Cloudspotting: Visual analytics for distributional semantics

Mariana Montes

2021-07-07

Contents

Abstract

The present study is part of the Nephological Semantics research project at QLVL, which aims to develop tools for large-scale corpus-based semantic analyses. A core aspect of the project involves representing semantic structure with vector space models (VSMs), a computational tool that currently requires a deeper understanding of its inner workings and how its results relate to cognitive theories of meaning.

Count-based VSMs represent words[1] as vectors of co-occurrence frequencies in a multidimensional space (**turney.pantel__2010**; **lenci__2018**). Basically, a word is represented by its association strength to other words. They can be generated at both type- and token-level (**heylen.etal__2012**; **heylen.etal__2015**; **depascale__2019**). At type level, two words are represented as more similar if they are attracted to the same contextual features (e.g. other words) and repelled by the same contextual features. This should allow us to identify semantic fields and other relationships between words, but collapses the full range of contexts of each word into one representation. At the token level, instead, we look at individual *occurrences*, and define them as more similar if the words in their contexts are attracted to and repelled by the same contextual features. This way we should be able to map the internal variation of the behavior of individual words, i.e. their semasiological structure.

Within the larger Nephological Semantics project, this work package is dedicated to the understanding of token-level vector space models as a tool for the study of polysemy. Concretely, we explore a number of parameter settings for the models, i.e. ways of defining the context used to represent each tokens, and their impact on the resulting representation, by means of visual analytics. We used manual annotation of sense tags as a heuristic, but without considering them a golden standard. Instead, we aim to map parameter settings to various semantic phenomena coded in the annotations, such as meaning granularity (e.g. distinguishing homonyms and senses within the homonyms). The vector space models, which take the form of large matrices, can be reduced to two dimensions via different methods, such as t-SNE (**Rtsne2008**; **Rtsne2014**). These coordinates can then be mapped onto a scatterplot, resulting in a variety

[1]The term *word* is used very loosely here to encompass different possible definitions.

of shapes, which we call *clouds*.

The workflow was applied to a set of 32 Dutch nouns, verbs and adjectives exhibiting a range of semantic phenomena. For each of them, 240-320 concordance lines were extracted, annotated and modeled. The combination of parameter settings, some of which included syntactic information, resulted in 200-212 different models. The models were clustered with Partition Around Medoids (**kaufman.rousseeuw_1990**; **R-cluster**) so that a manageable, representative set could be explored in more depth, in particular visualizing their t-SNE representations.

Preliminary results suggest that the shape of the clouds depends on the distinctiveness of the collocational patterns, which may or may not match the sense annotations. Noisier models can smooth over these sharp distinctions, while more refined models emphasize them. More importantly, there is no set of parameters that works across the board.

Acknowledgments

The words in this pages, the thoughts they try to convey, are the result of years of thinking, discussing, studying, learning. My voice weaves them together, but it draws from so many sources that have encouraged my growth, stood by me, fed my curiosity, passion and enthusiasm for everything that makes up this text.

For their support and their ideas, I want to thank my supervisors Dirk Geeraerts, Dirk Speelman and Benedikt Szmrecsanyi. Each of them gave their own piece, building my confidence, encouraging me to keep exploring and learning. My main supervisor, Dirk Geeraerts, deserves a special acknowledgment for all the hours in deep discussion on the complex issue that is *meaning*, and what all this is about after all. I appreciate his patience and his engagement. Every time we talked I left feeling more excited and passionate about the topic, more confident and happy. Hartelijk bedankt.

I must also thank my colleagues from the Linguistics Department at KU Leuven, which at the different stages of my years here have been a lovely company and support system. In particular, I would like to thank the members of the Nephological Semantics project, with whom I shared so much of the excitement and frustrations of our common project.

When I joined the project, I already brought with me my own history and connection to (cognitive) linguistics, corpora, statistics and programming. I honestly wouldn't be where I am today if it weren't for my parents, Miguel and Patricia. They have always supported and fostered my study and my interests, given me the tools to grow, to face new challenges. I know I can, because they believe it too.

Chapter 1

Introduction

This dissertation concerns itself with the application of distributional methods, developed within the field of Computational Linguistics (Section **??**), to lexicological research, in particular the theoretical framework of Cognitive Semantics (Section **??**). In addition, the study makes heavy use of visual analytics (Section **??**).

