1: Doc2Vec1, D2V2: Doc2Vec 2, ST: Skip-Thought, IS1: InferSent 1 (4096), IS2: InferSent 2 (4096), IS3: InferSent 1 (300), IS4: InferSent 2 (300), USE: Universal Sentence Encoder.

of 0.636 (using 300-dimensional InferSent vectors) and the ELMo score of 0.539. For **GS2011** we see a similar pattern, with a 300-dimensional InferSent model scoring 0.324 versus the 0.292 achieved by the BERT Large model, both of which are surpassed by a tensor-based model on the word2vec space which achieves a correlation score of 0.353. On **KS2013** the best tensor-based model scored 0.415, versus a score of 0.212 from a Doc2Vec model, and 0.349 for BERT Large. Only on **KS2014** we see the highest correlation being achieved by a sentence encoder: where the highest performing compositional model is a purely additive model with a correlation of 0.763, and this still outperforms the ELMo vectors which achieve 0.728 correlation, they are all outperformed by a 4096-dimensional InferSent model which achieves a correlation score of 0.784.

Comparison to MLELLDIS, ELLDIS, and ELLSIM We now note the difference between the neural composition models and the tensor-based composition models. Firstly, on MLELLDIS, the tensor-based modelling achieved a maximal correlation of 0.391 in a count based model, which is in contrast with the results achieved by sentence encoders (0.269 for a 4096-dimensional InferSent model) and contextualised embeddings (0.373 by BERT Small). This may be due to the fact that the dataset has the same annotations as **ML2008**, hence its results carrying over from the original intransitive sentence dataset. However, as noted before, while the tensor-based modelling generally shows that resolving the ellipsis is beneficial to the performance of the composition models, this is a bit less obvious for the neural composition methods: Although the InferSent and Universal Sentence Encoder models do show an increased performance when the ellipsis is resolved, this doesn't hold for the Doc2Vec and Skipthoughts models.

	ELMo	BERT Small	BERT Large
ML2008	**0.166**	0.105	0.030
ML2010$_v$	**0.539**	0.216	0.356
GS2011	0.108	0.187	**0.292**
KS2013	0.243	0.232	**0.349**
KS2014	**0.728**	0.520	0.616
MLELLDIS$_{lin}$	0.193	**0.373**	0.315
MLELLDIS$_{res}$	0.182	0.103	0.342
MLELLDIS$_{abl}$	0.123	0.089	0.193
ELLDIS$_{lin}$	0.232	0.360	**0.368**
ELLDIS$_{res}$	0.210	0.216	0.274
ELLDIS$_{abl}$	0.207	0.197	0.365
ELLSIM$_{lin}$	0.734	0.595	0.580
ELLSIM$_{res}$	**0.779**	0.631	0.647
ELLSIM$_{abl}$	0.703	0.560	0.582

TABLE 4.20: Spearman ρ scores for contextualised embeddings on all datasets. While **ELMo** performs very well on the smaller datasets, it is outperformed by **BERT** on the elliptical datasets, although for the latter there is no improvement by moving to the **BERT Large** model.

For ELLDIS, the sentence encoder results of Table 4.19 show the same trend that suggests that resolving ellipsis improves the quality of the embeddings: with the exception of the InferSent encoders, the resolved models gave a higher correlation than their linear baseline. However, none of the encoder models come near the results achieved using the compositional models. Since the verb disambiguation dataset contains pairs of sentence that only differ in the verb, the task becomes very much grammar-oriented, and so we argue that the tensor-based models work better since they explicitly emphasise syntactic structure.

The sentence encoders worked a lot better in the ELLSIM similarity task, with all non-linear resolved models outperforming the baseline model, and the InferSent model even outperforming non-linear addition on a Word2Vec space. We argue this is the case for two reasons: first, the similarity dataset is more diffuse than the verb disambiguation dataset since sentence pairs now differ for every word in the sentence, giving more opportunity to exploit semantic similarity rather than syntactic similarity. Second, the embeddings from the sentence encoder are larger (4096), allowing them to effectively store more information to benefit the similarity score.

4.5 Conclusion

This chapter introduced the second main contribution of this thesis. We introduced three new datasets in order to evaluate the models we developed in Chapter 3 for the case of verb phrase ellipsis. Then, we trained four vector spaces using different training techniques, and compared the performance of a large variety of composition models on the datasets from previous work and the newly introduced ellipsis dataset. We specifically compared linear models that don't resolve ellipsis with models that do perform the ellipsis resolution step. Moreover, we contrasted the results of these type-driven composition models against state of the art sentence encoders, and contextualised embeddings.

First we noted how tensor-based compositional models underperformed additive baselines on the intransitive sentence datasets **ML2008** and **ML2010**, as only one word (respectively the subject or object of the verb) is available as context in these datasets. In the case of transitive sentence datasets **GS2011** and **KS2013** we saw that tensor-based models showed the highest correlation scores. We argued that in the setting of transitive sentences using the composition models with tensor representations benefit performance, and even more so because the verb representations are constructed by considering both subjects and objects of the verbs.

With the verb phrase ellipsis experiments, we primarily wanted to show that resolving ellipsis improves performance. Indeed, non-linear models almost always performed better than linear ones in both a verb disambiguation and a sentence similarity task. On a second note, the highest performance on the verb disambiguation task was given by a grammar-driven, tensor-based model in a count-based vector space, whereas for the similarity task, the highest performance was achieved by the InferSent sentence encoder, followed by a non-linear additive model on a Word2Vec space. Although the neural word embeddings and sentence encoders were largely outperformed on the disambiguation dataset that places more emphasis on syntactic structure than on semantic similarity, they generally performed better in the sentence similarity case, where the distinction between syntactic and semantic similarity is more diffuse. However, sentence encoders generally still performed better than the state of the art contextualised embedding models ELMo and BERT.

Although this does not give conclusive evidence for one particular model being better than another, we do get two main overall conclusions: first, performing an ellipsis resolution step is generally important when constructing vector representations of sentences. Second, in a disambiguation task it is better to use a tensor-based model that can encode the relational information of verbs, though the added benefit of such a model gets lost in a similarity task.

This conclusion, however, may depend on the particular choice of verb representation, which so far was either a combination of vectors for the subjects and objects of a verb, or given by the outer product of a verb vector with itself. In the next chapter, we dive deeper in the topic of representation learning, and show how we can adapt the (neural) skipgram technique for learning higher order tensor representations, which we then evaluate on the verb disambiguation and sentence similarity datasets discussed so far.

Chapter 5

Lexical Semantics: Neural Tensor Embeddings

Chapter Abstract

The experimental work in Chapter 4 introduced a number of new datasets for evaluating compositional distributional semantics. One core component of type-driven compositional distributional semantics, however, is the lexical representation of words in a tensorial form. This chapter contains the third major contribution of this thesis: we introduce a generalisation of both the skipgram model for noun vectors and the tensor-based skipgram model of Maillard and Clark [MC15], that allows us to train and test a variety of different verb representations. We show how a decomposed matrix representation for a transitive verb achieves the highest performance compared to vector and cube representations and compared to previously used methods. This model then paves the way for generalisation to neural tensor embeddings*, the main topic for future work of this thesis. The content of this chapter includes text from a to-be submitted paper [WSC19] (joint work with Mehrnoosh Sadrzadeh and Stephen Clark).*

Where Chapter 3 lays the theoretical foundation for reasoning about ellipsis and anaphora in a compositional distributional model of meaning, and Chapter 4 provides the experimental component to test these models, in this chapter we approach the modelling from the point of view of *lexical semantics*. An essential assumption of type-driven approaches to compositional distributional semantics, such as the categorial framework of Coecke, Sadrzadeh, and Clark [CSC10] and Coecke, Grefenstette, and Sadrzadeh [CGS13], the functional approximation approach of Baroni, Bernardi, and Zamparelli [BBZ14], and the CCG-tensor contraction approach of Maillard, Clark, and Grefenstette [MCG14], is that the shape of embeddings for words that have functional types, such as adjectives and verbs, should be higher order, reflecting the idea that they represent functions which modify their respective arguments. Therefore, we need to address the issue of learning such representations.

This is not a new problem, however. In previous work on evaluating compositional distributional models, multiple proposals were given and implemented. Already in the previous chapter on evaluation we mentioned two analytical approaches to obtaining the content of verb tensors, either by taking the outer product of a verb vector representation (the Kronecker model, [GS11a]) or by taking the sum of the outer products of the vectors for observed subjects and objects of a given transitive verb (the Relational model, [GS11a]). Besides from these approaches, a number of other approaches have been proposed to learn the contents of tensor representations.

For adjectives, matrices have been learned by regression to approximate holistic adjective-noun phrase vectors [BZ10], and more recently, by lifting the skipgram model of Mikolov et al. [Mik+13] to learn a transformation between fixed vectors for nouns and adjective-noun combinations [MC15].

For verbs, besides from the analytical representations there is the work of Grefenstette et al. [Gre+13], which lifts the regression learning method of Baroni and Zamparelli [BZ10] and applies a multi-step regression algorithm to learn a cube representation of a verb by first approximating a holistic subject-verb vector, and subsequently a holistic verb-object matrix. A different approach is the plausibility model of Polajnar, Rimell, and Clark [PRC14], based on the model of Clark [Cla13], in which a verb matrix or cube is learnt by optimising a model that distinguishes between observed subject-verb-object triples (plausible) and randomly generated unobserved triples (implausible).

A major issue that these representations suffer from, is data sparsity: for the multi-step regression approach of Grefenstette et al. [Gre+13] one looks first for subject-verb occurrences in a corpus, and then one needs all the subject-verb-object occurrences in the corpus. Given that such occurrences are very sparsely available, and that a cube representation will contain a lot of parameters over which to perform a regression optimisation algorithm, there is simply not enough data to train a qualitative tensor representation, leading to a then decent correlation score of 0.32 [Gre+13] on the **GS2011** dataset, which by now has been surpassed already by the analytical approaches that we apply in Chapter 4. For the plausibility model and other models developed by Polajnar [PRC14; PFC14; Pol16] the same problem seems to persist.

On the other hand, since previous work on tensor representation learning was done, there have been many advances in deep neural network approaches to learning sentence representations, albeit in a uniformly vectorial way. There are sentence encoder models and contextualised embeddings that were already discussed in the background section in Chapter 1, but there are also the tree-structured Recursive Neural Networks [Soc+13] and their improved stack and lifted versions, e.g. [CWB18], which learn a composition operator based on the structures of sentences. These models, however, still uniformly represent each word as a tensor of the same order, regardless of grammatical types. That is, either every word in a sentence is represented as a vector (most approaches), or every word is represented as a matrix (in [CWB18]).

To attempt to bridge the gap between tensor-based compositional distributional models on the one hand, that allow for a transparent explicit incorporation of syntactic information into semantic representations, and neural approaches like sentence encoders and contextualised embeddings, we address the topic of lexical tensor representations in this chapter. We generalise the skipgram model with negative sampling of Mikolov et al. [Mik+13] to learn arbitrary tensor representations for words that have complex (or functional) types, based on the dependency information that is available for them. In a nutshell, given a word with a number of dependencies, its tensor representation transforms the vectors of a certain number of its dependencies to predict the remaining dependencies as a context. For instance, a transitive verb with two dependencies – a subject and an object – has two tensor representations, one of which transforms its object vectors to predict its subject context vectors, and another that predicts the object context vectors from its subject vectors. This approach allows us to learn tensors of higher ranks, as well as consider different possibilities for the context of the result of applying that tensor. This effectively helps us address the sparsity issues that some of the above mentioned approaches have. Moreover, working with a tensor that has a rank lower than the one dictated by a strict type-driven approach provides us with

a tuple of representations that can be independently applied and then further merged when composing sentence embeddings[1].

We formulate our tensor skipgram model in full generality, show how it reduces to the regular vector skipgram model, as well as the adjective matrix skipgram model of Maillard and Clark [MC15]. To test this general formulation of a tensor-skipgram model, we instantiate our model to learn a set of different tensor representations for *transitive verbs*, as these form a major component of the datasets that we discuss in this thesis. We evaluate these verb tensors on verb similarity and compositional sentence similarity and disambiguation tasks and show that in nearly all cases, the newly introduced and learned neural verb matrix embeddings outperform all of the other models. We moreover include a comparison with sentence encoders (Doc2Vec [LM14], SkipThoughts [Kir+15], InferSent [Con+17], Universal Sentence Encoder [Cer+18]) and state of the art contextualised embeddings given by ELMo [Pet+18] and BERT [Dev+19].

This chapter is structured as follows: first, we highlight a number of the related approaches to tensor representation learning that we briefly discussed above, to then illustrate and formulate the general tensor-skipgram model, and our verb-specific training methods. We then describe the concrete composition models and experimental setup that we use to evaluate the verb representations, and give the results on all the compositional datasets that we also evaluated on in Chapter 4, to end up with future directions to learn arbitrary tensor representations.

5.1 Representing Words as Tensors

We highlight previous approaches to learning tensor representations, and display their strengths and advantages. As mentioned in the previous chapter on evaluation, one of the first to experiment with composition models of distributional word representations were Mitchell & Lapata, who experimented with a number of algebraic models of composition [ML08; ML10]. However, the tensor-based approach gained attraction around the same time, with the work of Coecke, Sadrzadeh, and Clark [CSC10] and Baroni and Zamparelli [BZ10], in which functional words are represented not as vectors, but as higher order tensor representations.

Analytical Representations As an initial approach to representing verbs as tensors, Grefenstette and Sadrzadeh [GS11a] introduce two analytical methods (i.e. not by optimising a machine learning objective) for obtaining tensor representations. The first is the Kronecker representation which simply lifts a vector for a verb to a matrix by taking its outer product with itself:

$$\overline{verb} = \overrightarrow{verb} \otimes \overrightarrow{verb}$$

The second representation that was experimented with by Grefenstette and Sadrzadeh [GS11a] but also by Kartsaklis, Sadrzadeh, and Pulman [KSP13] and Milajevs et al. [Mil+14], is the Relational model, which uses dependency information about a words to construct its tensor representation. For a given word w that is connected to k sets of words $w_1, ..., w_n$ via a fixed set of dependencies $d_1, ..., d_n$, the tensor representation of w becomes

$$\mathbb{W} = \sum_i \left(\overrightarrow{w_1} \otimes \overrightarrow{w_2} \otimes ... \otimes \overrightarrow{w_n}\right)_i$$

[1] This is similar to the regression approach of Paperno and Baroni [PB+14].

That is, the word is represented by the summation over the tensor product between its dependency argument vectors. In the case of an adjective, this would give a vector

$$\overline{adj} = \sum_i \overrightarrow{noun}_i$$

over all the possible nouns that were modified by the adjective, whereas transitive verbs get represented by a matrix

$$\overline{verb} = \sum_i (\overrightarrow{subj}_i \otimes \overrightarrow{obj}_i)$$

Regression Methods In Baroni and Zamparelli [BZ10], it is argued that adjectives, being functional words, should be learned generalisations that (linearly) map a noun vector to its modified adjective-noun combination. To this end, the authors use a linear regression method, as follows: for a given adjective, create the *holistic* adjective-noun vectors by observing the co-occurrence statistics for adjective-noun word combinations. Assuming that noun vectors are available for all the nouns that were observed in combination with the adjective, the adjective matrix is now estimated using a standard linear regression technique to approximate

$$\overline{A} \times \overrightarrow{n} = \overrightarrow{An}$$

As experimentally verified [BZ10], this generally gives a suitable estimation of the adjective as a linear map.

For verbs, Grefenstette et al. [Gre+13] generalise the regression method for adjectives to the case of transitive verbs, using a multistep regression approach. The first step is to learn a matrix using linear regression that generalises over subjects versus verb phrases, i.e. given holistic subject-verb-object combinations, a verb-object matrix is learnt:

$$\overrightarrow{subj}^T \times \overline{verb\ obj} = \overrightarrow{subj\ verb\ obj}$$

In the next regression step, the matrices that have been learnt in the previous step form the input over which to approximate a cube representation for the verb. That is, the verb-object matrices of the first regression step are approximated by a cube that transforms object vectors into the relevant verb-object matrix:

$$\overline{\overline{verb}} \times \overrightarrow{obj} = \overline{verb\ obj}$$

In Grefenstette et al. [Gre+13] this approach is shown to lead to an improvement of results on the **GS2011** dataset, with a correlation score of 0.32 for the regression model with a dimensionality reduction technique applied. However, in general, this model is very data-intensive as a cube needs to be trained to generalise over previously trained matrices; if not enough data is available, the method can't produce high quality tensors.

Plausibility Models As an alternative approach, Polajnar, Rimell, and Clark [PRC14] proposes to modify the structure of the sentence space for training verb representations. Rather than approaching a holistic vector for a subject-verb-object combination, the model proposes a plausibility space, in which a combination of words will either be plausible or implausible. Concretely, this allows one to apply a logistic regression model as follows: given a dependency parsed corpus, gather all the subject-verb-object combinations in that corpus. These form the *plausible* triples. Now, either fully random or based on noun frequency bins, generate unobserved subject-verb-object triples, these are the *implausible* triples. Now, the verb

will be a matrix, that approximates a function that maps plausible subject and object vectors to 1, whereas it maps implausible subject and object vectors to 0. So the basic objective function that needs to optimised is

$$\frac{1}{N}\sum_i t_i \log \sigma(\overrightarrow{subj_i}^T \times \overline{verb} \times \overrightarrow{obj_i})$$

with N the number of (positive and negative) training examples, and t_i either 1 or 0, according to the plausibility of the subject and object. Although the plausibility model is reported to gain performance over analytical methods (in Polajnar, Rimell, and Clark [PRC14], the results are 0.35 on the **GS2011** dataset, 0.33 on the **KS2014** dataset), it's main strength appears to be in handling disambiguation (as the result on the sentence similarity dataset **KS2014** is significantly below that of a simple additive model). Moreover, this method would not easily generalise to arbitrary tensor representations.

Adjective Skipgram Maillard and Clark [MC15] take a different approach to learning a tensor representation, by adapting the skipgram model with negative sampling [Mik+13] to approximate matrices for adjectives. First, the skipgram model with negative sampling [Mik+13] generates word embeddings by optimising a logistic regression objective in which target vectors should have high inner product with context vectors for positive contexts (as observed from a linear context window around a target word), and low inner product with negatively sampled contexts. Given a target word n and a set of positive contexts C, a set of negative contexts $\overline{C}$ is sampled from a unigram distribution raised to some power (here: 3/4, after [LGD15]). The number k of negative contexts per positive one is a parameter of the model, here we consider $k = 10$. Context words are subsampled to decrease the difference between very frequent and very infrequent words. Initially, both target vectors $\mathbf{n}$ and context vectors $\mathbf{c}$ are randomly intialised, and during training the model updates both target and context vectors to maximise the objective function

$$\sum_{c\in C} \log \sigma(\mathbf{n}\cdot\mathbf{c}) + \sum_{\overline{c}\in\overline{C}} \log \sigma(-\mathbf{n}\cdot\overline{\mathbf{c}}) \tag{5.1}$$

To modify the skipgram model for the case of adjectives, Maillard and Clark [MC15] build a matrix representation into the model: a given adjective is represented as a matrix that, applied to a noun vector, predicts the context for the given adjective-noun combination. For a given adjective matrix $\overline{A}$, the model now lifts the objective function of skipgram to

$$\sum_{c\in C} \log \sigma(\overline{A}\mathbf{n}\cdot\mathbf{c}) + \sum_{\overline{c}\in\overline{C}} \log \sigma(-\overline{A}\mathbf{n}\cdot\overline{\mathbf{c}}) \tag{5.2}$$

As an adjective-noun combination itself is a noun, the context matrix is fixed and taken from a pretrained noun skipgram model.