It is part of a larger project, Nephological Semantics, an even a longer research programme within QLVL (Section **??**).

1.1 Distributional semantics and Computational Linguistics

Distributional semantics is a popular technique in Computational Linguistics, where it originated. [Brief mention of references, but I will elaborate what it's about in the first technical chapter.] It relies on the Distributional Hypothesis, which states a correlation between the distributional properties of words and their meaning (or rather, between differences in distribution and differences in meaning).

There are some differences between the use of this technique in Computational Linguistics and what we'll show in this thesis.

First, Computational Linguistics is typically task-oriented, and tests its models by comparing their results to benchmarks, or gold standards [reference to some of them]. In contrast, we will take manual annotation as a guideline, but not as a ground truth, and we are more interested in learning what the models can tell us about the behaviour of words and in *how* models differ than in their accuracy. [Admittedly, this is in part because there is no best model.]

Second, their distributional models mostly work at type-level and with word forms as units, whereas we will look into token-level models with lemmas as units. Of course, there is work at token-level in computational linguistics (as I will describe in the first chapter) but it is not as popular as the type level. A more recent exception is BERT and family.

Third, since the advent of prediction-based models [Mikolov et al 2013…], word embeddings have become increasingly popular; a number of papers have compared the performance of these and count-based models, with different results, but in practice, neural networks are the norm in NLP. For these studies, we looked into count-based model as a more transparent method, i.e. one in which we can trace the similarities between tokens to the words that co-occur with them and their type-level similarity. As the rest of the dissertation will show, the models are not necessarily as transparent as we thought they would be, but that intuition was still the reasoning behind the preference for count-based models.

[Can I equate Computational Linguistics and NLP?]

1.2 Distributional semantics and Cognitive Semantics

[For the computational audience?]

Cognitive Linguistics is a theoretical framework characterized by (among other principles) an emphasis on meaning (everything can be meaningful), the notion of fuzzy and prototypical categories and an usage-based approach. [Bunch of references!]

These three cornerstones inform our study of distributional models in this dissertation.

First, given that the Distributional Hypothesis that underlies the methodology suggests a correlation between distributional differences and semantic differences, what *kind* of semantic differences are at play? Can we model different semantic dimensions with different distributional properties? What kind of semantic phenomena (specialization, generalization, metaphor, metonymy… or even animacy, concreteness) can be modelled? [This from a semasiological point of view; refer to onomasiological and lectometric studies too :)]

Second, the notion of fuzzy and prototypical categories underlies a sceptical perspective towards the existence of senses as discrete categories [more references] and encourages us to pursue a mechanism that can represent semasiological structure in a non concrete way. To a certain degree, we need discrete entities to talk about them, we need to classify things, and both the sense annotation and the use of clustering algorithms respond to such needs. But we don't take them as a norm; instead, we embrace the presence of noise, of degrees of membership, of partial overlap between solutions and general tendencies. The full dissertation should be read in this light.

Third, the usage-based approach is easily mapped into this bottom-up, empirical, quantitative methodology, within an already established trend in Cognitive Linguistics [references!]. With a few exceptions (we'll see), all the examples will be (randomly?[1]) extracted from the samples taken for the case studies. The arguments exposed in this dissertation are backed up by the data used (which are available in The Cloud) and the specific analyses performed, which also means that their generalization power is limited to that data and analyses. Given that the results don't *look* like they would be specific to this corpus (not in terms of the concrete descriptions of the lemmas, but the methodological and theoretical generalizations), it is likely that they apply to other forms as well, but of course, it's an empirical question :)

1.3 Visual analytics

Brief introduction of the motivation and rationale of the visualization tools (which will get their own chapter.)

1.4 Nephological Semantics

Brief history and description of the project, how it brings the three points above together, and what this PhD shares with previous PhD's/publications within the project or what holes it fills. For this I still have to reread the project description, Stefano's subsection on his PhD and your Chapter 1 (any other suggestions?).

1.5 Structure of the dissertation

The rest of the dissertation will consist of three main parts. The first part will focus on the methodological procedure and choices taken in this study: the workflow to create vector space models (including an introduction to distributional semantics), the visualization tool and clustering algorithms, and the selection of a dataset, its annotation, parameter settings, etc. (Not necessarily in that order.) The second part will focus on the most important findings from the study [i.e. three main points]. The third (and less substantial?) part will round up the dissertation with a general practical guide (tips, tricks and warnings),

[1] I like this idea, because it's (1) more honest and (2) fast, but might not return the best illustrations

suggestions for further research and a conclusion. [Whether this amounts to one or two chapters, I'll still have to see.]

Part I

Visualization tool

Chapter 2

An interface to the world of clouds

In this and the following chapters, we will go through the steps needed to obtain "clouds" from corpora and how to explore them in the visualization tool developed within the Nephological Semantics project. It was originally created by Thomas Wielfart, using D3, a popular Javascript library for data-driven visualization, and then further developed by me, culminating in the present version (**montes.qlvl_2021**).

This section will include the technical description of the workflow, as it pertains to the tool itself and to the processing work made before (the Python module, other Python and R functions), and a sort of manual of how it's used.

Visual analytics aims to integrate statistical data analysis with techniques from information visualization so that human analysts can recognize, interpret and reason about the statistical patterns that the data analysis reveals (**card.etal_1999**). Importantly, a visual analytics approach offers a manipulable, interactive visualization that, unlike static diagrams, enables the exploration of a space of parameter values and modeling outputs.

The following chapters interlace some technical explanations of the methodology with description of the actual steps taken in this research. Chapter **??** offers an introduction to distributional models and everything that a researcher needs to know to go from a corpus to the material needed for the visualization. This abstract description is followed by an explanation of the specific parameters explored in our case studies in Chapter **??**. This information should be enough to understand the usefulness of the visualization, thoroughly described in **??**, which builds on **montes.heylen_Submitted**. Chapter **??** expands the analytical possibilities combining the output of the workflow with HDBSCAN and visualizing it on a Shiny App. Finally, Chapter **??** gives an overview of the annotation procedure. While the step itself was taken before any modeling was performed, it is only necessary for visual evaluation and more relevant to the theoretical insights in the second part of this thesis.

Chapter 3

From corpora to clouds

The main goal of the distributional models discussed in this text is to explore semasiological structure from textual data. The starting point is a corpus, and one of the most tangible outputs is the visual representation as a cloud. In this chapter we will describe how to generate clouds from the raw, seemingly indomitable ocean of corpora.

First, we will describe how token-level vector space models are created. Section **??** will explain count-based models, but this is by no means the only viable path. Other techniques, such as BERT (**BERT**)[1], that can generate vectors for individual instances of a word, can be used for the first stage of this workflow.

Once we have vector representations of individual instances of a lexical item, we need to process them. For visualization purposes, we need to reduce the numerous dimensions of the vectors to a manageable number, such as 2. Section **??** will explore and compare a few alternatives.

The same output that is put through dimensionality reduction for the visualization can also be submitted to other forms of analysis such as clustering algorithms, whose results may even be combined with the visualization. We will look at HDBSCAN in particular in Chapter **??**.

3.1 A cloud machine

At the core of vector space models, *aka* distributional models, we find the Distributional Hypothesis, which is most often linked to Harris's observation that "difference of meaning correlates with difference of distribution" (**harris_1954**), but also to **firth_1957a** and Wittgenstein. In other words,

[1]See also **devries.etal_2019** for a Dutch version.

items that occur in similar contexts in a given corpus will be semantically similar, while those that occur in different contexts will be semantically different (**jurafsky.martin__2020lenci__2018**). Crucially, this does not imply that we can describe an individual item with their distributional properties, but that comparing the distribution of two items can tell us something about their semantic relatedness (**sahlgren__2006**).

firth__1957a inspired a whole tradition of corpus linguistics which started to look at collocations as part of the semantic description of a lemma. The Cobuild dictionary was the first to integrate collocational information into their entries. The Birmingham school, pioneered by John Sinclair, used co-occurrence frequency information to describe a lexical item (most often word forms, in English; see **sinclair__1991** and; **stubbs__1995**) by the set of those context words most attracted to them. Researchers would normally transform the raw frequencies with association measures such as mutual information (**church.hanks__1989**; **stubbs__1995mcenery.etal__2010**; **gablasova.etal__2017**) or t-score among others (see **gablasova.etal__2017** for an overview), set a threshold (around 3 for mutual information) and rank the context words that survive such threshold.