A need for compact representations Most of the methods described above suffer from data sparsity issues: representing a verb as a cube means that the number of parameters of the tensor to be trained rises drastically; and this problem is barely mitigated by the amount of training data one has available: even in a large corpus the number of occurrences of verbs with their arguments is a lot lower than the number of word occurrences. So one of the desiderata of a tensor representation of words is that they be as low-dimensional as possible, without giving up on the core philosophy of representing words as functions. Where the

analytical methods did not suffer from data issues, their performance differs from dataset to dataset, i.e. they are not robust, and they depend too much on the particular data that was used to construct them. The skipgram modelling that Maillard and Clark [MC15] use is more robust, but covers only the limited case of adjectives, which allows the authors to restrict themselves to a one-faceted extension of the original skipgram model, as the notion of context does not need to be adapted. In the general case this simplification does not hold: already for verbs one can't argue that the context of a verb combined with a subject and and object will be the same as the context of a noun, so we need to change the notion of context in the case of verbs. Thus, our plan is to start from the skipgram model as it has shown to be robust and efficiently trainable for noun vectors and adjective matrices, but to generalise the model to represent arbitrary words as tensors in a structured way. We then perform an evaluation to test our model for the case of verbs, first of all on a number of verb similarity datasets, and consequently on all the datasets that we evaluated on in the previous chapter. The results of this study then pave the way for a new generation of tensor-based models, which we hope to implement in future work.

5.2 Neural Verb Tensor Embeddings

We generalise the skipgram model in two ways, to learn tensor representations of arbitrary words. First, we change the notion of *context* from a linear context window to contexts following a dependency tree, which is reminiscent of the dependency based word embeddings of Levy and Goldberg [LG14], which also uses dependency relations to generalise skipgram, but unlike their embeddings we do not incorporate the dependency labels themselves in our training model, but rather let the *architecture* of the model depend on these. The second modification is exactly to change the architecture according to the dependency relations of a word: for a given word w, the words $d_1, ..., d_n$ that modify w with the specified (fixed) dependency labels $l_1, ..., l_n$, determine the shape of the tensor that is being trained for w, as well as providing the context words for training. In addition, we may impose a specific tensor shape to the model of a given word, leading to three different types of models which we formally define below.

Full Tensor Skipgram For a given word W with dependency arguments $d_1, d_2, ..., d_n$ (as described above), a *full tensor* model trains a rank n tensor representation with the following objective function:

$$\sum_{c \in C} \log \sigma(\overline{\overline{\boldsymbol{W}}}\mathbf{d}_1 ... \mathbf{d}_n \cdot \mathbf{c}) + \sum_{\overline{c} \in \overline{C}} \log \sigma(-\overline{\overline{\boldsymbol{W}}}\mathbf{d}_1 ... \mathbf{d}_n \cdot \overline{\mathbf{c}}) \tag{5.3}$$

When W is an adjective, the objective function of the full tensor skipgram model instantiates the adjective skipgram model of Maillard and Clark [MC15], since an adjective has only one dependency argument d_1:

$$\sum_{c \in C} \log \sigma(\overline{\overline{\boldsymbol{W}}}\mathbf{d}_1 \cdot \mathbf{c}) + \sum_{\overline{c} \in \overline{C}} \log \sigma(-\overline{\overline{\boldsymbol{W}}}\mathbf{d}_1 \cdot \mathbf{c}) \tag{5.4}$$

When W is a noun, the objective function of the full tensor skipgram model reduces to the regular skipgram model with negative sampling since nouns do not have any dependency arguments:

$$\sum_{c \in C} \log \sigma(\overline{\overline{\boldsymbol{W}}} \cdot \mathbf{c}) + \sum_{\overline{c} \in \overline{C}} \log \sigma(-\overline{\overline{\boldsymbol{W}}} \mathbf{d}_1 ... \mathbf{d}_n \cdot \mathbf{c}) \tag{5.5}$$

Partial Tensor Skipgram The full tensor skipgram model above assumes that an appropriate notion of context is available after contracting the target tensor with all the vectors for its dependency arguments. Indeed, for adjectives this is the case as the adjective-noun combination is again a noun and so the linear context window can be carried over from the definition of the original skipgram model. This does not hold for arbitrary words, e.g., in the case of verbs the verb-subject-object combination is not a noun or a verb, but a sentence. To remedy this issue, we define a *partial* model in which one dependency is left out of the composition and used as context. Now, for a word W with dependency arguments $d_1, d_2, ..., d_n$, we train a tensor that is of rank $n-1$, providing a decomposition of the tensor of W into n separate lower rank tensors, one for each dependency. For an arbitrary dependency d_i, the objective function becomes as follows

$$\sum_{d_i \in C} \log \sigma(\overline{\overline{\boldsymbol{W}}} \mathbf{d}_1 ... \mathbf{d}_{i-1} \mathbf{d}_{i+1} ... \mathbf{d}_n \cdot \mathbf{d}_i) + \sum_{\overline{d_i} \in \overline{C}} \log \sigma(-\overline{\overline{\boldsymbol{W}}} \mathbf{d}_1 ... \mathbf{d}_{i-1} \mathbf{d}_{i+1} ... \mathbf{d}_n \cdot \mathbf{d}_i) \tag{5.6}$$

Lower Rank Tensor Skipgram If we decrease the rank of the tensor even more, we parameterise over the contexts as well as over the dependency arguments, leading to a choice of the dependencies to include in the context. Formally speaking, a rank $n-i$ tensor is learnt with the objective function as below:

$$\sum_{d_1 ... d_i \in C} \log \sigma(\overline{\overline{\boldsymbol{W}}} \mathbf{d}_{i+1} ... \mathbf{d}_n \cdot \mathcal{P}_+\{\mathbf{d}_1, ..., \mathbf{d}_i\}) + \sum_{\overline{d_1} ... \overline{d_i} \in \overline{C}} \log \sigma(\overline{\overline{\boldsymbol{W}}} \mathbf{d}_{i+1} ... \mathbf{d}_n \cdot \mathcal{P}_+\{\overline{\mathbf{d_1}}, ..., \overline{\mathbf{d_i}}\}) \tag{5.7}$$

where $\mathcal{P}_+\{\mathbf{d}_1, ..., \mathbf{d}_i\}$ means we choose some positive subset of the available dependency arguments as context. These models are exemplified in the following paragraphs for the case of transitive verbs.

Verb Skipgram We instantiate our tensor skipgram model on transitive verbs. These have as dependencies both a subject and an object. We summarise all trained models by rank of the tensor and choice of context in Table 5.1, and give a detailed explanation below.

Representation	Rank	Context
$\mathbf{v}_a$	vector	linear window
$\mathbf{v}_s / \mathbf{v}_o / \mathbf{v}_b$	vector	objects/subjects/both
$\overline{V}_a^S / \overline{V}_a^O$	matrix	full sentence
$\overline{V}_o^S / \overline{V}_s^O$	matrix	objects/subjects
$\overline{\overline{V}}_a$	cube	full sentence

TABLE 5.1: Our verb representations, ranging from vectors to cubes, with general or restricted context. $\mathbf{v}_a$ denotes a standard skipgram vector, with sub- and negative sampling. For the full sentence contexts, the arguments that are transformed by the representation are excluded from the context.

The full rank tensor model of Equation 5.3 for a transitive verb results in learning a rank 3 tensor, i.e. a cube. Given a verb V, we denote its first argument d_1, i.e. its subject, by s, and its second argument d_2, i.e. its object, by o. The full objective function of this model is as follows:

$$\sum_{c \in C} \log \sigma(\overline{\overline{V}}\mathbf{os} \cdot \mathbf{c}) + \sum_{\overline{c} \in \overline{C}} \log \sigma(-\overline{\overline{V}}\mathbf{os} \cdot \overline{\mathbf{c}}) \tag{5.8}$$

As all dependency arguments are transformed, the notion of context reduces to some context window, which we choose to let range over the full sentence, as the arguments themselves may occur at any position in the sentence in which the verb occurred. This model gives representation $\overline{\overline{V}}_a$ from Table 5.1.

Next, we instantiate the partial model of Equation 5.6, and train a subject matrix $\overline{V}^S$ and an object matrix $\overline{V}^O$. In the former case, the matrix tries to predict the object of the verb, given a fixed subject vector, as in Equation 5.9, where O refers to the set of contexts, which are *objects*. In the latter case, we train a matrix that predicts the subject of the verb, given the fixed object vector (Equation 5.10, contexts are *subjects* in S).

$$\sum_{o \in O} \log \sigma(\overline{V}\mathbf{s} \cdot \mathbf{o}) + \sum_{\overline{o} \in \overline{O}} \log \sigma(-\overline{V}\mathbf{s} \cdot \overline{\mathbf{o}}) \tag{5.9}$$

$$\sum_{s \in S} \log \sigma(\mathbf{s} \cdot \overline{V}\mathbf{o}) + \sum_{\overline{s} \in \overline{S}} \log \sigma(\overline{\mathbf{s}} \cdot -\overline{V}\mathbf{o}) \tag{5.10}$$

These models we refer to as $\overline{V}^S_o$ and $\overline{V}^O_s$ in Table 5.1, respectively. To see if this decomposition in combination with a dependency based context produces a sensible model, we compare these with $\overline{V}^S_a$ and $\overline{V}^O_a$, which still transform the subject (resp. object) vector, but predict a full sentential context rather than a single dependency.

Last, we instantiate Equation 5.7 to produce vector representations for the verb. Given that there are two dependencies, the different choice of context leads to three models $\mathbf{v}_s$, $\mathbf{v}_o$, $\mathbf{v}_b$ that respectively take only subjects, only objects, or both arguments as context. We compare these to $\mathbf{v}_a$, which is the original skipgram model with a linear context window.

Implementation Details We implemented all models in Python, using the `tensorflow` package [Aba+16][2]. Vectors were 100-dimensional; matrices and cubes were shaped accordingly. The dependency information was extracted from a dependency parsed corpus[3] containing ca. 130M sentences, on which the initial regular noun vectors were also trained. In the case of matrices and cubes with full context ($\overline{V}_a, \overline{\overline{V}}_a$), a pair of networks was trained separately for each verb, sharing the context matrix from the noun skipgram model. For the matrices with subject (resp. object) context, we trained a pair of networks with an embedding layer encoding all verbs. In these networks, the context matrix consists of all possible object (resp. subject) context vectors. Here we considered both a fixed context matrix (from the noun skipgram model) and a trainable context matrix and found that the trainable context matrix gave the best results[4], so we work with the latter. Negative samples were drawn from the distribution over possible objects (resp. subjects) of all verbs in the case of the partial tensor models. We considered $k = 10$ negative samples per subject (resp. object).

[2] Code and models will be made available online.

[3] We used the UKWaCkypedia corpus, available from `wacky.sslmit.unibo.it`

[4] We argue that this is because contexts in the noun skipgram model are more general as they serve as contexts to many different target words.

5.3 Evaluation

We now discuss the evaluation of the new representations that we learn on a number of similarity and compositional tasks. The main goal here is to see which of the different types of representation that we learn, works best in a number of tasks. As a second goal, we want to see if these representations outperform a non-neural baseline, for which we use the analytical models that we experimented with in the previous chapter. Finally, we want to establish whether the hybrid neural tensor-based setting is capable of achieving similar results to state of the art sentence encoders and contextualised embeddings. As we train verb representations, we evaluate on three types of tasks: verb similarity, verb disambiguation – where the goal is to distinguish a verb's meaning given a sentential context – and compositional sentence similarity. Below we discuss all tasks, composition models and similarity metrics that we experimented with, including a short reference to the sentence encoders and contextualised embeddings that we compare our approach with.

Verb Similarity For verb similarity, we consider the verb only partitions of a number of word similarity datasets, as well as datasets aimed at evaluating verb similarity in itself. First of all, we consider those pairs of words from the MEN [Bru+12] and SimLex-999 [HRK15] datasets that were labelled as verbs, obtaining 22 and 222 verb similarity pairs, respectively. Next to these partial datasets, we considered VerbSim [YP06], a dataset of 130 verb pairs, and the more recent SimVerb-3500 dataset [Ger+16], containing 3500 verb pairs. For the latter, we independently evaluate on the development set (500 pairs) and the test set (3000 pairs).

Compositional Tasks We consider the seven compositional tasks that we also evaluated on in Chapter 4: first, the two intransitive sentence datasets **ML2008** and **ML2010** introduced by Mitchell and Lapata [ML08; ML10], the first aiming at disambiguating the verb of each sentence, the second evaluating general sentence similarity. Next we consider the transitive variants of these datasets: the transitive verb disambiguation datasets of Grefenstette and Sadrzadeh [GS11a] (**GS2011**) and Kartsaklis and Sadrzadeh [KS13] (**KS2013**), and the transitive sentence similarity dataset of Kartsaklis, Sadrzadeh, and Pulman [KSP13] (**KS2014**). Finally, we also evaluate on the verb phrase elliptical disambiguation (**ELLDIS**) and similarity (**ELLSIM**) tasks introduced in the previous chapter.

Non-Neural Baseline Models We compare our neural verb representations (Table 5.1) with the two non-neural verb representation methods from the type-driven literature [GS11a], already mentioned in the introduction of this chapter, but reiterated below. On the left is the Kronecker representation, on the right is the relational representation:

$$\overline{V}_{Kron} = \mathbf{v}_a \otimes \mathbf{v}_a \qquad \overline{V}_{Rel} = \sum_i \mathbf{s}_i \otimes \mathbf{o}_i$$

Fusion We consider two ways of fusing our neural verb matrices into a single representation, *middle* and *late* fusion representations after Bruni, Tran, and Baroni [BTB14]. Middle fusion takes a weighted average of the two verb representations, using the result to compute similarity scores. Late fusion uses each representations to compute separate similarity scores and then averages the results. Given a weighted average $M_\alpha(A, B) = \alpha A + (1 - \alpha)B$ for $\alpha \in [0..1]$, and V_1, V_2 verbs, the middle and late fusion operations are defined as follows:

$$\text{sim}(M_\alpha(V_1^S, V_1^O), M_\alpha(V_2^S, V_2^O)) \tag{5.11}$$

$$M_\alpha(\text{sim}(V_1^S, V_2^S), \text{sim}(V_1^O, V_2^O)) \tag{5.12}$$

The same fusion operations are used in the compositional tasks, where either the verb matrices are averaged before composition, or the cosine scores are averaged after. Our values for α are 0.1 increments ranging from 0 (only the subject matrix) to 1 (only the object matrix).

Clustering In their article introducing the adjective skipgram model, Maillard and Clark [MC15] argue that "cosine similarity, while suitable for vectors, does not capture any information about the function of matrices as linear maps". This argument holds for generalisations of matrices to higher order tensors, such as cubes and tesseracts. The functions of these tensors are multilinear maps, e.g. bilinear for cubes and trilinear for tesseracts. Thus, following Maillard and Clark, we postulate that a suitable measure of similarity for two functions should be related to how similarly they transform their arguments. We say two words W, W' with tensor representations $\mathbb{W}, \mathbb{W}'$ and arguments $d_1, \cdots, d_n$ are similar whenever $\mathbb{W}\mathbf{d}_1 ... \mathbf{d}_n$ is similar to $\mathbb{W}'\mathbf{d}_1 ... \mathbf{d}_n$, for the vector $\mathbf{d}_i$ of every argument that they have transformed in the corpus. The degree of similarity between W and W' is obtained by taking the median of the degrees of similarities of their applications on the arguments. Since going through all the instantiations of the arguments is expensive, we cluster the most frequent argument vectors and work with the similarity between the two transformations applied to the centroids of each cluster. The resulting similarity function is defined as follows, for $\mathcal{D}$ the set of tuples of cluster centroids:

$$\text{tensorsim} : \underset{\langle \mathbf{d}_1, \ldots, \mathbf{d}_n \rangle \in \mathcal{D}}{\text{med}} \cos(\mathbb{W}\mathbf{d}_1 ... \mathbf{d}_n, \mathbb{W}'\mathbf{d}_1 ... \mathbf{d}_n) \tag{5.13}$$

In the case of cube embeddings for transitive verbs, this definition is equivalent to considering the most frequent subjects and objects of the verb, clustering them separately, then applying the cube to the centroid vectors and take the median. The metrics that we use for the different representations from Table 5.1 are given in Table 5.2.

Metric	**Formula**
vecsim	$\cos(\mathbf{a}, \mathbf{b}) = \dfrac{\mathbf{a} \cdot \mathbf{b}}{\lvert\mathbf{a}\rvert\lvert\mathbf{b}\rvert}$
matsim^S	$\underset{\mathbf{s} \in \mathcal{S}}{\text{med}} \cos(\overline{V}_1\mathbf{s}, \overline{V}_2\mathbf{s})$
matsim^O	$\underset{\mathbf{o} \in \mathcal{O}}{\text{med}} \cos(\overline{V}_1\mathbf{o}, \overline{V}_2\mathbf{o})$
cubesim	$\underset{\langle \mathbf{s}, \mathbf{o} \rangle \in \mathcal{A}}{\text{med}} \cos(\mathbb{V}_1\mathbf{os}, \mathbb{V}_2\mathbf{os})$

TABLE 5.2: Similarity metrics on vectors, matrices and cubes, based on clustering centroids.

Composition Models Similar to the evaluation study in Chapter 4, we define a number of composition models to test on the datasets involving sentences. For the current study, we consider three baseline models: either a non-compositional model involving just the verb representation (with clustering applied in the cases of matrix and cube representations), or a compositional baseline, which are the arithmetic models of [ML10], which given a sentence $w_1 w_2 ... w_n$ produces the addition or multiplication of its word vectors. Furthermore, the compositional models based on the non-neural verb representations are also considered.

The results for these will be different from those in Chapter 4, as the underlying word vectors are different, i.e. 100-dimensional skipgram vectors. Then, we evaluate composition models for the different neural verb representations, where the main new models take a verb matrix/cube representation, compose it with vectors of its subject and object, and compute a final sentence representation via middle or late fusion. Composition in these models is (a variant of) tensor contraction, as used for example in [MCG14; KSP13; Mil+14].