Count-based vectors basically compare two items by contrasting their list of collocates, but instead of implementing a binary distinction based on an arbitrary threshold, the magnitude of the association strengths will play a role. Rather than measuring the overlap between the traditional collocational profiles, it will depend on what the actual values are.

Distributional models operationalize[2] this idea by representing words as vectors (i.e. arrays of numbers) coding frequency information. Typically, the raw frequency is transformed to some association strength measure, such as pointwise mutual information (**church.hanks__1989**), which compares the frequency with which two words occur close to each other and the expected frequency if the words were independent. For example, Table **??** shows small vectors representing the English nouns *linguistic*, *lexicography*, *research* and *chocolate*, as well as the adjective *computational*, as series of association strengths with a set of lemmas. Empty cells indicate that the word in the row and the word in the column never co-occur in the corpus (given a certain window span).

Table 3.1: Example of type-level vectors.

target	language/n	word/n	flemish/j	english/j	eat/v	speak/v
linguistics/n	4.37	0.99	-	3.16	-	0.41
lexicography/n	3.51	2.18	-	2.19	-	2.09

PMI values based on symmetric window of 10; frequency data from GloWbE.

[2]See **stefanowitsch__2010** for a discussion on definitions and operationalizations of *meaning* and their relationship to an overarching theoretical notion of *meaning*.

target	language/n	word/n	flemish/j	english/j	eat/v	speak/v
computational/j	1.6	0.08	-	-1	-	-1.8
research/n	0.2	-0.84	0.04	-0.5	-0.68	-0.38
chocolate/n	-1.72	-0.53	1.28	-0.73	3.08	-1.13

PMI values based on symmetric window of 10; frequency data from GloWbE.

Each row is a vector coding the distributional information of the lemma it represents. By *lemma* we refer to the combination of a stem and a part of speech, e.g. *chocolate/n* covers *chocolate*, *chocolates*, *Chocolate*... How we define the unit is a decision we must make when we build a model (see Chapter **??**); in computational linguistic research, this unit is often a word form, and the difference between using word forms or lemmas varies drastically depending on the language of the corpus. More importantly, we must remember that the vectors represent a distributional profile of the lemma, which is automatically extracted from a corpus, not of a meaning (**bolognesi_2020**).

Table **??** offers a brief example in which semantically similar lemmas (e.g. *linguistics* and *lexicography*) have similar vectors, while semantically different lemmas (e.g. *linguistics* and *chocolate*) have different vectors.

These are type-level vectors: each of them aggregates over all the instances of a given lemma, e.g. *linguistics*, to build an overall profile. As a result, it collapses the internal variation of the lemma, i.e. its semasiological structure. In order to uncover such information, we need to build vectors for the individual instances or tokens, relying on the same principle: items occurring in similar contexts will be semantically similar. For instance, we might want to model the three (artificial) occurrences of *study* in (1) through (3), where the target item is in italics.

(1) Would you like to *study* lexicography?
(2) They *study* this in computational linguistics as well.
(3) I eat chocolate while I *study*.

Given that, at the aggregate level, a word can co-occur with thousands of different words, type-level vectors can include thousands of values. In contrast, token-level vectors can only have as many values as the individual window size comprises, which drastically reduces the chances of overlap between vectors. In fact, the three examples don't share any item other than the target. As a solution, inspired by **schutze_1998**, we replace the context words around the token with their respective type-level vectors (**heylen.etal_2015**; **depascale_2019**).

For example, we could represent example (1) with the vector for its context word *lexicography*, that is, the second row in Table **??**; example (2) with the sum of the vectors for *linguistics* (row 1) and *computational* (row 3); and example (3) with the vector for *chocolate* (row 5). This not only solves the sparsity issue, ensuring overlap between the vectors, but also allows us to find similarity between (1) and (2) based on the similarity between the vectors for *lexicography* and *linguistics*.

From applying this method we obtain numerical representations of occurrences of a word. We can compare them to each other by calculating pairwise distances, which is at the base of clustering analyses (such as HDBSCAN, see Chapter **??**) and visualization techniques based on dimensionality reduction (section **??**).

However, in order to obtain this result we need to make a number of decisions related, among other things, with how we define *word* and *context* (**bolognesi_2020**). These decisions will be discussed in Chapter **??**.

3.2 Dimensionality reduction

Dimensionality reduction algorithms try to reduce the number of dimensions of a high-dimensional entity while retaining as much information as possible. In Latent Semantic Analysys (LSA) it is used to reduce type-level spaces from thousands of dimensions to a few hundred, and the resulting reduced dimensions have been found to correspond to semantic fields. **vandecruys_2008** tried both SVD and non-negative matrix factorization on type-level BOW-based models with whole paragraphs as windows but they did not bring any improvement.

It could also be used for token-level spaces, but the comparisons discussed in **depascale_2019** indicate that they don't necessarily perform better than non reduced spaces. Besides, as we will see in Chapter **??**, we can also generate spaces that are already in the few hundred dimensions by selecting the first order context words as second order features, which does not underperform in our studies. Skipping the SVD step also implies more transparent vectors, facilitating the understanding of what is going on under the hood, but, as many other paths we have not taken, the application of SVD remains a parameter to explore in the future.

Both dimensionality reduction techniques and neural networks are suggested as ways of condensing very long, sparse vectors (**jurafsky.martin_2020**; **bolognesi_2020**), but if the FOC-based selection of second order dimensions suffices, it is not really needed. Indeed, we might want to compare SVD versions and embeddings with these vectors, but for the purposes of understanding what is going on and obtaining meaningful information for corpora, this is still cool!

The algorithms we will discuss in this section, instead, try to locate different items on a low-dimensional space (e.g. 2D) preserving their distances in the high-dimensional space (e.g. 5000D) as well as possible. The literature up to today tends to go for either multidimensional scaling (MDS) or t-stochastic neighbor embeddings (t-SNE); recently, an interesting alternative called UMAP has been introduced, which we'll discuss shortly.

MDS is an ordination technique, like principal components analysis (PCA). It tries out different low-dimensional configurations aiming to maximize the correlation between the pairwise distances in the high-dimensional space and those in the low-dimensional space: items that are close together in one space should stay close together in the other, and items that are far apart in one space should stay far apart in the other. It can be evaluated via the stress level, the complement of the correlation coefficient: if the correlation between the pairwise distances is 0.85, the stress level is 0.15. Unlike PCA, however, the dimensions are not meaningful *per se*; two different runs of MDS may result in plots that mirror each other while representing the same thing. Nonetheless, the R implementation (in particular, `metaMDS()` of the {vegan} package, see **R-vegan**) rotates the plot so that the horizontal axis represents the maximum variation. In cognitive linguistics literature both metric (**hilpert.correiasaavedra_2017**; **hilpert.flach_2020**; **koptjevskaja-tamm.sahlgren_2014**) and non-metric MDS (**heylen.etal_2015**; **heylen.etal_2012**; **depascale_2019**; **perek_2016**) have been used.

The second technique, t-SNE (**Rtsne2008**; **Rtsne2014**), has also been incorporated in cognitive distributional semantics (**depascale_2019**; **perek_2018**). It is also popular in computational linguistics (**smilkov.etal_2016**; **jurafsky.martin_2020**); in R, it can be implemented with the `Rtsne` function of the homonymous package (**R-Rtsne**). The algorithm is quite different from MDS. For our purposes the crucial point is that it prioritizes preserving local similarity structure instead of the global structure: items that are close together in the high-dimensional space should stay close together in the low-dimensional space, but those that are far apart in the high-dimensional space may be even farther apart in low-dimensional space.

This leads to nice, tight clusters but the distance between them is less interpretable than in an MDS plot (where, in any case, tight clusters are a rare occurrence in these studies). In other words, we can interpret tight groups of tokens as being similar to each other, but we cannot extract meaningful information from the distance between these groups. In addition, it would seem that points that are far away in a multidimensional space might show up close together in the low dimensional space (**oskolkov_2021**). Uniform Manifold Approximation and Projection (**mcinnes.etal_2020**), instead, penalizes this sort of discrepancies. It would be an interesting avenue for further research, but a brief test on the current data did not reveal such great differences between t-SNE and UMAP to warrant the replacement of the technique within the duration of this project. Other apparent advantages such as speed were not observed in the small samples under consideration (in fact, UMAP —or at least its R implementation with the {umap} package (**R-umap**)— was even slower).