Intransitive Models The datasets **ML2008** and **ML2010**, respectively, contain pairs of subject-verb, and verb-object phrases. Next to the arithmetic baseline that adds the vectors, we apply middle and late fusion on the separate subject-verb and verb-object matrices, with as a special case an unmixed model for the case where a single matrix verb embedding is available. The specification of these fusion models for subject-verb and verb-object phrases are in the table below:

Phrase	*subj verb*	*verb obj*
Middle	$M_\alpha(\overline{V}^S, \overline{V}^O)\ \mathbf{s}$	$M_\alpha(\overline{V}^S, \overline{V}^O)\ \mathbf{o}$
Late	$M_\alpha(\overline{V}^S\mathbf{s}, \overline{V}^O\mathbf{s})$	$M_\alpha(\overline{V}^S\mathbf{o}, \overline{V}^O\mathbf{o})$

Transitive Models To model a transitive sentence of the form *subj verb obj*, we compare verb-only and arithmetic baselines with tensor-based models as below:

Model type	**Formula**
Middle	$T(\mathbf{s}, M_\alpha(\overline{V}^S, \overline{V}^O), \mathbf{o})$
Late	$M_\alpha(T(\mathbf{s}, \overline{V}^S, \mathbf{o}), T(\mathbf{s}, \overline{V}^O, \mathbf{o}))$
Two	$M_\alpha(T_s(\mathbf{s}, \overline{V}^S, \mathbf{o}), T_o(\mathbf{s}, \overline{V}^O, \mathbf{o}))$
Cube	$\mathbb{V}\mathbf{os}$

TABLE 5.3: Composition models for transitive sentences. T represents any standard tensor-based composition model for transitive sentences, T_s is subject-directed composition, T_o is object-directed composition. When $\alpha = 0$ or $\alpha = 1$, the models reduce to the case of using one of the two verb matrix embeddings.

However, note that here we define a new class of models (*Two*), where we apply a separate model for the $\overline{V}^S$ matrix with the subject vector, and a distinct model for the $\overline{V}^O$ matrix with the object vector, that then get fused to give a final representation. Concretely, this leads to three new composition models, Copy Argument, Copy Argument Sum and Categorical Argument, described below:

Model	**Formula**
CA	$M_\alpha\left(\mathbf{s}^T\overline{V}^S \odot \mathbf{o}, \overline{V}^O\mathbf{o} \odot \mathbf{s}\right)$
CAS	$M_\alpha\left(\mathbf{s}^T\overline{V}^S + \mathbf{o}, \overline{V}^O\mathbf{o} + \mathbf{s}\right)$
CATA	$M_\alpha\left(\mathbf{s}^T\overline{V}^S, \overline{V}^O\mathbf{o}\right)$

Sentence Encoders and Contextualised Representations Similar to the previous chapter, we include also the results of state of the art sentence encoders and contextualised representations, to allow for a comparison with our proposed modelling. In the case of sentence encoders, we take a number of different pretrained models and directly encode the sentences in the datasets of interest, where for the verb phrase elliptical datasets we report also the resolved sentence encodings. We consider the same encoders used in Chapter 4: Doc2vec [LB16], Skipthoughts [Kir+15], InferSent [Con+17], and Universal Sentence Encoder [Cer+18]. For the case of contextualised embeddings, we take pretrained models for ELMo [Pet+18] and BERT [Dev+19][5] to give a contextualised representation for the words in a sentence, then take the mean of these to give a sentence embedding.

5.4 Results

In this section we present the experimental results on the tasks described in section 5.3, comparing the various different verb representations.

Verb Similarity The correlation results on verb similarity tasks are displayed in Table 5.4. The first two columns are Spearman's ρ between the original skipgram verb vectors and the vectors trained on subjects and objects as context using the similarity metrics of Table 5.2. For the matrix and cube representations, we report the best scores of the subject-verb or verb-object matrices, using the matrix similarity metric from Table 5.2 and the parameterised middle/late fusion, and refer the reader to Appendix B. Finally, for the cubes the reported score is the highest obtained by the different verb argument clustering configurations, again with the full results tables in Appendix B.

	$\mathbf{v}_a$	$\mathbf{v}_{s/o/b}$	$\overline{V}_a$	$\overline{V}_{s/o}$	$\mathbb{V}_a$
MEN$_v$	0.282	0.248	0.500	**0.589**	0.035
SimLex$_v$	0.046	0.272	0.163	**0.340**	0.024
VerbSim	0.338	**0.563**	0.085	0.550	-0.076
SimVerb$_d$	0.224	0.249	-0.023	**0.291**	-0.012
SimVerb$_t$	0.183	0.197	0.019	**0.240**	-0.025

TABLE 5.4: Spearman ρ correlation on verb similarity datasets. The subscript v indicates that we are looking at the partial verb-only dataset. For SimVerb we distinguish between the development **SimVerb**$_d$ and test set **SimVerb**$_d$. We compare standard skipgram vectors $\mathbf{v}_a$ with specific context vectors $\mathbf{v}_{s/o/b}$, matrices with full sentence context $\overline{V}_a$, our model that predicts one dependency argument $\overline{V}_{s/o}$, and cubes with a full sentence context $\mathbb{V}_a$.

For the case of verb vectors, the general skipgram model is outperformed by the vectors trained on the verb arguments as context, and in fact these show the highest performance on the VerbSim dataset. That the matrix and cube representations with the full sentence as

[5]For ELMo, we used Google's module at https://tfhub.dev/google/elmo/2, for BERT we used the python `bert_embedding` package from https://github.com/imgarylai/bert-embedding.

context perform rather poorly, and in many cases worse than the vector representations, illustrates that the choice of context is too general for these higher-order representations. On four out of the five tasks, however, our proposed method of training matrices with a restricted notion of context, outperforms all other models, most significantly so for the 3000 entry test subset of the SimVerb dataset where we observed an increase from 0.183 to 0.240.

Neural Tensor Clustering Table 5.5 shows the correlation scores on the verbs of the compositional tasks discussed in the previous section. In this experiment, we are performing the sentence disambiguation and similarity tasks by only using the verbs of the sentences. In each case, we first apply the verb tensors to their subject and object clusters, use middle or late fusion where appropriate, compute their degrees of similarity, and use this degree for disambiguation or a straight similarity calculation. The implementation configurations are the same as in the verb similarity tasks. We observe the same pattern in the results: re-

	$\mathbf{v}_a$	$\mathbf{v}_{s/o/b}$	$\overline{V}_a$	$\overline{V}_{s/o}$	$\mathbb{V}_a$
ML2008	0.067	0.055	0.161	0.124	**0.178**
ML2010$_v$	0.396	0.528	-0.004	**0.638**	0.003
GS2011	0.226	0.331	0.369	**0.399**	-0.028
KS2013a	0.184	0.100	0.062	**0.218**	0.003
KS2013b	0.445	0.638	-0.055	**0.695**	-0.025
ELLDIS	0.341	0.389	0.386	**0.516**	0.047
ELLSIM	0.370	0.577	0.022	**0.643**	0.011

TABLE 5.5: Spearman ρ correlation for verbs of compositional tasks. Each score is a maximum score out of possible clusters and fusion weights. We again compare standard skipgram vectors $\mathbf{v}_a$ with specific context vectors $\mathbf{v}_{s/o/b}$, matrices with full sentence context $\overline{V}_a$, our model that predicts one dependency argument $\overline{V}_{s/o}$, and cubes with a full sentence context $\mathbb{V}_a$.

training of the verb vectors slightly improves the performance. This is against the erratic performance of the full context matrix representations and the very poor performance of the cube representations (on all but the **ML2008** dataset). Again, our proposed matrix representations with a restricted context significantly outperforms the other methods. However, note that these results again are the highest of a parameter sweep over the cluster and fusion settings of the representations, with full results in Appendix B.

Compositional Models The most interesting results come from the compositional tasks. These compose a representation for each sentence of the dataset by taking into account the representations of all of the words within that sentence, rather than by only working with individual word representations, as was done in the previous two tasks. The results in Table 5.6 show three baseline models on the left, and the three tensor skipgram representations that we trained, on the right. First, there is the arithmetic baseline $C(+, \odot)$ where we compose either by adding or point-wise multiplying all the vectors in a sentence. Then, there are two non-neural baseline models for which we compose by using either the Kronecker verb matrix, $C(\overline{V}_{Kron})$, or the relational matrix $C(\overline{V}_{Rel})$, in one of the composition models described above. We do the same for the two neural matrix representations on the right: first,

there is the matrix that is trained by predicting a full sentence context after transforming either the verb's subject or object ($C(\overline{V}_a)$), whereas our proposed representations is the verb matrix that transforms one of its arguments (subject/object) and predicts the other argument as context ($C(\overline{V}_{s/o})$). Finally, we compare to the cube model, in which a cube is trained by transforming both subject and object vectors, and predicting the remainder of the sentence as context ($C(\mathbb{V}_a)$).

	Baseline			**Neural**		
	$C(+,\odot)$	$C(\overline{V}_{Kron})$	$C(\overline{V}_{Rel})$	$C(\overline{V}_a)$	$C(\overline{V}_{s/o})$	$C(\mathbb{V}_a)$
ML2008	0.171	0.082	**0.192**	-0.045	0.188	—
ML2010$_v$	0.541	0.402	0.511	0.000	**0.550**	—
GS2011	0.187	0.205	0.323	0.247	**0.536**	-0.021
KS2013	0.181	0.281	0.188	0.203	**0.372**	-0.043
KS2013b	0.672	0.530	0.511	0.542	**0.753**	0.064
ELLDIS	0.308	0.304	0.368	0.221	**0.559**	0.030
ELLSIM	0.671	0.522	0.646	0.532	**0.759**	0.093

TABLE 5.6: Spearman ρ scores on compositional tasks. $C(+,\odot)$ denotes arithmetic models, whereas the other rows represent the best score for a compositional model with the different verb representations ($C(\overline{V}_{Kron})$: Kronecker matrix, $C(\overline{V}_{Rel})$: Relational matrix, $C(\overline{V}_a)$: Skipgram matrix with sentence context, $C(\overline{V}_{s/o})$: Skipgram matrix with argument as context, $C(\mathbb{V}_a)$: Skipgram cube with sentence context).

The results table shows that the neurally trained verb matrices with full sentences as context don't significantly improve performance compared to the non-neural compositional baselines, and in the case of disambiguation they are inferior. This shows that the choice of context matters a lot: here, the full sentence is taken as a context, but this is not discriminatory enough to achieve high correlation, whereas for instance the Relational matrix directly encodes subject and object information, allowing it to be more robust on the compositional tasks.

Similarly to the verb similarity results, the cubes show a very poor performance, which we argue is due to data sparsity. Even though the cubes implicitly model properties of arguments of the verbs, their representation is too sparse to effectively model anything. Moreover, as with the verb matrices $\overline{V}_a$, they consider the full sentence as context since there is no straightforward other way of defining this. Our proposed matrix model remedies both the sparsity problem and the choice of context, and outperforms all the other representations, save on the **ML2008** dataset.

Sentence Encoders and Contextualised Representations We compare the results of our proposed neural tensor embeddings with sentence encoder models in Table 5.7, and with the ELMo and BERT embeddings in Table 5.8.

Where we see a pattern similar to our study in Chapter 4 for the sentence encoders, namely that the tensor-based models work well on disambiguation tasks, but alternative sentence encoding methods work better on similarity tasks, we find that our embeddings generally outperform the contextualised encodings (ELMO, BERT) on all tasks. Although it is still an open question to what extent such language models are able to encode syntactic information, they definitely do not encode dependency information explicitly as in our proposal, which

	$C(\overline{V}_{s/o})$	D2V1	D2V2	ST	IS1	IS2	IS3	IS4	USE
ML2008	0.188	0.139	0.192	0.078	0.181	**0.220**	0.149	0.169	0.039
ML2010	0.550	0.512	0.447	0.494	0.631	0.492	**0.636**	0.405	0.325
GS2011	**0.536**	0.098	0.102	-0.157	0.297	0.320	0.324	0.213	0.094
KS2013	**0.372**	0.193	0.212	0.051	0.172	0.032	0.176	-0.021	0.210
KS2014	0.753	0.692	0.705	0.546	**0.784**	0.676	0.720	0.586	0.539
MLELLDIS$_{lin}$	—	0.090	0.233	0.232	0.199	0.224	0.108	0.135	0.105
MLELLDIS$_{res}$	0.221	0.089	0.216	0.167	0.228	**0.269**	0.144	0.156	0.154
MLELLDIS$_{abl}$	—	0.095	0.242	0.159	0.215	0.226	0.141	0.169	0.109
ELLDIS$_{lin}$	—	0.199	0.227	-0.193	0.347	0.384	0.330	0.344	0.269
ELLDIS$_{res}$	**0.559**	0.231	0.253	-0.172	0.344	0.337	0.293	0.248	0.277
ELLDIS$_{abl}$	—	0.195	0.259	-0.130	0.353	0.357	0.300	0.291	0.240
ELLSIM$_{lin}$	—	0.593	0.622	0.585	0.779	0.701	0.748	0.641	0.647
ELLSIM$_{res}$	0.760	0.698	0.692	0.604	**0.803**	0.749	0.768	0.687	0.680
ELLSIM$_{abl}$	—	0.652	0.655	0.471	0.782	0.730	0.749	0.682	0.640

TABLE 5.7: Spearman ρ scores on compositional tasks, with state of the art sentence encoders. D2V1: Doc2Vec1, D2V2: Doc2Vec 2, ST: Skip-Thought, IS1: InferSent 1 (4096), IS2: InferSent 2 (4096), IS3: InferSent 1 (300), IS4: InferSent 2 (300), USE: Universal Sentence Encoder.

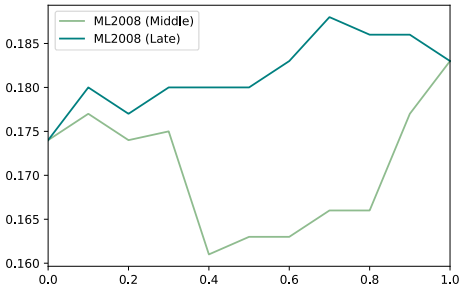
could explain the beneficial performance of our representations on the evaluation tasks that tend to contain relatively short sentences with a focus on syntactic awareness. We see this reflected in the fact that the contextualised embeddings of ELMO perform best on the **ELLSIM** dataset, granted that the ellipsis is resolved first.

The influence of the fusion parameter One interesting aspect of our proposed matrix model is that two separate matrices are trained, that each optimise the prediction of one of the verbs dependency arguments (subject/object), given the other argument. When composing a sentence embeddings, we then have a choice of setting the parameter α to fuse together the matrices, or their respective compositions. To see what is the effect of this parameter, we look at the influence of the value of $alpha$ — 0 for the pure subject-verb matrix, 1 for the pure verb-object matrix, weighted sum of both in between — on the performance on the compositional tasks. For each dataset, we show the average effect across all tested composition models, showing this effect both for middle and late fusion. Table 5.9 displays the effect of the α parameter on the intransitive sentence datasets **ML2008** and **ML2010**.
In the case of ML2008 there is a preference for the subject-verb matrix (objects as contexts), and it is the other way around for **ML2010**. Performance goes down once the matrices are mixed in middle fusion (the light green line), whereas generally the late fusion boosts performance (the dark green line), illustrating the important of the choice between middle and late fusion.

Table 5.10 shows the effect of α on all other datasets. As a general pattern, on these datasets the choice of middle versus late fusion has significance (with generally later fusion being the better choice) though the effect of α is the same regardless of the fusion type, except on the **GS2011** dataset. What is most notable is that for both **GS2011** and **ELLDIS**, which contain the same verbs, there is a preference for the subject-verb matrix, with the peaks of the graphs for values of α lower than 0.5, whereas the verb-object matrix is more important

	$C(\overline{V}_{s/o})$	ELMo	BERT Small	BERT Large
ML2008	**0.188**	0.166	0.105	0.030
ML2010$_v$	**0.550**	0.539	0.216	0.356
GS2011	**0.536**	0.108	0.187	0.292
KS2013	**0.372**	0.243	0.232	0.349
KS2014	**0.753**	0.728	0.520	0.616
MLELLDIS$_{lin}$	—	0.193	**0.373**	0.315
MLELLDIS$_{res}$	0.221	0.182	0.103	0.342
MLELLDIS$_{abl}$	—	0.123	0.089	0.193
ELLDIS$_{lin}$	—	0.232	0.360	0.368
ELLDIS$_{res}$	**0.559**	0.210	0.216	0.274
ELLDIS$_{abl}$	—	0.207	0.197	0.365
ELLSIM$_{lin}$	—	0.734	0.595	0.580
ELLSIM$_{res}$	0.759	**0.779**	0.631	0.647
ELLSIM$_{abl}$	—	0.703	0.560	0.582

TABLE 5.8: Spearman ρ scores on compositional tasks, with state of the art contextualised embeddings. For BERT, we use the small and large versions, for the small version we use both uncased and cased book corpus.



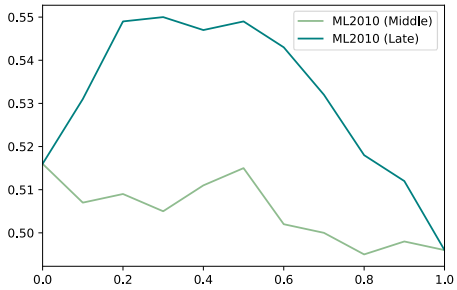


TABLE 5.9: The effect of the α fusion parameter on middle and late fusion for intransitive sentence datasets.

in the other datasets. The explicit results for all values of α and per composition operator are listed in Appendix B.

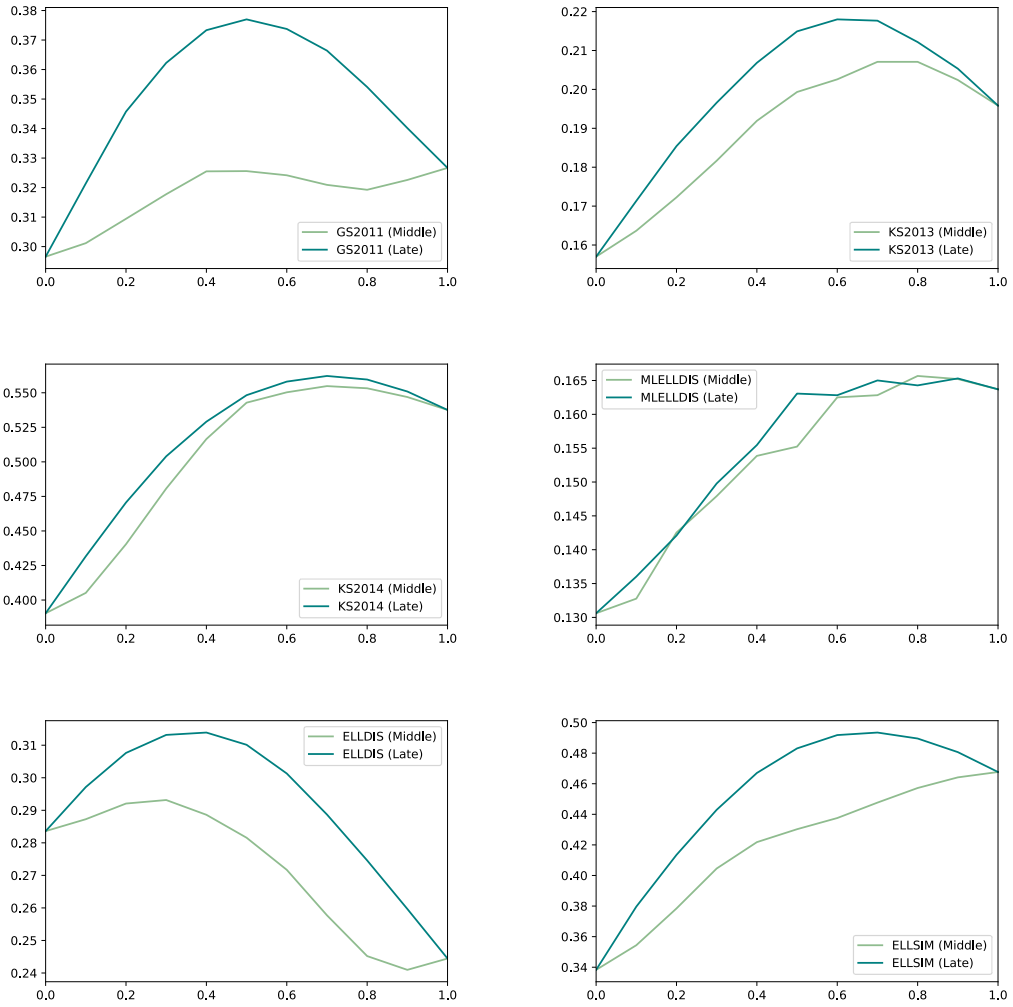


TABLE 5.10: The effect of the α fusion parameter on middle and late fusion for datasets **GS2011**, **KS2013**, KS2014, **MLELLDIS**, **ELLDIS**, and **ELLSIM**.