In both cases we need to state the desired number of dimensions before running the algorithm —for visualization purposes, the most useful choice is 2. Three dimensions are difficult to interpret if projected on a 2D space, such as a screen (**card.etal_1999wielfaert.etal_2019**). In addition, t-SNE requires setting a parameter called perplexity, which basically sets how many neighbors the preserved local structure should cover.

Chapter 4

Parameter settings

In this chapter I will describe the various parameter settings we have explored: which are the possible decisions, which ones we have set and which were looked at, why. This should be preceded by an explanation of the workflow itself.

4.1 Fixed decisions

There are a number of parameters that were set fixed for the studies in this project, but which are by no means the only possibility. In some cases we rely on previous work to set a fixed solution, while in other cases it was rather a matter of practicality: time and resources are limited.

- Corpus: QLVLNews corpus —language: Dutch, register: newspapers, mode: written, period: 1999-2004
- Unit definition: stem + part of speech (it could have been wordform, just stem, wordform and part of speech, or even include dependency information) See **kiela.clark_2014** for results on English, favoring stemming but not requiring pos tags.
- Cosine distances between vectors See **jurafsky.martin_2020** 105, but **kiela.clark_2014** suggest correlation instead; euclidean distances between models; log transformation of ranks...
- PPMI values (**kiela.clark_2014**; **depascale_2019**) — **vandecruys_2008** distinguishes between local and global PPMI; we use global PPMI
- Window size for second order: 4 (used by depascale I think, did not show very different results from win 10)

4.2 First steps

Both the targets and the first and second order features are lemma/part-of-speech pairs, such as *haak/verb* (the verb *haken* 'to hook/crochet'), *beslissing/noun* (the noun *beslissing* 'decision'), *in/prep* (the preposition *in* 'in'). The features (or context words) can have any part of speech except for punctuation and have a minimum relative frequency frequency of 1 in 2 million (absolute frequency of 227) after discarding punctuation from the token count in the full QLVLNews corpus. There are 60533 such lemmas in the corpus.

(This threshold is more or less arbitrary, but we're assuming that words with a lower frequency won't have a rich enough vectorial representation.)

In the steps between defining corpus and types and obtaining a the token-level vectors, we have two main kinds of parameters to explore. **First-order parameters** influence which context features will be selected from the immediate environment of the target tokens, while the **second-order parameters** influence the shape of the vectors that represent such first-order features.

In order to visualize the tokens, we have performed dimensionality reduction, i.e. a process by which we try to represent relative distances between items in a low-dimensional space while preserving the distances in high-dimensional space as much as possible. This procedure was described in Section **??**.

4.3 First-order selection parameters

We call the immediate context of a token the **first order context**: therefore, first-order parameters are those that influence which elements in the immediate environment of the token will be included in modeling said token. This was made in two stages: one dependent on whether syntactic information was use, and one independent of it.

It goes without saying that the parameter space is virtually unlimited, and decisions had to be made regarding which particular settings would be explored. We tried to keep the parameter settings different enough from each other to have some variation. The decisions were based on a mix of literature (**kiela.clark_2014**), linguistic intuition and generalizations over the annotation of our very targets. As part of the annotation task, the annotators had to select the items in the immediate context that had helped them select the appropriate tag. In order to remove noise from misunderstandings and idiosyncrasy, we only looked at pairs (or trios) of annotators that had agreed with each other and with our final annotation and ranked the context words over which they had agreed. The distance and dependency information of these context words were used to inform some of our decisions below.

On the first stage, the main distinction is made by `BASE`: between bag-of-words (`BOW`) based and dependency-based models (`LEMMAPATH` and `LEMMAREL`). The former are further split by window size (`FOC-WIN`), part-of-speech filters (`FOC-POS`) and whether sentence boundaries are respected (`BOUND`).

`FOC-WIN` **(first order window)** A symmetric window of 3, 5 or 10 tokens to each side of the target was used.
Of course, virtually any other value is possible [add references!]. Windows of 5 and 10 are typical in the literature [sources?], while 3 was enough to capture most of the context words tagged as informative by the annotators.

`FOC-POS` **(first order part-of-speech)** A restriction was placed to only select (common) nouns, adjectives, verbs and adverbs (`lex`) in the surroundings of the token. If no restriction is placed, the value of this parameter is `all`.
Of course, other selections are possible. [add reference] distinguish between `nav`, which only includes common nouns, adjectives, and verbs, and `nav-nap`, which expand the selection to proper nouns, adverbs and prepositions.
A more detailed research on different combinations would be material for further research. As we will see, the `lex` filter is often redundant with the one based on association strength.

`BOUNDARIES` Given information on the limits of sentences (e.g. in corpora annotated for syntactic dependencies), we can exclude context words beyond the sentence of the target (`bound`) or include them (`nobound`).
This parameter seems to be virtually irrelevant. It was thought as a way of leveling the comparison with the dependency-based models, which by definition don't include context words beyond the sentence, but they don't seem to make a difference.

The distinction between BOW- and dependency-based model doesn't rely so much on which context words are selected but on how tailored the selection is to the specific tokens. For example, a closed-class element like a preposition may be distinctive of particular usage patterns in which a term might occur. However, such a frequent, multifunctional word could easily occur in the immediate raw context of the target without actually being related to it. Unfortunately, just narrowing the window span doesn't solve the problem, since it would also drastically reduce the number of context words available for the token and for any other token in the model. In contrast, we could also have context words that are directly linked to the target but separated by many other words in between, and enlarging the window to include them would imply too much noise for this token and for any other token in the model.

A dependency-based model, instead, will only include context words in a certain syntactic relationship to the target, regardless of the number of words in between. The actual selection process takes two forms in our case: by path length and by relationship. The former, which we call `LEMMAPATH`, is similar to a window size but counts the steps in a dependency path instead of slots in a bag-of-words window. The latter, `LEMMAREL`, matches the dependency paths to specific templates inspired by the context words tagged as informative by the annotators.

To exemplify, let's look at (4) and take *herhalen* 'to repeat' as the target.

(4) *De geschiedenis rond Remmelink herhaalt zich.* 'The history around Remmelink repeats itself.'

This is different from say **vandecruys_2008** where the combination of dependency relation and item make up the feature (I think). This requires a (carefully planned) type-level matrix with dependency features, which we didn't get to generate, but it might be interesting for future research, if it indeed is better at distinguishing between nouns with different degrees of concreteness (although apparently this is also achieved by a BOW model with global PPMI, which is what we use...).

`LEMMAPATH` This set of dependency-based models selects the features that enter a syntactic relation with the target with a maximum number of steps.
The possible values we have included are `selection2` and `selection3`, which filter out context words more than two or three steps away, respectively, and `weight`, which gives a larger weight to context words that are closer in the dependency path.
A one-step dependency path is either the head of the target or its direct dependent. Such features are included by both `selection2` and `selection3` and receive a weight of 1 in `weight`. In (4) this includes the subject, *geschiedenis* 'history', and the reflexive pronoun *zich*, which depend directly on it. If the target was *geschiedenis* 'history', *herhalen* 'to repeat', its head, would be selected.
A two-step dependency path is either the head of the head of the target, the dependent of its dependent, or its sibling. Such features are included by both `selection2` and `selection3` and receive a weight of 2/3 in `weight`. In (4) this includes the determiner *de* and the modifier *rond* 'around' directly depending on a *geschiedenis* 'history'.
A three-step dependency path is either the head of the head of the head of the target, the sibiling of the head of its head, the dependent of the dependent of its dependent, or the dependent of a sibling. A typical case of the last path is the subject of a passive construction with a modal, where the target is the verb in participium (*belastingen* 'taxes' in *de belastingen moeten geheven worden* 'the taxes must be levied'). Such features are included in `selection3` but excluded from `selection2` and receive a weight of 1/3 in `weight`. In (4) this corresponds to *Remmelink*, the object of *rond* 'around'.
Features more than 3 steps away from the target are always excluded. While some features four steps away can be interesting, such as passive subjects of a verb with two modals, they are not that frequent and may not be worth the noise included by accepting all features with so many steps between them and the target. To catch those relationships, `LEMMAREL` is a more efficient method. There are no context words more than three steps away from the target in (4).

See **kiela.clark_2014** 24-25 for a discussion of type-level results with a similar approach compared to a small BOW window. I don't think it's really comparable though.