5.5 Conclusion

Type-driven compositional distributional semantics has shown that the symbolic formal semantic structure of a sentence can be transformed into a vectorial form, by representing the words therein as tensors whose ranks depend on their grammatical roles: nouns are represented as vectors, matrices as adjectives, transitive verbs as cubes and so on. Tensors are multilinear maps and thus the type-driven distributional models offer a canonical form of composing them: via tensor contraction. The tensors, however, are high dimensional, and it is not clear how they should be learned.

In this final contribution chapter of this thesis, we generalised the widely used skipgram model to learn neural tensor embeddings for words with any number of dependencies. The role of a word tensor is to transform the embeddings of its dependencies and train a skipgram objective to predict context vectors for the results of these transformations. The notion of context can vary here: we worked with full sentence contexts as well as restricted versions thereof, where a tensor is applied to some of its dependencies and the results are used

to predict the rest. Our model reduces to the original noun skipgram model when no dependencies are involved, and covers the adjective-noun skipgram model of Maillard and Clark [MC15], where there is only one dependency.

We implemented our model on transitive verbs, learning cubes for them in a full sentence context, and matrix and vector approximations in a restricted subject and object context. In the approximated cases, the learned verb-subject matrix is applied to the subject vectors to predict the object context vectors, and the verb-object matrix is applied to the object vectors to predict the subject context vectors.

We experimented on word similarity, sentence similarity and verb disambiguation tasks. For verb similarity, we considered the verb-only fragments of MEN and SimLex-999, the 130 element VerbSim dataset [YP06], and the SimVerb-3500 dataset of Gerz et al. [Ger+16]. Our neural matrix embeddings provided the best results in all of the tasks, beating the full rank tensors as well as the baseline vectors – both with the general and restricted contexts. We further tested our models on the intransitive sentence similarity dataset of Mitchell and Lapata [ML10] and its transitive extension [KSP13], as well as to the intransitive sentence disambiguation dataset of Mitchell and Lapata [ML08] and its transitive extensions [GS11a; KS13]. The results have the same pattern: the neural matrix approximations of verbs outperform their neural cube representations, their non-neural matrix representations, the additive and multiplicative models, and the verb-only vector and tensor baselines. We inspected the effect of fusion on task performance, and saw that some tasks benefitted more from a focus on the objects-as-contexts verb matrices, whereas other tasks preferred the subjects-as-context verb matrices.

Given the full generality of the model and the promising initial experimental results, the tensor skipgram model paves the way for a new generation of type-driven distributional semantic models. In the disambiguation and similarity tasks that we evaluated on in this study, we found that the best models of verbs were those that used a specific context — the subject or object — which came from a dependency parsed corpus. Although this is not a *pure* tensor-based model as it uses two matrix approximations of the verb, this model strikes a balance between feasibility of training on the one hand and specificity of the encoded information on the other hand. In future work, we aim to expand this model in two ways: first, we wish to investigate more of the properties of the representations to gain a better insight of what is really encoded in the representations themselves — as opposed to the more task-centric view we held here. Second, we would like to develop these representations for any word in a dependency parsed sentence, which would then allow us to evaluate not just on focussed tasks but also on more general natural language understanding tasks, such as natural language inference, or question answering.

Part IV

Further Down

Chapter 6

Conclusion & Future Work

This thesis presented the result of a three years' investigation into compositional distributional models. We summarise here the main contributions of the thesis and end with some directions for future endeavours.

6.1 Summary

Where we started off by giving the general background of word embeddings and their compositionality in Chapter 1, in Chapter 2 we delved deeper into a particular, type-driven, approach to composition, which started off with the work of Coecke, Sadrzadeh, and Clark [CSC10] and Coecke, Grefenstette, and Sadrzadeh [CGS13], describing in the language of category theory how to interpret the grammatical structure of a sentence as a multi-linear transformation applied to the embeddings of the individual words in a sentence.

The models developed along these lines have been experimented with extensively [GS15; KSP13; Mil+14], but all assume that there is a one-on-one relation between the text of a sentence and its meaning. Thus, the challenge we addressed in this thesis, is that of finding a compositional distributional model that is robust against cases in which the meaning of a sentence is not explicitly given by its surface form; the test case for this was ellipsis with anaphora. To solve this challenge, we developed the theory for a compositional vector space model of ellipsis and anaphora, relying on a unimodal extension of the Lambek Calculus to provide a grammatical model for ellipsis and anaphora, in Chapter 3. This model could then deal with the recovering of implicit semantic content (as one finds in examples of ellipsis), by means of a limited form of contraction in the grammar logic. We discussed how this model uses different structural rules to accommodate different linguistic phenomena, and as such can be ported to deal with pronoun relativisation as well. That the theory does not always suit the implementation was shown by the fact that a model that directly maps types to vector spaces and proofs to (multi-)linear maps will give unwanted predictions in the presence of ambiguous elliptical phrases: different interpretations of an ambiguous sentence were shown to coincide in meaning. To amend this, we relaxed the model to a setting in which a non-linear term calculus interprets our grammar logic with controlled contraction. We then show how the structural ambiguity puzzle can be solved on the level of semantics.

In order to give experimental support for the models that deal with ellipsis and anaphora, we introduced in Chapter 4 three new datasets that allow one to contrast concrete distributional models that do not resolve ellipsis, i.e. the what-you-see-is-what-you-get approach, with models that perform linguistic analysis to give the intended meaning of a sentence.

Using these new tasks, we showed that indeed resolving verb phrase ellipsis gives a positive boost to the correlation of a model with human judgments. Moreover, the experiments showed that state of the art neural sentence embeddings are not always the optimal choice when assessing sentence comprehension.

Finally, in Chapter 5 we address the issue of lexical semantics in a tensor-based model: although some methods have been around to concretely derive the content of word tensors, most of these approaches either suffer from data sparsity issues or from overparameterised training models. To amend this, we formulated a generalisation of the well-known skip-gram model [Mik+13] to describe a class of models that may be implemented for any word of any grammatical type. We instantiated this model on the case of transitive verbs, and evaluated on all the compositional distributional tasks that we had considered so far. The results indicated that our matrix decomposition model, which trains two separate matrices per verb, always outperformed previous analytical approaches to verb representation, and mostly outperformed neural sentence encoders and contextualised embeddings.

6.2 Further Down

The work in this thesis offers a number of potential future research directions, both on the level of theory and on the level of experimentation.

On the theoretical side, this thesis gives a theoretical extension of the categorical framework of Coecke, Grefenstette, and Sadrzadeh [CGS13] by making use of the Lambek Calculus with the structural control modalities of [Moo96]. A number of issues with our proposal remain open. First of all, we worked with a controlled form of contraction. However, the results of Kanovich, Kuznetsov, and Scedrov [KKS16; KKS17] show that even a controlled form of contraction may lead to undecidability of the employed grammar logic. Although the possibility of undecidability does not directly impede concrete experimentation with the models, it does beg further investigation on the formal properties of the proposed theoretical model. One possible solution is to bound the number of applications of the contraction rule to get a light version of the logic that by its bound becomes decidable. Another theoretical issue was that the direct interpretation of types and proofs as vector spaces and multi-linear maps was not able to distinguish the strict and sloppy readings of verb phrase ellipsis. Although we mended this by modelling the vector semantics with a lambda calculus as an intermediary language, a current approach to solve this issue without relying on lambdas, is to replace vector spaces with Fock spaces: these are spaces which offer a choice between any number of tensor products of a given vector space. One can then extract the relevant number of copies of a vector space whenever the grammar logic wills it.

On the experimental side, one of the limits of the type-driven approach to composition of distributional representation is *scalability*: while neural approaches are able to encode arbitrary sentences, the type-driven approach lacks this flexibility for two reasons. First, unbounded parsing remains a challenge in the typelogical approach; second, the tensor-based approach relies on high quality tensor representations for words, and methods to learn these have been theoretically established — especially in the last chapter of this thesis — but haven't been implemented on full scale yet. Thus, one potential avenue for research is to make use of systems that have learned how to assign types to words in a typelogical grammar.

Work on extracting categorial lexicons form structured data is abundant: there is the unification algorithm of Buszkowski and Penn [BP90], extensions [BR01], and the thesis of Kanazawa [Kan95]. In addition, there is the CCGBank [HS07], which contains CCG derivations for the dependency trees in the Penn Treebank. On top of the CCGBank, parsers and supertaggers have been developed [XAC15; LLZ16]. On the multimodal front, Moot [Moo15] describes a typelogical treebank for French and Moot [Moo18] a language-independent chart parsing method. One promising new direction that interacts well with the approach advocated in this thesis, uses self-attention neural networks to create a system that can, after training, map raw text to word types [KMD19].

Next to extracting grammatical types, the other main avenue for research is to expand the implementation that we gave in Chapter 5 of a tensor-based skipgram model to scale up to longer sentences, and be able to evaluate these representations on large datasets such as the Stanford Natural Language Inference datasets (SNLI, [Bow+15]) and it's multi-genre and crosslingual incarnations (MNLI, [WNB18], XNLI, [Con+18]).

Finally, another potential line of research is to extract a large scale tensor-based model from current state of the art contextualised embeddings like ELMo [Pet+18] and BERT [Dev+19]. Such models assume that the embedding of a word is a vector that is a function of all other words in the sentence the word occurs in. This parameterisation naturally allows one to model some words as a higher order tensor that, by the fact that it encodes a multilinear map, contextualises itself in its arguments. For example, in the adjective-noun case, the vector for the adjectivally modified noun depends on the vector for the noun and the effect of the adjective matrix on it.

Then, to end: although the contributions of this thesis provide support for the relevance of type-driven models in a world that appears to be focused on neural methods, many questions arise. It is my hope that future research in this field may improve our understanding of the relevance of linguistic knowledge in semantic representations.

Bibliography

[Aba+16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. "Tensorflow: A system for large-scale machine learning". In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 265–283.

[Abr09] Samson Abramsky. "No-cloning in categorical quantum mechanics". In: *Semantic Techniques in Quantum Computation* (2009), pp. 1–28.

[ALM19] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. "A simple but tough-to-beat baseline for sentence embeddings". In: *5th International Conference on Learning Representations, ICLR 2017*. 2019.

[BBZ14] Marco Baroni, Raffaela Bernardi, and Roberto Zamparelli. "Frege in space: A program of compositional distributional semantics". In: *LiLT (Linguistic Issues in Language Technology)* 9 (2014).

[Bim14] Katalin Bimbó. *Proof theory: Sequent calculi and related formalisms*. CRC Press, 2014.

[Boj+17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching word vectors with subword information". In: *Transactions of the Association of Computational Linguistics* 5.1 (2017), pp. 135–146. URL: https://www.aclweb.org/anthology/Q17-1010.

[Bow+15] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. "A large annotated corpus for learning natural language inference". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 632–642.

[BP90] Wojciech Buszkowski and Gerald Penn. "Categorial grammars determined from linguistic data by unification". In: *Studia Logica* 49.4 (1990), pp. 431–454.

[BR01] Roberto Bonato and Christian Retoré. "Learning rigid lambek grammars and minimalist grammars from structured sentences". In: *Third workshop on learning language in logic, Strasbourg*. 2001, pp. 23–34.

[Bru+12] Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. "Distributional semantics in technicolor". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume*

1. Association for Computational Linguistics. 2012, pp. 136–145. URL: https://www.aclweb.org/anthology/P12-1015.

[BS11] Johan Bos and Jennifer Spenader. "An annotated corpus for the analysis of VP ellipsis". In: *Language Resources and Evaluation* 45.4 (2011), pp. 463–494.

[BTB14] Elia Bruni, Nam-Khanh Tran, and Marco Baroni. "Multimodal distributional semantics". In: *Journal of Artificial Intelligence Research* 49 (2014), pp. 1–47.

[Bus01] Wojciech Buszkowski. "Lambek grammars based on pregroups". In: *International Conference on Logical Aspects of Computational Linguistics*. Springer. 2001, pp. 95–109.

[BZ10] 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*. Association for Computational Linguistics. 2010, pp. 1183–1193. URL: https://www.aclweb.org/anthology/D10-1115.

[Cer+18] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. "Universal sentence encoder". In: *arXiv preprint arXiv:1803.11175* (2018). URL: https://arxiv.org/pdf/1803.11175.pdf.

[CGS13] Bob Coecke, Edward Grefenstette, and Mehrnoosh Sadrzadeh. "Lambek vs. Lambek: Functorial vector space semantics and string diagrams for Lambek calculus". In: *Annals of pure and applied logic* 164.11 (2013), pp. 1079–1100.

[CH16] Karel Chvalovskỳ and Rostislav Horčík. "Full lambek calculus with contraction is undecidable". In: *The Journal of Symbolic Logic* 81.2 (2016), pp. 524–540.

[Chu40] Alonzo Church. "A formulation of the simple theory of types". In: *The Journal of Symbolic Logic* 5.2 (1940), pp. 56–68.

[Cla13] Stephen Clark. "Type-driven syntax and semantics for composing meaning vectors". In: *Quantum Physics and Linguistics: A Compositional, Diagrammatic Discourse* (2013), pp. 359–377.

[Cla15] Stephen Clark. "Vector Space Models of Lexical Meaning". In: *The Handbook of Contemporary Semantic Theory*. John Wiley & Sons, Ltd, 2015. Chap. 16, pp. 493–522. ISBN: 9781118882139. DOI: 10.1002/9781118882139.ch16. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118882139.ch16. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118882139.ch16.

[Con+17] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 670–680.

[Con+18] Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. "XNLI: Evaluating Cross-lingual Sentence Representations". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 2475–2485. DOI: 10.18653/v1/D18-1269. URL: https://www.aclweb.org/anthology/D18-1269.

[CP01] Peter W Culicover and Paul M Postal. *Parasitic gaps: A history*. MIT Press, 2001.

[CSC10] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. "Mathematical foundations for a compositional distributional model of meaning". In: *Linguistic Analysis* 36.1 (2010), pp. 345–384.

[CWB18] WooJin Chung, Sheng-Fu Wang, and Samuel R. Bowman. "The Lifted Matrix-Space Model for Semantic Composition". In: *Proceedings of the 22nd Conference on Computational Natural Language Learning*. 2018, pp. 508–518.

[DC19] András Dobó and János Csirik. "A comprehensive study of the parameters in the creation and comparison of feature vectors in distributional semantic models". In: *Journal of Quantitative Linguistics* (2019), pp. 1–28.

[Dev+19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2019.

[DSP91] Mary Dalrymple, Stuart M Shieber, and Fernando CN Pereira. "Ellipsis and higher-order unification". In: *Linguistics and philosophy* 14.4 (1991), pp. 399–452.

[Fin+01] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. "Placing search in context: The concept revisited". In: *Proceedings of the 10th international conference on World Wide Web*. ACM. 2001, pp. 406–414.

[Fir57] John R Firth. "A synopsis of linguistic theory, 1930-1955". In: *Studies in linguistic analysis* (1957).

[FPC15] Daniel Fried, Tamara Polajnar, and Stephen Clark. "Low-rank tensors for verbs in compositional distributional semantics". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Vol. 2. 2015, pp. 731–736.

[Ger+16] Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. "SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 2173–2182.

[Gre+13] Edward Grefenstette, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. "Multi-step regression learning for compositional distributional semantics". In: *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)*. 2013.

[Gre13] Edward Grefenstette. "Category-Theoretic Quantitative Compositional Distributional Models of Natural Language Semantics". PhD thesis. Oxford, UK: University of Oxford, 2013. URL: http://arxiv.org/abs/1311.1539.

[Gro01] Philippe de Groote. "Towards Abstract Categorial Grammars". In: *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*. ACL '01. Toulouse, France: Association for Computational Linguistics, 2001, pp. 252–259.

[GS11a] Edward Grefenstette and Mehrnoosh Sadrzadeh. "Experimental support for a categorical compositional distributional model of meaning". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2011, pp. 1394–1404.

[GS11b] Edward Grefenstette and Mehrnoosh Sadrzadeh. "Experimenting with transitive verbs in a discocat". In: *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. Association for Computational Linguistics. 2011, pp. 62–66.

[GS15] Edward Grefenstette and Mehrnoosh Sadrzadeh. "Concrete models and empirical evaluations for the categorical compositional distributional model of meaning". In: *Computational Linguistics* 41.1 (2015), pp. 71–118.

[Har54] Zellig S Harris. "Distributional structure". In: *Word* 10.2-3 (1954), pp. 146–162.

[Hen95] P. Hendriks. *Comparatives and Categorial Grammar*. Groningen dissertations in linguistics. Grodil, 1995. URL: https://books.google.co.uk/books?id=FeNUuAAACAAJ.

[HRK15] Felix Hill, Roi Reichart, and Anna Korhonen. "Simlex-999: Evaluating semantic models with (genuine) similarity estimation". In: *Computational Linguistics* 41.4 (2015), pp. 665–695.

[HS07] Julia Hockenmaier and Mark Steedman. "CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank". In: *Computational Linguistics* 33.3 (2007), pp. 355–396.

[HS19] Jules Hedges and Mehrnoosh Sadrzadeh. "A Generalised Quantifier Theory of Natural Language in Categorical Compositional Distributional Semantics with Bialgebras". In: *Mathematical Structures in Computer Science* 29.6 (2019), pp. 783–809.

[Iyy+15] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. "Deep unordered composition rivals syntactic methods for text classification". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Vol. 1. 2015, pp. 1681–1691.

[Jac11] Bart Jacobs. "Bases as coalgebras". In: *International Conference on Algebra and Coalgebra in Computer Science*. Springer. 2011, pp. 237–252.

[Jac99] Pauline Jacobson. "Towards a variable-free semantics". In: *Linguistics and philosophy* 22.2 (1999), pp. 117–185.

[Jäg06] Gerhard Jäger. *Anaphora and type logical grammar*. Vol. 24. Springer Science & Business Media, 2006.

[Jäg98] Gerhard Jäger. "A Multi-Modal Analysis of Anaphora and Ellipsis". In: *University of Pennsylvania Working Papers in Linguistics* 5.2 (1998), p. 2.

[Kan95] Makoto Kanazawa. "Learnable Classes of Categorial Grammars". PhD thesis. Stanford University, 1995.

[Kar15] Dimitri Kartsaklis. "Compositional Distributional Semantics with Compact Closed Categories and Frobenius Algebras". PhD thesis. University of Oxford, 2015.

[Kar16] Dimitri Kartsaklis. "Coordination in categorical compositional distributional semantics". In: *arXiv preprint arXiv:1606.01515* (2016).

[KC14] Douwe Kiela and Stephen Clark. "A systematic study of semantic vector space model parameters". In: *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*. 2014, pp. 21–30.

[Kem+15] Ruth Kempson, Ronnie Cann, Eleni Gregoromichelaki Arasheshghi, and Matthew Purver. "Ellipsis". In: *Chapter 4 of The Handbook of Contemporary Semantic Theory* 3 (2015), p. 114.

[Kin01] Walter Kintsch. "Predication". In: *Cognitive science* 25.2 (2001), pp. 173–202.

[Kir+15] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. "Skip-thought vectors". In: *Advances in neural information processing systems*. 2015, pp. 3294–3302.

[KKS16] Max Kanovich, Stepan Kuznetsov, and Andre Scedrov. "Undecidability of the Lambek calculus with a relevant modality". In: *International Conference on Formal Grammar*. Springer. 2016, pp. 240–256.

[KKS17] Max Kanovich, Stepan Kuznetsov, and Andre Scedrov. "Undecidability of the Lambek calculus with subexponential and bracket modalities". In: *International Symposium on Fundamentals of Computation Theory*. Springer. 2017, pp. 326–340.

[KKS19] Max Kanovich, Stepan Kuznetsov, and Andre Scedrov. "Undecidability of a Newly Proposed Calculus for CatLog3". In: *International Conference on Formal Grammar*. Springer. 2019, pp. 67–83.

[KL17] Yusuke Kubota and Robert Levine. "Pseudogapping as Pseudo-VP-Ellipsis". In: *Linguistic Inquiry* 48.2 (2017), pp. 213–257.

[KM97] Natasha Kurtonina and Michael Moortgat. "Structural control". In: *Specifying syntactic structures* (1997), pp. 75–113.

[KMD19] Konstantinos Kogkalidis, Michael Moortgat, and Tejaswini Deoskar. "Constructive Type-Logical Supertagging with Self-Attention Networks". In: *4th Workshop on Representation Learning for NLP, ACL* (2019).

[KPS16] Dimitri Kartsaklis, Matthew Purver, and Mehrnoosh Sadrzadeh. "Verb Phrase Ellipsis using Frobenius Algebras in Categorical Compositional Distributional Semantics". In: *DSALT Workshop, European Summer School on Logic, Language and Information* (2016).

[Kru+16] Germán Kruszewski, Denis Paperno, Raffaella Bernardi, and Marco Baroni. "There is no logical negation here, but there are alternatives: Modeling conversational negation with distributional semantics". In: *Computational Linguistics* 42.4 (2016), pp. 637–660.

[KS13] Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. "Prior disambiguation of word tensors for constructing sentence vectors". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013, pp. 1590–1601.

[KS14] Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. "A Study of Entanglement in a Categorical Framework of Natural Language". In: *Proceedings of the 11th Workshop on Quantum Physics and Logic (QPL)*. Kyoto ,Japan. 2014.

[KSP12] Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. "A Unified Sentence Space for Categorical Distributional- Compositional Semantics: Theory and Experiments". In: *Proceedings of 24th International Conference on Computational Linguistics (COLING): Posters*. Mumbai ,India. 2012.

[KSP13] Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. "Separating Disambiguation from Composition in Distributional Semantics". In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 114–123. URL: https://www.aclweb.org/anthology/W13-3513.

[Lam58] Joachim Lambek. "The mathematics of sentence structure". In: *The American Mathematical Monthly* 65.3 (1958), pp. 154–170.

[Lam61] Joachim Lambek. "On the calculus of syntactic types". In: *Structure of language and its mathematical aspects* 166 (1961), p. C178.

[Lam68] Joachim Lambek. "Deductive systems and categories". In: *Theory of Computing Systems* 2.4 (1968), pp. 287–318.

[Lam88] Joachim Lambek. "Categorial and categorical grammars". In: *Categorial grammars and natural language structures*. Springer, 1988, pp. 297–317.

[Lam99] Joachim Lambek. "Type grammar revisited". In: *International Conference on Logical Aspects of Computational Linguistics*. Springer. 1999, pp. 1–27.

[LB16] Jey Han Lau and Timothy Baldwin. "An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation". In: *Proceedings of the 1st Workshop on Representation Learning for NLP*. 2016, pp. 78–86. DOI: 10.18653/v1/W16-1609. URL: https://www.aclweb.org/anthology/W16-1609.

[LB96] Kevin Lund and Curt Burgess. "Producing high-dimensional semantic spaces from lexical co-occurrence". In: *Behavior research methods, instruments, & computers* 28.2 (1996), pp. 203–208.

[LD97] Thomas K Landauer and Susan T Dumais. "A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge." In: *Psychological review* 104.2 (1997), p. 211.

[Len08] Alessandro Lenci. "Distributional semantics in linguistic and cognitive research". In: *Italian journal of linguistics* 20.1 (2008), pp. 1–31.

[LG14] Omer Levy and Yoav Goldberg. "Dependency-based word embeddings". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vol. 2. 2014, pp. 302–308.

[LGD15] Omer Levy, Yoav Goldberg, and Ido Dagan. "Improving distributional similarity with lessons learned from word embeddings". In: *Transactions of the Association for Computational Linguistics* 3 (2015), pp. 211–225.

[LLZ16] Mike Lewis, Kenton Lee, and Luke Zettlemoyer. "Lstm ccg parsing". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016, pp. 221–231.

[LM14] Quoc Le and Tomas Mikolov. "Distributed representations of sentences and documents". In: *International Conference on Machine Learning*. 2014, pp. 1188–1196.

[LS88] Joachim Lambek and Philip J Scott. *Introduction to higher-order categorical logic*. Vol. 7. Cambridge University Press, 1988.

[Mar+14] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. "A SICK cure for the evaluation of compositional distributional semantic models". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. Reykjavik, Iceland: European Languages Resources Association (ELRA), May 2014, pp. 216–223. URL: http://www.lrec-conf.org/proceedings/lrec2014/pdf/363_Paper.pdf.

[MC15] Jean Maillard and Stephen Clark. "Learning adjective meanings with a tensor-based skip-gram model". In: *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. 2015, pp. 327–331.

[MC91] George A Miller and Walter G Charles. "Contextual correlates of semantic similarity". In: *Language and cognitive processes* 6.1 (1991), pp. 1–28.

[MCG14] Jean Maillard, Stephen Clark, and Edward Grefenstette. "A type-driven tensor-based semantics for CCG". In: *Proceedings of the EACL 2014 Type Theory and Natural Language Semantics Workshop*. 2014, pp. 46–54.

[Mer01] Jason Merchant. *The syntax of silence: Sluicing, islands, and the theory of ellipsis*. Oxford University Press on Demand, 2001.

[Mik+13] 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*. 2013, pp. 3111–3119.

[Mil+14] 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)*. 2014, pp. 708–719. URL: https://www.aclweb.org/anthology/D14-1079.

[ML08] Jeff Mitchell and Mirella Lapata. "Vector-based models of semantic composition". In: *proceedings of ACL-08: HLT* (2008), pp. 236–244.

[ML10] Jeff Mitchell and Mirella Lapata. "Composition in distributional models of semantics". In: *Cognitive science* 34.8 (2010), pp. 1388–1429.

[MMS96] Glyn Morrill and Josep-Maria Merenciano Saladrigas. "Generalising discontinuity". In: *TAL. Traitement automatique des langues* 37.2 (1996), pp. 119–143.

[Mon70a] Richard Montague. "English as a formal language". In: *Linguaggi nella Società e nella Tecnica* (1970), pp. 189–224.

[Mon70b] Richard Montague. "Universal grammar". In: *Theoria* 36.3 (1970), pp. 373–398.

[Mon73] Richard Montague. "The proper treatment of quantification in ordinary English". In: *Approaches to natural language*. Springer, 1973, pp. 221–242.

[Moo09] Michael Moortgat. "Symmetric categorial grammar". In: *Journal of Philosophical Logic* 38.6 (2009), pp. 681–710.

[Moo15] Richard Moot. "A type-logical treebank for french". In: *Journal of Language Modelling* 3.1 (2015), pp. 229–264.

[Moo18] Richard Moot. "Chart Parsing Multimodal Grammars". In: *arXiv preprint arXiv:1804.02286* (2018).

[Moo96] Michael Moortgat. "Multimodal linguistic inference". In: *Journal of Logic, Language and Information* 5.3-4 (1996), pp. 349–385.

[Moo97] Michael Moortgat. "Categorial type logics". In: *Handbook of logic and language*. Ed. by Johan van Benthem and Alice ter Meulen. Amsterdam: Elsevier, 1997, pp. 93–177. ISBN: 978-0-444-81714-3. DOI: https://doi.org/10.1016/B978-044481714-3/50005-9. URL: http://www.sciencedirect.com/science/article/pii/B9780444817143500059.

[Mor12] Glyn V Morrill. *Type logical grammar: Categorial logic of signs*. Springer Science & Business Media, 2012.

[MS16] Reinhard Muskens and Mehrnoosh Sadrzadeh. "Context Update for Lambdas and Vectors". In: *International Conference on Logical Aspects of Computational Linguistics*. Springer. 2016, pp. 247–254.

[MS17] Reinhard Muskens and Mehrnoosh Sadrzadeh. "Lambdas, Vectors, and Word Meaning in Context". In: *Proceedings of the 21st Amsterdam Colloquium*. 2017, pp. 65–74. URL: https://semanticsarchive.net/Archive/jZiM2FhZ/AC2017-Proceedings.pdf.

[Mus03] Reinhard Muskens. "Language, Lambdas, and Logic". In: *Resource-Sensitivity, Binding and Anaphora*. Dordrecht: Springer Netherlands, 2003, pp. 23–54.

[MV10] Glyn Morrill and Oriol Valentın. "On calculus of displacement". In: *Proceedings of the 10th International Workshop on Tree Adjoining Grammars and Related Formalisms*. 2010, pp. 45–52.

[MV15] Glyn Morrill and Oriol Valentín. "Computational Coverage of TLG: Nonlinearity". In: *Proceedings of NLCS'15. Third Workshop on Natural Language and Computer Science*. Vol. 32. EasyChair Publications. 2015, pp. 51–63.

[MV16] Glyn Morrill and Oriol Valentín. "On the Logic of Expansion in Natural Language". In: *Logical Aspects of Computational Linguistics. Celebrating 20 Years of LACL (1996–2016) 9th International Conference, LACL 2016, Nancy, France, December 5-7, 2016, Proceedings 9*. Springer. 2016, pp. 228–246.

[MVF11] Glyn Morrill, Oriol Valentín, and Mario Fadda. "The displacement calculus". In: *Journal of Logic, Language and Information* 20.1 (2011), pp. 1–48.

[MW17] Michael Moortgat and Gijs Wijnholds. "Lexical and Derivational Meaning in Vector-Based Models of Relativisation". In: *Proceedings of the 21st Amsterdam Colloquium*. 2017.

[Pan05] Patrick Pantel. "Inducing ontological co-occurrence vectors". In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 2005, pp. 125–132.

[PB+14] Denis Paperno, Marco Baroni, et al. "A practical and linguistically-motivated approach to compositional distributional semantics". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 2014, pp. 90–99.

[Pet+18] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2018, pp. 2227–2237.

[PF01] Martin J Pickering and Steven Frisson. "Processing ambiguous verbs: Evidence from eye movements." In: *Journal of Experimental Psychology: Learning, Memory, and Cognition* 27.2 (2001), p. 556.

[PFC14] Tamara Polajnar, Luana Fagarasan, and Stephen Clark. "Reducing dimensions of tensors in type-driven distributional semantics". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1036–1046.

[PL07] Anne Preller and Joachim Lambek. "Free compact 2-categories". In: *Mathematical Structures in Computer Science* 17.2 (2007), pp. 309–340.

[Pol16] Tamara Polajnar. "Collaborative Training of Tensors for Compositional Distributional Semantics". In: *arXiv preprint arXiv:1607.02310* (2016).

[PRC14] Tamara Polajnar, Laura Rimell, and Stephen Clark. "Using sentence plausibility to learn the semantics of transitive verbs". In: *arXiv preprint arXiv:1411.7942* (2014).

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[RG65] Herbert Rubenstein and John B Goodenough. "Contextual correlates of synonymy". In: *Communications of the ACM* 8.10 (1965), pp. 627–633.

[Rim+16] Laura Rimell, Jean Maillard, Tamara Polajnar, and Stephen Clark. "Relpron: A relative clause evaluation data set for compositional distributional semantics". In: *Computational Linguistics* 42.4 (2016), pp. 661–701.

[Sad16] Mehrnoosh Sadrzadeh. "Quantifier Scope in Categorical Compositional Distributional Semantics". In: *arXiv preprint arXiv:1608.01404* (2016).

[SCC13] Mehrnoosh Sadrzadeh, Stephen Clark, and Bob Coecke. "The Frobenius anatomy of word meanings I: subject and object relative pronouns". In: *Journal of Logic and Computation* 23.6 (2013), pp. 1293–1317.

[SCC14] Mehrnoosh Sadrzadeh, Stephen Clark, and Bob Coecke. "The Frobenius anatomy of word meanings II: possessive relative pronouns". In: *Journal of Logic and Computation* (2014), exu027.

[Sch98] Hinrich Schütze. "Automatic word sense discrimination". In: *Computational linguistics* 24.1 (1998), pp. 97–123.

[SMW19] Mehrnoosh Sadrzadeh, Michael Moortgat, and Gijs Wijnholds. "A Frobenius Algebraic Analysis for Parasitic Gaps". In: *Workshop on Semantic Spaces at the Intersection of NLP, Physics, and Cognitive Science*. 2019.

[SO96] Bayu Surarso and Horoakira Ono. "Cut elimination in noncommutative substructural logics". In: *Reports on Mathematical Logic* 30 (1996), pp. 13–29.

[Soc+13] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 1631–1642.

[Ste00] Mark Steedman. *The syntactic process*. MIT Press, 2000.

[SU06] Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard isomorphism*. Vol. 149. Elsevier, 2006.

[TC19] Aarne Talman and Stergios Chatzikyriakidis. "Testing the generalization power of neural network models across NLI benchmarks". In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. 2019, pp. 85–94.

[TP10] Peter D Turney and Patrick Pantel. "From frequency to meaning: Vector space models of semantics". In: *Journal of artificial intelligence research* 37 (2010), pp. 141–188.

[Val14] Oriol Valentín. "The hidden structural rules of the discontinuous Lambek calculus". In: *Categories and Types in Logic, Language, and Physics*. Springer, 2014, pp. 402–420.

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[Ver96] Jacobus Antonius Gerardus Versmissen. "Grammatical composition: modes, models, modalities: logical and linguistic aspects of multimodal categorial grammars". PhD thesis. 1996.

[Wan90] Heinrich Wansing. "Formulas-as-types for a hierarchy of sublogics of intuitionistic propositional logic". In: *Workshop on Nonclassical Logics and Information Processing*. Springer. 1990, pp. 125–145.

[Wij14] Gijs Wijnholds. "Categorical foundations for extended compositional distributional models of meaning". In: *MSc. thesis* (2014).

[Wij17] Gijs Jasper Wijnholds. "Coherent diagrammatic reasoning in compositional distributional semantics". In: *International Workshop on Logic, Language, Information, and Computation*. Springer. 2017, pp. 371–386.

[Wij19] Gijs Wijnholds. "A proof-theoretic approach to scope ambiguity in compositional vector space models". In: *Journal of Language Modelling* 6.2 (2019), pp. 261–286.

[WNB18] Adina Williams, Nikita Nangia, and Samuel R Bowman. "A broad-coverage challenge corpus for sentence understanding through inference". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2018, pp. 1112–1122.

[WS18] Gijs Wijnholds and Mehrnoosh Sadrzadeh. "Classical Copying versus Quantum Entanglement in Natural Language: The Case of VP-ellipsis". In: *EPTCS 283, 2018, pp. 103-119* (2018), pp. 103–119. DOI: 10.4204/EPTCS.283.8.

[WS19a] Gijs Wijnholds and Mehrnoosh Sadrzadeh. "A Type-Driven Vector Semantics for Ellipsis with Anaphora using Lambek Calculus with Limited Contraction". In: *Journal of Logic, Language and Information* (2019).

[WS19b] Gijs Wijnholds and Mehrnoosh Sadrzadeh. "Evaluating Composition Models for Verb Phrase Elliptical Sentence Embeddings". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2019.

[WSC19] Gijs Wijnholds, Mehrnoosh Sadrzadeh, and Stephen Clark. "Representation Learning for Type-Driven Composition". In: *Preprint*. 2019.

[XAC15] Wenduan Xu, Michael Auli, and Stephen Clark. "CCG supertagging with a recurrent neural network". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. 2015, pp. 250–255.

[YP06] Dongqiang Yang and David Martin Powers. "Verb similarity on the taxonomy of WordNet". In: *Proceedings of the Third International WordNet Conference GWC 2006, South Jeju Island, Korea*. Ed. by C. Fellbaum P. Sojka K.-S. Choi and P. Vossen. Masaryk University, 2006, pp. 121–128.

Appendix A

Evaluation Results (All Models)

This appendix accompanies Chapter 4 and contains tables with all the evaluation results for multiple datasets.

A.1 Results on transitive sentence datasets

GS2011	Count Based	Word2Vec	GloVe	FastText
Verb Only Vector	0.273	0.210	**0.302**	0.212
Verb Only Tensor	0.167	0.342	0.241	0.292
Additive	0.291	0.255	0.248	0.140
Multiplicative	0.431	0.204	0.217	0.193
Relational	0.222	**0.343**	0.252	0.323
Copy Subject $\bar{\cdot}$	0.149	0.302	0.217	0.292
Copy Object $\bar{\cdot}$	0.244	0.326	**0.315**	**0.339**
Frobenius Add $\bar{\cdot}$	0.220	**0.353**	0.295	0.330
Frobenius Mult $\bar{\cdot}$	0.217	0.285	0.102	0.271
Frobenius Outer $\bar{\cdot}$	0.216	0.334	0.298	**0.353**
Relational $\tilde{\cdot}$	0.392	0.202	0.257	0.188
Copy Subject $\tilde{\cdot}$	0.313	0.308	0.217	0.258
Copy Object $\tilde{\cdot}$	**0.454**	-0.010	0.225	0.205
Frobenius Add $\tilde{\cdot}$	0.403	0.192	0.258	0.227
Frobenius Mult $\tilde{\cdot}$	**0.447**	0.066	0.189	0.121
Frobenius Outer $\tilde{\cdot}$	0.392	0.069	0.262	0.187

TABLE A.1: Spearman ρ correlation scores on the **GS2011** dataset, with the highest score for each space in bold and in a box, the second highest result in bold. Results are against an inter-annotator agreement score of 0.739.

KS2013	Count Based	Word2Vec	GloVe	FastText
Verb Only Vector	0.108	0.199	0.132	0.112
Verb Only Tensor	0.093	0.100	0.065	0.040
Additive	0.104	0.210	**0.174**	0.117
Multiplicative	**0.279**	0.334	0.110	0.302
Relational $\bar{\cdot}$	0.126	0.188	0.052	0.129
Copy Subject $\bar{\cdot}$	0.047	0.223	0.041	0.125
Copy Object $\bar{\cdot}$	0.145	0.279	0.042	0.238
Frobenius Add $\bar{\cdot}$	0.152	0.295	0.055	0.211
Frobenius Mult $\bar{\cdot}$	0.131	0.187	0.011	0.169
Frobenius Outer $\bar{\cdot}$	0.129	0.305	0.040	0.186
Relational $\tilde{\cdot}$	0.215	0.314	0.105	0.268
Copy Subject $\tilde{\cdot}$	0.171	**0.408**	0.113	**0.334**
Copy Object $\tilde{\cdot}$	0.258	0.343	0.123	0.290
Frobenius Add $\tilde{\cdot}$	0.227	**0.415**	**0.140**	**0.368**
Frobenius Mult $\tilde{\cdot}$	**0.322**	0.182	0.102	0.218
Frobenius Outer $\tilde{\cdot}$	0.215	0.327	0.114	0.232

TABLE A.2: Spearman ρ correlation scores on the **KS2013** dataset, with the highest score for each space in bold and in a box, the second highest result in bold. Results are against an inter-annotator agreement score of 0.575.

KS2014	Count Based	Word2Vec	GloVe	FastText
Verb Only Vector	0.521	0.665	**0.535**	**0.705**
Verb Only Tensor	0.456	0.617	0.504	0.563
Additive	0.677	**0.763**	**0.719**	**0.764**
Multiplicative	0.719	0.528	0.283	0.587
Relational $\bar{\cdot}$	0.721	**0.623**	0.418	0.637
Copy Subject $\bar{\cdot}$	0.588	0.468	0.329	0.509
Copy Object $\bar{\cdot}$	0.672	0.499	0.405	0.593
Frobenius Add $\bar{\cdot}$	0.694	0.508	0.405	0.563
Frobenius Mult $\bar{\cdot}$	0.696	0.459	0.325	0.616
Frobenius Outer $\bar{\cdot}$	0.718	0.537	0.427	0.641
Relational $\tilde{\cdot}$	**0.739**	0.578	0.278	0.634
Copy Subject $\tilde{\cdot}$	0.670	0.481	0.208	0.569
Copy Object $\tilde{\cdot}$	0.715	0.270	0.267	0.316
Frobenius Add $\tilde{\cdot}$	0.726	0.440	0.278	0.542
Frobenius Mult $\tilde{\cdot}$	**0.745**	0.173	0.259	0.308
Frobenius Outer $\tilde{\cdot}$	**0.739**	0.189	0.278	0.339

TABLE A.3: Spearman ρ correlation scores on the **KS2014** dataset, with the highest score for each space in bold and in a box, the second highest result in bold. Results are against an inter-annotator agreement score of 0.754.

A.2 Results on verb phrase elliptical phrase datasets

MLELLDIS	Count Based	Word2Vec	GloVe	FastText
Verb Only Vector	0.078	0.153	0.329	0.095
Verb Only Tensor	0.067	-0.035	0.033	0.036
Additive	0.040	0.179	0.249	0.142
Multiplicative	**0.206**	0.081	0.003	0.103
Multiplicative $\odot$	**0.391**	0.085	0.017	0.124
Multiplicative $+$	0.179	**0.236**	0.055	0.156
Additive $\odot$	0.172	0.195	0.078	**0.201**
Additive $+$	0.078	**0.229**	**0.336**	0.170
Kronecker $\odot$	0.128	0.136	0.294	0.133
Kronecker $+$	0.076	0.207	**0.366**	0.085
Categorical $\odot$	0.060	0.143	0.008	0.190
Categorical $+$	0.047	0.177	0.092	**0.211**
FROB Kron	0.076	-0.001	0.032	0.091
FROB Cat	0.056	-0.084	0.004	0.071

TABLE A.4: Spearman ρ correlation scores on the **MLELLDIS** dataset, with the highest score for each space in bold and in a box, the second highest result in bold. Results are against an inter-annotator agreement score of 0.66.

ELLDIS	Count Based	Word2Vec	GloVe	FastText
Verb Only Vector	0.436	0.241	**0.445**	0.229
Verb Only Tensor	0.329	0.438	**0.394**	0.388
Additive	0.442	0.273	0.305	0.141
Additive Ablate	0.445	0.309	0.309	0.135
Multiplicative	0.325	-0.012	0.182	0.293
Multiplicative Ablate	0.524	0.278	0.286	0.136
Additive Non-Linear	0.445	0.328	0.326	0.140
Multiplicative Non-Linear	0.503	0.209	0.245	0.044
Relational $\overline{\odot}$	0.380	0.362	0.153	0.356
Copy Subject $\overline{\odot}$	0.309	0.336	0.182	0.383
Copy Object $\overline{\odot}$	0.348	0.353	0.174	0.350
Copy Subject Sum $\overline{\odot}$	0.302	0.392	0.267	0.396
Copy Object Sum $\overline{\odot}$	0.265	0.357	0.331	0.368
Frobenius Add $\overline{\odot}$	0.364	0.393	0.195	0.389
Frobenius Mult $\overline{\odot}$	0.315	0.270	0.126	0.235
Frobenius Outer $\overline{\odot}$	0.367	0.371	0.208	0.429
Relational $\overline{+}$	0.372	0.427	0.334	0.438
Copy Subject $\overline{+}$	0.316	0.401	0.299	0.442
Copy Object $\overline{+}$	0.336	0.443	0.358	0.432
Copy Subject Sum $\overline{+}$	0.303	0.396	0.361	0.420
Copy Object Sum $\overline{+}$	0.268	0.429	0.373	0.400
Frobenius Add $\overline{+}$	0.365	**0.462**	0.369	**0.465**
Frobenius Mult $\overline{+}$	0.354	0.400	0.121	0.374
Frobenius Outer $\overline{+}$	0.366	**0.454**	0.345	**0.494**
Relational $\widetilde{\odot}$	0.511	0.168	0.242	0.072
Copy Subject $\widetilde{\odot}$	0.457	0.151	0.230	0.065
Copy Object $\widetilde{\odot}$	0.518	-0.008	0.230	0.181
Copy Subject Sum $\widetilde{\odot}$	0.419	0.194	0.150	0.206
Copy Object Sum $\widetilde{\odot}$	0.420	0.050	0.174	0.208
Frobenius Add $\widetilde{\odot}$	0.521	0.132	0.261	0.180
Frobenius Mult $\widetilde{\odot}$	0.373	0.015	0.228	0.167
Frobenius Outer $\widetilde{\odot}$	0.511	0.019	0.247	0.179
Relational $\widetilde{+}$	0.523	0.252	0.337	0.229
Copy Subject $\widetilde{+}$	0.491	0.320	0.322	0.291
Copy Object $\widetilde{+}$	**0.539**	0.244	0.335	0.180
Copy Subject Sum $\widetilde{+}$	0.433	0.300	0.392	0.280
Copy Object Sum $\widetilde{+}$	0.431	0.275	0.423	0.169
Frobenius Add $\widetilde{+}$	**0.526**	0.323	0.348	0.264
Frobenius Mult $\widetilde{+}$	0.518	0.158	0.237	0.197
Frobenius Outer $\widetilde{+}$	0.521	0.209	0.326	0.266

TABLE A.5: Spearman ρ scores for baseline and non-linear composition models on the **ELLDIS** dataset, with the highest score for each space in bold and in a box, the second highest result in bold. Results are against an inter-annotator agreement score of 0.584.

ELLDIS	Count Based	Word2Vec	GloVe	FastText
FROB Relational $\overline{\odot}$	0.386	0.406	0.341	0.426
FROB Copy Subject $\overline{\odot}$	0.321	0.348	0.252	0.424
FROB Copy Object $\overline{\odot}$	0.340	0.249	0.326	0.324
FROB Frobenius Add $\overline{\odot}$	0.367	0.361	**0.397**	0.448
FROB Frob. Mult $\overline{\odot}$	0.368	0.217	0.139	0.281
FROB Frobenius Outer $\overline{\odot}$	0.379	0.289	0.347	0.383
FROB Relational $\overline{+}$	0.372	0.427	0.334	0.438
FROB Copy Subject $\overline{+}$	0.316	0.401	0.299	0.442
FROB Copy Object $\overline{+}$	0.336	0.443	0.358	0.432
FROB Frobenius Add $\overline{+}$	0.365	**0.462**	**0.369**	**0.465**
FROB Frobenius Mult $\overline{+}$	0.356	0.418	0.139	0.373
FROB Frobenius Out $\overline{+}$	0.368	**0.460**	0.363	**0.497**
FROB Relational $\widetilde{\odot}$	**0.527**	0.245	0.342	0.238
FROB Copy Subject $\widetilde{\odot}$	0.492	0.310	0.294	0.294
FROB Copy Object $\widetilde{\odot}$	**0.539**	0.015	0.335	0.076
FROB Frobenius Add $\widetilde{\odot}$	0.526	0.271	0.417	0.255
FROB Frobenius Mult $\widetilde{\odot}$	0.486	0.021	0.249	0.062
FROB Frobenius Outer $\widetilde{\odot}$	**0.527**	0.031	0.343	0.123
FROB Relational $\widetilde{+}$	0.523	0.252	0.337	0.229
FROB Copy Subject $\widetilde{+}$	0.491	0.320	0.322	0.291
FROB Copy Object $\widetilde{+}$	0.538	0.244	0.335	0.180
FROB Frobenius Add $\widetilde{+}$	0.526	0.323	0.348	0.264
FROB Frobenius Mult $\widetilde{+}$	0.520	0.172	0.246	0.128
FROB Frobenius Outer $\widetilde{+}$	0.523	0.217	0.338	0.239

TABLE A.6: Spearman ρ scores for linear Frobenius models on the **ELLDIS** dataset.

ELLSIM	Count Based	Word2Vec	GloVe	FastText
Verb Only Vector	0.457	0.583	0.435	0.647
Verb Only Tensor	0.394	0.566	0.443	0.534
Additive	0.700	0.726	**0.696**	**0.741**
Additive Ablate	0.680	**0.745**	**0.696**	0.734
Multiplicative	0.633	0.130	0.367	0.199
Multiplicative Ablate	0.712	0.320	0.309	0.462
Additive Non-Linear	0.681	**0.762**	**0.710**	**0.739**
Multiplicative Non-Linear	0.723	0.355	0.244	0.450
Relational $\overline{\odot}$	0.709	0.668	0.345	0.635
Copy Subject $\overline{\odot}$	0.590	0.524	0.364	0.496
Copy Object $\overline{\odot}$	0.659	0.661	0.213	0.688
Copy Subject Sum $\overline{\odot}$	0.358	0.589	0.397	0.611
Copy Object Sum $\overline{\odot}$	0.414	0.597	0.418	0.614
Frobenius Add $\overline{\odot}$	0.706	0.665	0.371	0.626
Frobenius Mult $\overline{\odot}$	0.662	0.433	0.298	0.463
Frobenius Outer $\overline{\odot}$	0.708	0.620	0.357	0.671
Relational $\overline{+}$	0.725	0.706	0.476	0.682
Copy Subject $\overline{+}$	0.597	0.604	0.417	0.600
Copy Object $\overline{+}$	0.675	0.616	0.403	0.636
Copy Subject Sum $\overline{+}$	0.362	0.590	0.466	0.583
Copy Object Sum $\overline{+}$	0.398	0.653	0.454	0.592
Frobenius Add $\overline{+}$	0.717	0.627	0.491	0.631
Frobenius Mult $\overline{+}$	0.711	0.633	0.360	0.639
Frobenius Outer $\overline{+}$	0.726	0.671	0.482	0.699
Relational $\widetilde{\odot}$	**0.741**	0.439	0.245	0.552
Copy Subject $\widetilde{\odot}$	0.685	0.332	0.316	0.421
Copy Object $\widetilde{\odot}$	0.706	0.263	0.166	0.402
Copy Subject Sum $\widetilde{\odot}$	0.514	0.384	0.164	0.511
Copy Object Sum $\widetilde{\odot}$	0.514	0.217	0.164	0.317
Frobenius Add $\widetilde{\odot}$	**0.737**	0.358	0.290	0.497
Frobenius Mult $\widetilde{\odot}$	0.721	0.074	0.262	0.399
Frobenius Outer $\widetilde{\odot}$	**0.741**	0.165	0.245	0.440
Relational $\widetilde{+}$	0.720	0.623	0.281	0.652
Copy Subject $\widetilde{+}$	0.669	0.542	0.249	0.597
Copy Object $\widetilde{+}$	0.693	0.353	0.225	0.511
Copy Subject Sum $\widetilde{+}$	0.457	0.584	0.451	0.640
Copy Object Sum $\widetilde{+}$	0.457	0.380	0.483	0.530
Frobenius Add $\widetilde{+}$	0.714	0.517	0.302	0.581
Frobenius Mult $\widetilde{+}$	0.732	0.315	0.286	0.497
Frobenius Outer $\widetilde{+}$	0.721	0.399	0.277	0.515

TABLE A.7: Spearman ρ scores for baseline and non-linear composition models on the **ELLSIM** dataset, with the highest score for each space in bold and in a box, the second highest result in bold. Results are against an inter-annotator agreement score of 0.425.

ELLSIM	Count Based	Word2Vec	GloVe	FastText
FROB Relational $\overline{\odot}$	0.692	0.645	0.447	0.624
FROB Copy Subject $\overline{\odot}$	0.592	0.528	0.349	0.486
FROB Copy Object $\overline{\odot}$	0.676	0.071	0.384	0.286
FROB Frobenius Add $\overline{\odot}$	0.694	0.514	0.409	0.482
FROB Frobenius Mult $\overline{\odot}$	0.690	0.096	0.338	0.281
FROB Frobenius Outer $\overline{\odot}$	0.692	0.060	0.450	0.329
FROB Relational $\overline{+}$	0.725	**0.706**	0.476	**0.682**
FROB Copy Subject $\overline{+}$	0.597	0.604	0.417	0.600
FROB Copy Object $\overline{+}$	0.675	0.616	0.403	0.636
FROB Frobenius Add $\overline{+}$	0.717	0.627	**0.491**	0.631
FROB Frobenius Mult $\overline{+}$	0.710	0.633	0.361	0.647
FROB Frobenius Outer $\overline{+}$	0.724	**0.668**	**0.486**	**0.702**
FROB Relational $\widetilde{\odot}$	0.728	0.472	0.316	0.554
FROB Copy Subject $\widetilde{\odot}$	0.658	0.424	0.318	0.484
FROB Copy Object $\widetilde{\odot}$	0.693	-0.003	0.238	0.036
FROB Frobenius Add $\widetilde{\odot}$	0.730	0.390	0.358	0.463
FROB Frobenius Mult $\widetilde{\odot}$	**0.732**	0.012	0.333	0.054
FROB Frobenius Outer $\widetilde{\odot}$	0.728	-0.007	0.315	0.124
FROB Relational $\widetilde{+}$	0.720	0.623	0.281	0.652
FROB Copy Subject $\widetilde{+}$	0.669	0.542	0.249	0.597
FROB Copy Object $\widetilde{+}$	0.693	0.353	0.225	0.511
FROB Frobenius Add $\widetilde{+}$	0.714	0.517	0.302	0.581
FROB Frobenius Mult $\widetilde{+}$	**0.732**	0.318	0.276	0.482
FROB Frobenius Outer $\widetilde{+}$	0.720	0.376	0.281	0.519

TABLE A.8: Spearman ρ scores for linear Frobenius models on the **ELLSIM** dataset.

A.3 Results for sentence encoders and contextualised embeddings

	D2V1	D2V2	ST	IS1	IS2	IS3	IS4	USE
ML2008	0.139	0.192	0.078	0.181	**0.220**	0.149	0.169	0.039
ML2010	0.512	0.447	0.494	0.631	0.492	**0.636**	0.405	0.325
GS2011	0.098	0.102	-0.157	0.297	0.320	**0.324**	0.213	0.094
KS2013	0.193	**0.212**	0.051	0.172	0.032	0.176	-0.021	0.210
KS2014	0.692	0.705	0.546	**0.784**	0.676	0.720	0.586	0.539
MLELLDIS$_{lin}$	0.090	0.233	0.232	0.199	0.224	0.108	0.135	0.105
MLELLDIS$_{res}$	0.089	0.216	0.167	0.228	**0.269**	0.144	0.156	0.154
MLELLDIS$_{abl}$	0.095	0.242	0.159	0.215	0.226	0.141	0.169	0.109
ELLDIS$_{lin}$	0.199	0.227	-0.193	0.347	**0.384**	0.330	0.344	0.269
ELLDIS$_{res}$	0.231	0.253	-0.172	0.344	0.337	0.293	0.248	0.277
ELLDIS$_{abl}$	0.195	0.259	-0.130	0.353	0.357	0.300	0.291	0.240
ELLSIM$_{lin}$	0.593	0.622	0.585	0.779	0.701	0.748	0.641	0.647
ELLSIM$_{res}$	0.698	0.692	0.604	**0.803**	0.749	0.768	0.687	0.680
ELLSIM$_{abl}$	0.652	0.655	0.471	0.782	0.730	0.749	0.682	0.640

TABLE A.9: Spearman ρ scores for sentence encoders on all datasets. D2V1: Doc2Vec1, D2V2: Doc2Vec 2, ST: Skip-Thought, IS1: InferSent 1 (4096), IS2: InferSent 2 (4096), IS3: InferSent 1 (300), IS4: InferSent 2 (300),USE: Universal Sentence Encoder.

	ELMo	**BERT Small**	**BERT Large**
ML2008	**0.166**	0.105	0.030
ML2010$_v$	**0.539**	0.216	0.356
GS2011	0.108	0.187	**0.292**
KS2013	0.243	0.232	**0.349**
KS2014	**0.728**	0.520	0.616
MLELLDIS$_{lin}$	0.193	**0.373**	0.315
MLELLDIS$_{res}$	0.182	0.103	0.342
MLELLDIS$_{abl}$	0.123	0.089	0.193
ELLDIS$_{lin}$	0.232	0.360	**0.368**
ELLDIS$_{res}$	0.210	0.216	0.274
ELLDIS$_{abl}$	0.207	0.197	0.365
ELLSIM$_{lin}$	0.734	0.595	0.580
ELLSIM$_{res}$	**0.779**	0.631	0.647
ELLSIM$_{abl}$	0.703	0.560	0.582

TABLE A.10: Spearman ρ scores for contextualised embeddings on all datasets. While **ELMo** performs very well on the smaller datasets, it is outperformed by **BERT** on the elliptical datasets, although for the latter there is no improvement by moving to the **BERT Large** model.

Appendix B

Evaluation Results for Neural Verb Tensors

This appendix accompanies Chapter 5 and contains tables with all the evaluation results for multiple datasets.

B.1 Results for Neural Tensor Clustering

	Linear Context Window	Subject and Object
MEN	0.282	**0.294**
SimLex-999	0.046	**0.276**
SimVerb$_d$	0.224	**0.275**
SimVerb$_t$	0.183	**0.247**
VerbSim	0.338	**0.493**
ML2008	0.067	**0.088**
ML2010	0.396	**0.557**
GS2011	0.226	**0.302**
KS2013	**0.184**	0.080
KS2014	0.445	**0.664**
ELLDIS	0.341	**0.381**
ELLSIM	0.370	**0.584**

TABLE B.1: Performance of vectors that take both subject and object of the verb as context, as opposed to taking a linear context window as context.

Mid Fusion / α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
MEN	0.157	0.176	0.208	0.206	0.302	0.356	0.383	0.411	**0.418**	0.392	0.371
SimLex-999	0.347	0.356	**0.357**	0.348	0.335	0.317	0.299	0.281	0.271	0.263	0.260
SimVerb$_d$	0.243	0.240	0.241	0.248	0.258	0.268	0.275	0.281	0.286	0.292	**0.297**
SimVerb$_t$	0.235	0.234	0.234	0.237	0.243	0.248	0.253	0.256	0.259	0.261	**0.263**
VerbSim	0.409	0.414	0.416	0.427	0.435	0.447	0.467	0.480	0.500	**0.501**	**0.501**
ML2008	0.103	**0.107**	0.099	0.092	0.090	0.060	0.042	0.051	0.046	0.043	0.054
ML2010	0.429	0.416	0.413	0.400	0.405	0.426	0.465	0.494	0.527	0.547	**0.556**
GS2011	0.187	0.204	0.213	0.257	0.300	0.327	0.349	**0.381**	0.368	0.341	0.341
KS2013	**0.181**	0.167	0.151	0.092	0.079	0.085	0.068	0.070	0.071	0.077	0.105
KS2014	0.514	0.497	0.489	0.478	0.498	0.529	0.567	0.595	0.633	0.655	**0.664**
ELLDIS	0.298	0.322	0.339	0.405	0.439	0.447	0.441	**0.456**	0.439	0.393	0.393
ELLSIM	0.422	0.406	0.400	0.395	0.417	0.449	0.488	0.519	0.557	0.576	**0.587**

TABLE B.2: Results for Middle Fusion of subject/object vectors on all datasets. The results show that the vectors with just the object as context perform the best in most cases.

Late Fusion / α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
MEN	0.157	0.155	0.198	0.245	0.263	0.304	0.320	0.349	0.351	0.368	**0.371**
SimLex-999	0.347	**0.348**	**0.348**	0.345	0.341	0.338	0.329	0.314	0.299	0.278	0.260
SimVerb$_d$	0.243	0.256	0.266	0.275	0.284	0.292	0.298	**0.300**	**0.300**	0.299	0.297
SimVerb$_t$	0.235	0.243	0.251	0.258	0.263	0.267	**0.269**	**0.269**	**0.269**	0.267	0.263
VerbSim	0.409	0.426	0.441	0.455	0.466	0.474	0.489	0.489	0.495	0.499	**0.501**
ML2008	**0.103**	0.091	0.098	0.098	0.073	0.069	0.077	0.061	0.058	0.067	0.054
ML2010	0.429	0.451	0.469	0.482	0.505	0.516	0.526	0.537	0.548	0.553	**0.556**
GS2011	0.187	0.210	0.220	0.228	0.239	0.256	0.285	0.320	0.314	**0.344**	0.341
KS2013	**0.181**	0.177	0.175	0.149	0.138	0.124	0.121	0.109	0.085	0.081	0.105
KS2014	0.514	0.542	0.567	0.585	0.612	0.625	0.637	0.646	0.658	0.660	**0.664**
ELLDIS	0.298	0.325	0.331	0.341	0.351	0.367	0.375	0.403	0.388	**0.408**	0.393
ELLSIM	0.422	0.449	0.471	0.488	0.514	0.532	0.549	0.561	0.577	0.580	**0.587**

TABLE B.3: Results for Late Fusion of subject/object vectors on all datasets. The results show that the vectors with just the object as context perform the best in most cases.

Middle Fusion / α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
MEN	0.527	**0.557**	0.552	0.544	0.514	0.506	0.538	0.525	0.496	0.479	0.478
SimLex-999	0.337	**0.340**	0.337	0.324	0.312	0.298	0.277	0.252	0.232	0.219	0.210
SimVerb$_d$	0.279	0.283	0.288	**0.291**	0.284	0.274	0.262	0.245	0.234	0.231	0.225
SimVerb$_t$	0.217	0.219	0.223	0.227	**0.230**	**0.230**	0.226	0.219	0.213	0.210	0.209
VerbSim (130)	0.426	0.436	0.441	0.463	0.494	0.521	**0.550**	0.545	0.535	0.511	0.498
ML2008	0.109	0.098	0.099	0.094	0.081	0.059	0.069	0.075	0.070	0.104	**0.124**
ML2010	0.575	0.562	0.573	0.588	**0.589**	0.569	0.567	0.568	0.569	0.577	0.578
GS2011	0.256	0.262	0.267	0.285	0.323	0.373	0.367	0.368	**0.387**	0.365	0.371
KS2013	0.195	0.181	0.178	0.171	0.160	0.159	0.195	0.187	0.182	0.195	**0.199**
KS2014	0.546	0.532	0.541	0.561	0.590	0.606	0.628	0.637	0.653	0.671	**0.688**
ELLDIS	0.435	0.425	0.435	**0.466**	0.450	0.457	0.422	0.391	0.389	0.396	0.385
ELLSIM	0.508	0.499	0.479	0.495	0.523	0.546	0.569	0.583	0.598	0.621	**0.639**

TABLE B.4: Results for Middle Fusion of the subject/object verb matrices. The results show the highest performance per fusion value of α, but considering clusters of either nouns, subject, or objects, with cluster centroids taken from the 500, 1000, or 2000 most frequent words, with the number of centroids ranging from 2 to 10.

Late Fusion / α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
MEN	0.527	0.581	0.563	0.576	0.585	0.562	0.560	**0.589**	0.563	0.514	0.478
SimLex-999	**0.337**	0.332	0.325	0.314	0.301	0.287	0.275	0.263	0.247	0.226	0.210
SimVerb (Dev)	0.279	0.281	**0.284**	0.283	0.279	0.275	0.268	0.259	0.248	0.237	0.225
SimVerb (Test)	0.217	0.225	0.232	0.237	**0.240**	**0.240**	0.239	0.234	0.228	0.219	0.209
VerbSim (130)	0.426	0.448	0.473	0.493	0.511	0.512	0.518	**0.520**	0.516	0.507	0.498
ML2008	0.109	0.112	0.117	0.117	0.105	0.089	0.100	0.110	0.111	0.113	**0.124**
ML2010	0.575	0.589	0.616	0.627	**0.638**	0.632	0.633	0.621	0.610	0.593	0.578
GS2011	0.256	0.288	0.283	0.320	0.352	0.366	0.377	0.388	0.392	**0.399**	0.371
KS2013	0.195	0.181	0.193	0.186	0.196	0.201	**0.218**	0.205	0.195	0.188	0.199
KS2014	0.546	0.574	0.618	0.644	0.666	0.683	0.694	**0.695**	0.692	0.693	0.688
ELLDIS	0.435	0.459	0.471	0.501	**0.516**	0.467	0.472	0.463	0.454	0.449	0.385
ELLSIM	0.508	0.531	0.551	0.579	0.602	0.614	0.625	0.630	0.640	**0.643**	0.639

TABLE B.5: Results for Late Fusion of the subject/object verb matrices. The results show the highest performance per fusion value of α, but considering clusters of either nouns, subject, or objects, with cluster centroids taken from the 500, 1000, or 2000 most frequent words, with the number of centroids ranging from 2 to 10.

B.2 Results for Composition Models

B.2.1 Baseline results

	ML2008	**ML2010**
Verb Only Vector	0.067	0.396
Verb Only Tensor	-0.028	0.390
Additive	0.107	**0.541**
Multiplicative	0.171	0.468
Kronecker	0.076	0.403
Categorical	**0.192**	0.511

TABLE B.6: Results for composition models on intransitive datasets **ML2008** and **ML2010**.

	GS2011	**KS2013**	**KS2014**
Verb Only Vector	0.226	0.184	0.445
Verb Only Tensor	0.203	0.151	0.452
Additive	0.187	0.181	**0.672**
Multiplicative	0.110	0.136	0.394
Relational $\bar{\cdot}$	0.248	0.150	0.493
Copy Subject $\bar{\cdot}$	0.200	0.160	0.432
Copy Object $\bar{\cdot}$	**0.323**	0.183	0.501
Frobenius Add $\bar{\cdot}$	0.279	0.188	0.511
Frobenius Mult $\bar{\cdot}$	0.175	0.081	0.405
Frobenius Outer $\bar{\cdot}$	0.272	0.181	0.452
Relational $\tilde{\cdot}$	0.199	0.209	0.530
Copy Subject $\tilde{\cdot}$	0.205	0.210	0.390
Copy Object $\tilde{\cdot}$	0.059	**0.270**	0.140
Frobenius Add $\tilde{\cdot}$	0.185	0.281	0.389
Frobenius Mult $\tilde{\cdot}$	-0.031	0.061	0.042
Frobenius Outer $\tilde{\cdot}$	0.107	0.210	0.057

TABLE B.7: Results for composition models on transitive sentence datasets **GS2011**, **KS2013**, and **KS2014**.

	MLELLDIS
Verb Only Vector	0.067
Verb Only Tensor	-0.028
Additive	0.085
Multiplicative	0.005
Multiplicative $\odot$	0.070
Multiplicative $+$	0.163
Additive $\odot$	0.090
Additive $+$	0.101
Kronecker $\odot$	0.111
Kronecker $+$	0.130
Categorical $\odot$	0.154
Categorical $+$	**0.204**

TABLE B.8: Results for composition models on the intransitive elliptical dataset **MLELLDIS**.

	ELLDIS	**ELLSIM**
Verb Only Kronecker	0.341	0.372
Verb Only Tensor	0.266	0.426
Additive	0.241	0.651
Additive Ablate	0.286	**0.671**
AdditiveNL	0.308	**0.671**
Multiplicative	0.153	0.180
Multiplicative Ablate	0.064	0.428
MultiplicativeNL	-0.031	0.404
Relational $\overline{\odot}$	0.170	0.499
Copy Subject $\overline{\odot}$	0.129	0.496
Copy Object $\overline{\odot}$	0.257	0.523
Copy Subject Sum $\overline{\odot}$	0.255	0.496
Copy Object Sum $\overline{\odot}$	0.345	0.547
Frobenius Add $\overline{\odot}$	0.192	0.534
Frobenius Mult $\overline{\odot}$	0.098	0.453
Frobenius Outer $\overline{\odot}$	0.257	0.520
Relational $\overline{+}$	0.318	0.645
Copy Subject $\overline{+}$	0.241	0.583
Copy Object $\overline{+}$	**0.368**	0.587
Copy Subject Sum $\overline{+}$	0.312	0.480
Copy Object Sum $\overline{+}$	0.362	0.542
Frobenius Add $\overline{+}$	0.321	0.646
Frobenius Mult $\overline{+}$	0.125	0.516
Frobenius Outer $\overline{+}$	0.309	0.628
Relational $\widetilde{\odot}$	-0.010	0.463
Copy Subject $\widetilde{\odot}$	-0.014	0.383
Copy Object $\widetilde{\odot}$	-0.109	0.191
Frobenius Add $\widetilde{\odot}$	0.040	0.404
Frobenius Mult $\widetilde{\odot}$	-0.104	0.052
Frobenius Outer $\widetilde{\odot}$	-0.088	0.126
Relational $\widetilde{+}$	0.298	0.522
Copy Subject Sum Kron	0.292	0.442
Copy Object Sum Kron	0.247	0.314
Frobenius Add $\widetilde{+}$	0.304	0.466
Frobenius Mult $\widetilde{+}$	0.050	0.201
Frobenius Outer $\widetilde{+}$	0.246	0.286

TABLE B.9: Results for composition models on the transitive elliptical datasets **ELLDIS** and **ELLSIM**.

B.2.2 Fusion results

Categorical	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
ML2008 (Mid)	0.174	0.177	0.174	0.175	0.161	0.163	0.163	0.166	0.166	0.177	**0.183**
ML2008 (Late)	0.174	0.180	0.177	0.180	0.180	0.180	0.183	**0.188**	0.186	0.186	0.183
ML2010 (Mid)	**0.516**	0.507	0.509	0.505	0.511	0.515	0.502	0.500	0.495	0.498	0.496
ML2010 (Late)	0.516	0.531	0.549	**0.550**	0.547	0.549	0.543	0.532	0.518	0.512	0.496

TABLE B.10: The effect of the α Fusion parameter on Middle and Late Fusion of the Categorical composition model for the **ML2008** and **ML2010** datasets using the subject/object verb matrices. The results show that late fusion generally has a more dramatic effect on performance.

GS2011 Middle / α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Relational	0.239	0.233	0.232	0.237	0.241	0.253	0.258	0.267	0.273	0.277	**0.281**
Frobenius Outer	0.348	0.355	**0.368**	0.364	**0.368**	0.357	0.341	0.329	0.331	0.336	0.345
Frobenius Mult	0.070	0.088	0.114	0.136	0.141	0.165	**0.182**	0.160	0.129	0.130	0.132
Frobenius Add	0.370	0.383	0.388	**0.389**	0.384	0.374	0.367	0.370	0.374	0.381	0.382
Copy Subject	**0.427**	0.424	0.420	0.400	0.380	0.354	0.347	0.335	0.331	0.337	0.344
Copy Object	0.175	0.183	0.204	0.235	0.262	0.289	0.314	0.326	0.341	0.355	**0.361**
Copy Argument Sum	0.303	0.319	0.333	0.342	**0.349**	0.335	0.315	0.308	0.305	0.301	0.299
Copy Argument	0.427	0.431	**0.440**	0.431	0.429	0.408	0.389	0.377	0.368	0.362	0.361
Copy Argument Inverse Sum	0.229	0.229	0.228	0.243	0.266	0.275	0.273	0.269	0.265	0.271	**0.280**
Copy Argument Inverse	0.175	0.171	0.181	0.219	0.267	0.288	0.314	0.331	0.335	0.338	**0.344**
Cat Argument	**0.533**	**0.533**	0.528	0.515	0.489	0.459	0.427	0.402	0.389	0.382	0.382
Cat Argument Inverse	0.263	0.265	0.277	0.302	0.330	0.350	0.363	0.377	0.390	0.401	**0.409**

TABLE B.11: The effect of the α Fusion parameter on Middle Fusion of the composition models for the **GS2011** dataset using the subject/object verb matrices.

GS2011 Late / α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Relational	0.239	0.261	0.274	0.274	0.275	0.282	**0.286**	**0.286**	0.284	0.282	0.281
Copy Subject	0.427	0.445	0.459	**0.465**	**0.465**	0.452	0.434	0.416	0.394	0.369	0.344
Copy Object	0.175	0.207	0.244	0.279	0.305	0.331	0.350	**0.364**	0.361	0.363	0.361
Frobenius Outer	0.348	0.378	0.398	0.409	**0.411**	0.409	0.406	0.397	0.384	0.364	0.345
Frobenius Mult	0.070	0.092	0.112	0.131	0.150	**0.161**	0.159	0.150	0.147	0.139	0.132
Frobenius Add	0.370	0.401	0.427	0.442	0.449	**0.451**	0.445	0.430	0.413	0.399	0.382
Copy Argument Sum	0.303	0.329	0.357	0.375	**0.383**	0.378	0.362	0.353	0.337	0.315	0.299
Copy Argument	0.427	0.449	0.467	0.474	**0.477**	0.467	0.450	0.429	0.411	0.384	0.361
Copy Argument Inverse Sum	0.229	0.252	0.287	0.311	0.334	0.344	**0.352**	0.345	0.325	0.303	0.280
Copy Argument Inverse	0.175	0.203	0.241	0.279	0.312	0.328	0.343	**0.350**	0.348	0.344	0.344
Cat Argument	0.533	0.534	**0.536**	0.528	0.512	0.494	0.470	0.448	0.427	0.402	0.382
Cat Argument Inverse	0.263	0.306	0.347	0.380	0.407	0.427	0.428	**0.429**	0.418	0.418	0.409

TABLE B.12: The effect of the α Fusion parameter on Late Fusion of the composition models for the **GS2011** dataset using the subject/object verb matrices.

KS2013 Mid / α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Relational	0.051	0.052	0.055	0.063	0.065	0.073	0.078	**0.080**	0.077	0.070	0.065
Frobenius Outer	0.146	0.153	0.162	0.184	0.223	0.246	0.260	**0.278**	0.274	0.260	0.255
Frobenius Mult	0.045	0.069	0.090	0.067	0.049	0.028	0.036	0.083	0.137	0.165	**0.187**
Frobenius Add	0.173	0.183	0.200	0.212	0.228	0.249	0.269	0.283	**0.288**	0.287	0.278
Copy Subject	0.210	0.210	0.206	0.208	0.217	0.223	0.235	0.254	0.269	**0.277**	0.272
Copy Object	0.130	0.136	0.153	0.175	0.200	0.230	**0.245**	0.241	0.228	0.221	0.211
Copy Argument Sum	0.088	0.092	0.097	0.110	**0.122**	0.120	0.109	0.097	0.084	0.077	0.068
Copy Argument	0.210	0.209	0.199	0.197	0.195	0.202	0.205	**0.211**	0.207	0.210	**0.211**
Copy Argument Inverse Sum	0.088	0.087	0.089	0.102	**0.107**	0.106	0.092	0.073	0.064	0.056	0.053
Copy Argument Inverse	0.130	0.144	0.170	0.205	0.236	0.263	0.280	0.294	**0.298**	0.285	0.272
Cat Argument	0.363	0.364	0.370	**0.372**	0.365	0.356	0.339	0.316	0.290	0.260	0.226
Cat Argument Inverse	0.250	0.265	0.276	0.285	**0.296**	**0.296**	0.283	0.275	0.269	0.261	0.252

TABLE B.13: The effect of the α Fusion parameter on Middle Fusion of the composition models for the **KS2013** dataset using the subject/object verb matrices.

KS2013 Late / α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Relational	0.051	0.052	0.060	0.062	0.066	0.064	0.068	**0.072**	**0.072**	0.064	0.065
Frobenius Outer	0.146	0.161	0.179	0.197	0.213	0.226	0.237	0.256	**0.263**	0.262	0.255
Frobenius Mult	0.045	0.063	0.079	0.096	0.126	0.151	0.166	0.172	0.177	0.183	**0.187**
Frobenius Add	0.173	0.197	0.222	0.244	0.258	0.272	0.280	**0.287**	0.285	**0.287**	0.278
Copy Subject	0.210	0.232	0.249	0.267	0.285	0.298	**0.305**	0.304	0.296	0.285	0.272
Copy Object	0.130	0.146	0.163	0.183	0.192	0.206	0.211	0.215	**0.217**	**0.217**	0.211
Copy Argument Sum	0.088	0.102	0.115	0.117	0.118	**0.119**	0.117	0.101	0.091	0.079	0.068
Copy Argument	0.210	0.227	0.239	0.247	0.258	**0.260**	0.254	0.245	0.233	0.223	0.211
Copy Argument Inverse Sum	0.088	0.092	0.105	0.111	0.115	0.114	**0.119**	0.107	0.090	0.072	0.053
Copy Argument Inverse	0.130	0.153	0.177	0.199	0.222	0.244	0.254	0.268	0.271	**0.274**	0.272
Cat Argument	0.363	**0.369**	0.365	0.361	0.347	0.334	0.317	0.299	0.278	0.255	0.226
Cat Argument Inverse	0.250	0.262	0.272	0.275	0.282	**0.291**	0.288	0.286	0.273	0.263	0.252

TABLE B.14: The effect of the α Fusion parameter on Late Fusion of the composition models for the **KS2013** dataset using the subject/object verb matrices.

KS2014 Mid / α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Relational	0.454	0.473	0.512	0.547	0.567	0.586	**0.592**	0.585	0.581	0.574	0.567
Frobenius Outer	0.181	0.209	0.244	0.280	0.314	0.371	0.413	0.447	0.462	**0.482**	0.476
Frobenius Mult	0.021	0.014	0.085	0.180	0.246	0.249	0.240	0.285	**0.328**	0.324	0.304
Frobenius Add	0.461	0.473	0.503	0.548	0.596	0.622	**0.623**	0.606	0.583	0.559	0.528
Copy Subject	0.240	0.259	0.277	0.292	0.304	0.322	0.340	0.361	0.368	**0.384**	**0.384**
Copy Object	0.355	0.366	0.418	0.484	0.545	0.585	0.594	**0.601**	0.592	0.568	0.558
Copy Argument Sum	0.636	0.645	0.656	0.668	0.673	0.684	0.687	0.676	0.679	**0.689**	0.688
Copy Argument	0.240	0.271	0.331	0.393	0.455	0.517	0.552	0.563	**0.565**	0.558	0.558
Copy Argument Inverse Sum	0.648	0.655	0.660	0.664	0.669	**0.672**	0.667	0.664	0.664	0.651	0.641
Copy Argument Inverse	0.355	0.385	0.440	0.480	0.515	**0.519**	0.497	0.464	0.436	0.406	0.384
Cat Argument	0.606	0.598	0.610	0.637	0.677	0.716	0.729	**0.737**	0.723	0.717	0.718
Cat Argument Inverse	0.488	0.514	0.549	0.595	0.637	**0.671**	0.670	0.669	0.658	0.651	0.645

TABLE B.15: The effect of the α Fusion parameter on Middle Fusion of the composition models for the **KS2014** dataset using the subject/object verb matrices.

KS2014 Late / α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Relational	0.454	0.494	0.518	0.542	0.561	0.570	0.576	0.583	**0.584**	0.580	0.567
Frobenius Outer	0.181	0.230	0.291	0.325	0.348	0.382	0.405	0.427	0.455	0.471	**0.476**
Frobenius Mult	0.021	0.077	0.142	0.194	0.239	0.273	0.297	0.308	**0.311**	0.306	0.304
Frobenius Add	0.461	0.485	0.509	0.534	0.553	0.559	**0.566**	0.556	0.552	0.543	0.528
Copy Subject	0.240	0.268	0.275	0.299	0.305	0.329	0.360	0.378	0.389	**0.391**	0.384
Copy Object	0.355	0.408	0.456	0.495	0.540	0.565	0.572	**0.582**	0.580	0.571	0.558
Copy Argument Sum	0.636	0.670	0.695	0.713	0.730	0.734	**0.735**	0.732	0.722	0.707	0.688
Copy Argument	0.240	0.302	0.364	0.440	0.487	0.528	0.550	0.567	0.569	**0.571**	0.558
Copy Argument Inverse Sum	0.648	0.668	0.691	0.709	**0.718**	0.716	0.713	0.707	0.688	0.666	0.641
Copy Argument Inverse	0.355	0.390	0.426	0.457	0.477	**0.485**	0.480	0.463	0.438	0.415	0.384
Cat Argument	0.606	0.650	0.686	0.709	0.727	0.749	0.752	**0.753**	0.746	0.731	0.718
Cat Argument Inverse	0.488	0.538	0.594	0.631	0.664	0.689	**0.691**	0.690	0.681	0.659	0.645

TABLE B.16: The effect of the α Fusion parameter on Late Fusion of the composition models for the **KS2014** dataset using the subject/object verb matrices.

MLELLDIS	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Categorical (Mid) $\odot$	0.088	0.088	0.105	0.112	0.118	0.121	**0.130**	0.125	0.124	0.116	0.106
Categorical (Late) $\odot$	0.088	0.093	0.101	0.110	0.113	**0.120**	0.116	0.116	0.111	0.110	0.106
Categorical (Mid) $+$	0.174	0.177	0.180	0.184	0.190	0.189	0.195	0.201	0.208	0.214	**0.221**
Categorical (Late) $+$	0.174	0.179	0.184	0.190	0.198	0.206	0.210	0.214	0.217	0.221	**0.221**

TABLE B.17: The effect of the α Fusion parameter on Middle and Late Fusion of the composition models for the **MLELLDIS** dataset using the subject/object verb matrices.

ELLDIS Middle / α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Relational $\odot$	0.131	0.132	0.136	0.116	0.141	0.196	0.226	**0.232**	0.228	0.224	0.221
Copy Subject $\odot$	0.081	0.092	0.114	0.147	0.172	**0.186**	0.184	0.147	0.104	0.076	0.076
Copy Object $\odot$	0.105	0.119	0.137	0.161	0.166	0.168	0.173	0.169	0.168	0.180	**0.192**
Copy Subject Sum $\odot$	0.304	0.297	0.292	0.293	0.299	**0.305**	0.300	0.290	0.284	0.284	0.284
Copy Object Sum $\odot$	0.199	0.199	0.200	0.201	0.203	0.194	0.189	0.189	0.194	0.203	**0.210**
Frobenius Add $\odot$	0.106	0.134	0.167	0.198	0.201	0.200	**0.202**	0.198	0.181	0.163	0.159
Frobenius Mult $\odot$	0.100	0.139	0.157	0.163	0.180	**0.192**	0.178	0.142	0.073	0.046	0.058
Frobenius Outer $\odot$	0.139	0.147	0.172	0.199	0.206	0.215	**0.225**	0.198	0.173	0.157	0.166
Cat Argument $\odot$	**0.324**	0.320	0.306	0.295	0.289	0.292	0.293	0.289	0.278	0.268	0.265
Copy Argument $\odot$	0.081	0.079	0.103	0.126	0.136	0.147	0.159	0.172	0.181	0.188	**0.192**
Copy Argument Sum $\odot$	0.304	0.309	**0.314**	0.306	0.296	0.279	0.258	0.236	0.224	0.216	0.210
Relational $+$	**0.342**	0.321	0.305	0.296	0.293	0.289	0.291	0.293	0.297	0.300	0.305
Copy Subject $+$	**0.450**	0.443	0.428	0.405	0.375	0.351	0.325	0.303	0.290	0.284	0.286
Copy Object $+$	0.402	0.404	**0.405**	0.401	0.391	0.375	0.357	0.347	0.341	0.342	0.349
Copy Subject Sum $+$	**0.393**	0.377	0.364	0.357	0.349	0.344	0.334	0.329	0.325	0.332	0.342
Copy Object Sum $+$	0.282	0.276	0.276	0.282	0.284	**0.285**	0.279	0.270	0.268	0.267	0.269
Frobenius Add $+$	0.464	**0.464**	0.461	0.451	0.423	0.389	0.363	0.349	0.342	0.340	0.341
Frobenius Mult $+$	0.167	0.200	0.235	**0.241**	0.216	0.186	0.165	0.134	0.107	0.107	0.121
Frobenius Outer $+$	**0.468**	0.467	0.462	0.449	0.420	0.381	0.338	0.311	0.305	0.309	0.322
Cat Argument $+$	**0.558**	0.551	0.540	0.526	0.500	0.466	0.433	0.413	0.398	0.395	0.392
Copy Argument $+$	0.450	0.453	**0.454**	0.442	0.422	0.400	0.378	0.365	0.356	0.350	0.349
Copy Argument Sum $+$	0.393	0.398	**0.400**	0.396	0.388	0.355	0.328	0.296	0.277	0.269	0.269

TABLE B.18: The effect of the α Fusion parameter on Middle Fusion of the composition models for the **ELLDIS** dataset using the subject/object verb matrices.

ELLDIS Late / α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Relational $\odot$	0.131	0.150	0.168	0.185	0.198	0.208	0.216	0.221	**0.228**	0.227	0.221
Copy Subject $\odot$	0.081	0.087	0.094	0.100	0.106	**0.110**	0.106	0.099	0.092	0.085	0.076
Copy Object $\odot$	0.105	0.121	0.138	0.154	0.166	0.177	0.182	0.189	0.192	**0.192**	0.192
Copy Subject Sum $\odot$	0.304	0.316	0.329	0.339	0.341	0.346	**0.346**	0.337	0.323	0.305	0.284
Copy Object Sum $\odot$	0.199	0.220	0.239	0.253	0.261	**0.263**	0.258	0.249	0.235	0.222	0.210
Frobenius Add $\odot$	0.106	0.121	0.134	0.148	0.155	0.160	**0.163**	0.161	0.162	0.161	0.159
Frobenius Mult $\odot$	0.100	0.110	0.111	0.115	0.115	**0.116**	0.113	0.101	0.090	0.075	0.058
Frobenius Outer $\odot$	0.139	0.150	0.161	0.170	0.178	0.183	**0.184**	0.182	0.178	0.172	0.166
Cat Argument $\odot$	0.324	**0.329**	0.325	0.317	0.307	0.299	0.293	0.287	0.279	0.271	0.265
Copy Argument $\odot$	0.081	0.100	0.122	0.140	0.161	0.171	0.180	0.190	**0.192**	0.192	0.192
Copy Argument Sum $\odot$	0.304	0.315	0.326	0.335	**0.340**	0.331	0.312	0.288	0.264	0.237	0.210
Relational $+$	0.342	0.355	**0.360**	0.352	0.342	0.342	0.337	0.328	0.320	0.313	0.305
Copy Subject $+$	0.450	0.469	**0.477**	0.471	0.458	0.441	0.416	0.383	0.350	0.316	0.286
Copy Object $+$	0.402	0.424	0.441	0.452	**0.456**	0.450	0.434	0.414	0.390	0.371	0.349
Copy Subject Sum $+$	0.393	0.403	0.410	0.412	**0.413**	0.409	0.403	0.390	0.377	0.360	0.342
Copy Object Sum $+$	0.282	0.295	0.307	0.315	**0.316**	0.316	0.309	0.299	0.290	0.280	0.269
Frobenius Add $+$	0.464	0.485	0.498	**0.504**	0.501	0.488	0.463	0.432	0.401	0.373	0.341
Frobenius Mult $+$	0.167	0.176	0.184	0.190	**0.193**	0.182	0.169	0.157	0.140	0.132	0.121
Frobenius Outer $+$	0.468	0.477	**0.481**	0.476	0.463	0.445	0.425	0.399	0.373	0.346	0.322
Cat Argument $+$	0.558	0.557	**0.559**	0.546	0.527	0.505	0.481	0.457	0.434	0.412	0.392
Copy Argument $+$	0.450	0.471	0.484	**0.490**	0.486	0.476	0.456	0.429	0.401	0.374	0.349
Copy Argument Sum $+$	0.393	0.409	0.420	**0.428**	0.424	0.403	0.383	0.360	0.330	0.299	0.269

TABLE B.19: The effect of the α Fusion parameter on Late Fusion of the composition models for the **ELLDIS** dataset using the subject/object verb matrices.

ELLSIM Mid / α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Relational $\odot$	0.405	0.416	0.439	0.484	0.519	**0.525**	0.521	0.518	0.516	0.517	0.516
Copy Subject $\odot$	0.071	0.085	0.120	0.145	0.143	0.114	0.094	0.108	0.144	0.158	**0.162**
Copy Object $\odot$	0.204	0.240	0.276	0.283	0.264	0.240	0.240	0.257	0.274	0.285	**0.292**
Copy Subject Sum $\odot$	0.272	0.273	0.274	0.277	0.282	0.299	0.326	0.361	0.399	0.432	**0.454**
Copy Object Sum $\odot$	0.378	0.381	0.387	0.399	0.414	0.436	0.458	0.475	0.488	0.495	**0.500**
Frobenius Add $\odot$	0.252	0.273	0.308	0.329	0.336	0.331	0.326	0.351	0.368	0.373	**0.384**
Frobenius Mult $\odot$	-0.035	-0.018	0.016	0.107	0.141	0.147	0.156	0.199	0.223	0.247	**0.254**
Frobenius Outer $\odot$	0.115	0.116	0.128	0.148	0.151	0.134	0.129	0.148	0.184	**0.185**	0.177
Cat Argument $\odot$	0.237	0.253	0.262	0.278	0.296	0.318	0.334	0.350	0.369	0.390	**0.412**
Copy Argument $\odot$	0.071	0.182	0.261	0.312	0.331	**0.336**	0.331	0.324	0.315	0.308	0.292
Copy Argument Sum $\odot$	0.272	0.323	0.383	0.444	0.488	**0.504**	0.502	0.502	0.502	0.502	0.500
Relational $+$	0.605	0.608	0.615	0.615	**0.616**	0.615	0.613	0.613	0.613	0.613	0.613
Copy Subject $+$	0.297	0.305	0.326	0.355	0.390	0.426	0.452	0.459	**0.460**	0.458	0.454
Copy Object $+$	0.416	0.427	0.452	0.490	0.515	0.537	**0.546**	0.543	0.536	0.535	0.533
Copy Subject Sum $+$	0.640	0.637	0.636	0.635	0.636	0.637	0.648	0.658	0.668	0.678	**0.688**
Copy Object Sum $+$	0.647	0.647	0.649	0.652	0.660	0.670	0.679	0.690	0.696	0.704	**0.710**
Frobenius Add $+$	0.468	0.485	0.519	0.559	0.595	0.610	**0.612**	0.605	0.595	0.587	0.588
Frobenius Mult $+$	0.260	0.256	0.268	0.263	0.257	0.257	0.288	0.322	0.355	0.386	**0.402**
Frobenius Outer $+$	0.340	0.358	0.381	0.396	0.405	0.416	**0.425**	0.413	0.402	0.404	0.408
Cat Argument $+$	0.587	0.583	0.594	0.616	0.654	0.685	0.702	0.707	0.708	0.708	**0.708**
Copy Argument $+$	0.297	0.319	0.367	0.427	0.480	0.511	0.523	0.526	0.527	0.531	**0.533**
Copy Argument Sum $+$	0.640	0.650	0.665	0.686	0.707	0.718	**0.721**	0.719	0.717	0.714	0.710

TABLE B.20: The effect of the α Fusion parameter on Middle Fusion of the composition models for the **ELLSIM** dataset using the subject/object verb matrices.

ELLSIM Late / α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Relational $\odot$	0.405	0.432	0.454	0.471	0.491	0.508	0.520	0.527	**0.530**	0.524	0.516
Copy Subject $\odot$	0.071	0.091	0.112	0.138	0.165	0.178	**0.183**	0.181	0.180	0.174	0.162
Copy Object $\odot$	0.204	0.233	0.266	0.298	0.324	0.335	**0.337**	0.332	0.323	0.309	0.292
Copy Subject Sum $\odot$	0.272	0.309	0.344	0.379	0.414	0.443	0.461	**0.473**	0.472	0.465	0.454
Copy Object Sum $\odot$	0.378	0.409	0.437	0.464	0.486	0.503	0.515	0.519	**0.519**	0.512	0.500
Frobenius Add $\odot$	0.252	0.294	0.322	0.347	0.369	0.387	0.397	**0.402**	0.400	0.394	0.384
Frobenius Mult $\odot$	-0.035	0.035	0.077	0.123	0.162	0.181	0.203	0.215	0.231	0.247	**0.254**
Frobenius Outer $\odot$	0.115	0.178	0.186	0.192	0.203	0.209	**0.213**	0.211	0.206	0.195	0.177
Cat Argument $\odot$	0.237	0.279	0.315	0.348	0.383	0.402	0.413	**0.415**	0.415	0.415	0.412
Copy Argument $\odot$	0.071	0.126	0.185	0.245	0.290	0.314	0.327	**0.330**	0.324	0.311	0.292
Copy Argument Sum $\odot$	0.272	0.316	0.367	0.420	0.465	0.498	0.520	**0.529**	0.525	0.514	0.500
Relational $+$	0.605	0.627	0.644	0.657	0.662	**0.664**	0.661	0.658	0.650	0.633	0.613
Copy Subject $+$	0.297	0.329	0.365	0.399	0.427	0.452	0.467	**0.474**	0.472	0.464	0.454
Copy Object $+$	0.416	0.460	0.501	0.540	0.569	0.591	**0.596**	0.590	0.572	0.552	0.533
Copy Subject Sum $+$	0.640	0.658	0.674	0.686	0.694	0.700	**0.705**	0.704	0.702	0.696	0.688
Copy Object Sum $+$	0.647	0.666	0.684	0.695	0.704	0.710	0.717	0.719	**0.719**	0.714	0.710
Frobenius Add $+$	0.468	0.513	0.553	0.586	0.611	0.630	**0.637**	0.633	0.623	0.608	0.588
Frobenius Mult $+$	0.260	0.313	0.353	0.382	0.397	0.407	0.413	**0.418**	0.416	0.412	0.402
Frobenius Outer $+$	0.340	0.428	0.467	**0.478**	0.477	0.475	0.468	0.458	0.445	0.427	0.408
Cat Argument $+$	0.587	0.626	0.666	0.689	0.711	0.729	0.738	**0.740**	0.734	0.726	0.708
Copy Argument $+$	0.297	0.356	0.418	0.478	0.524	0.554	0.569	**0.572**	0.564	0.550	0.533
Copy Argument Sum $+$	0.640	0.676	0.705	0.730	0.748	0.758	**0.759**	0.757	0.749	0.731	0.710

TABLE B.21: The effect of the α Fusion parameter on Late Fusion of the composition models for the **ELLSIM** dataset using the subject/object verb matrices.