`LEMMAREL` This set of dependency-based models selects the features that enter in a certain syntactic relation with the target. They are tailored to the part-of-speech of the target, and each group expands on the selection of the group before it. The specific selections are listed in Table **??**.

groups	nouns
1	modifiers and determiners of the target, items of which the target is modifier or determiner, a
2	conjuncts of the target (with or without conjunction), objects of the modifier of the target, and
3	objects and modifiers of items of which the target is subject or modifier, subjects and modifie

4.3.1 PPMI weighting

The `PPMI` parameter is taken outside the set of first-order parameters because it can both filter out first-order features and reshape their vector representations. In truth, the choice of **p**ositive **p**ointwise **m**utual **i**nformation (PPMI) over other weighting mechanisms, as well as setting a threshold or not, is already a parameter setting, which in these circumstances is set to PPMI and a threshold of 0. In all cases, the PPMI was calculated based on a 4-4 window (that could also be a variable parameter).

This parameter can take three values. `selection` and `weight` mean that only the first-order features with a PPMI > 0 with the target type are selected, and the rest discarded, while `no` does not apply the filter. The difference between `selection` and `weight` is that the former only uses the value to filter the context features, while the latter also weighs their vectors with that value.

4.3.2 Second-order selection

The selection of second-order features influences the shape of the vectors: how the selected first-order features are represented. While the frequency transformation and the window on which such values were computed could be varied, they were set to fixed values, namely PPMI and 4-4 respectively. The parameters that were varied across, although we don't expect drastic differences between the models, are vector length and part-of-speech.

`SOC-POS` (second order part-of-speech) This parameter can take two values: `nav` and `all`. In the former case, a selection of 13771 lect-neutral nouns, adjectives and verbs made by Stefano is taken as the set of possible second-order features. In the latter, all lemmas with frequency above 227 and any part-of-speech are considered.

`LENGTH` Vector length is the number of second-order features and therefore the dimensionality of the matrices on which the distance matrices are based, although the amount is not all that changes. It is applied after filtering by part-of-speech.
We have selected two values: `5000` and `FOC`. The former includes the 5000 most frequent elements of the possible features, while the latter takes the intersection between the possible second-order-features and the first-order-features, regardless of frequency. With `SOC-POS:all`, `FOC` will include all first-order features of that model, while with `SOC-POS:nav`, only those included in Stefano's selection.
The actual number of dimensions resulting from `FOC` depends on the strictness of the first order filter. This information can be found on the plots that, for each staal, show how many first order context words are left after each combination of first order filters.

kiela.clark_2014 24 say it is not worth it to make it longer than 50k; jurafsky.martin_2020 did mention something of the kind too Does stefano say anything?

4.3.2.1 FOC as SOC

What does it mean to use the same first-order context words as second-order context words?

First, depending on the number of target tokens and the strictness of the filter, there could be a different number of context words, ranging in the hundreds or low thousands.

Second, the context words will be compared based on their co-occurrence with each other. The behaviour of a context word outside the context of the target will be largely ignored: of course, the association strength between two items has to do with their co-occurrence across the whole corpus, as well as their non-co-occurrence, but it will only be included in the second order vector of the first item if the second is also among the first order context words.

4.4 Medoids

The multiple parameters return a huge number of models, and while purely quantitative methods might be able to process and compare them, it is not feasible for a human to look at hundreds of clouds and stay sane enough to make out anything from them. A more efficient –and easier on the human mind– way to approach this is, instead, to look at representative models.

This method requires us to choose a number of medoids beforehand, which is not an easy task. If we wanted the medoids to represent the best clustering solution, we could run the algorithm with different values of k and compare the results with measures such as silhouette width, as suggested by **levshina_2015**. However, that is not necessarily our goal. We want to be able to see as much variation as possible, while keeping the number of different models manageable (i.e. below 9). It is not particularly problematic if these models are redundant, as long as we can ensure that all the phenomena that we are interested in are represented in them.

For example, given a lemma with multiple senses, it might be the case that some models group the tokens of one sense, and others group the tokens of another: we would like to see representatives of both kinds.

There is no guarantee that the method with the best silhouette returns all the variation we are interested in –our goal is, rather, to limit the number of different models we need to examine from the total number, say 200, to a more manageable amount, like 8. In the same terms, there is also no guarantee that when we identify something interesting in a medoid, i.e. an island for a particular usage pattern, all the models in the cluster of that medoid, and only those models, will share that characteristic. In order to check that, we can look at random samples (again, of 8 or 9 models) of each of the clusters and visually compare them to their medoids. This doesn't need to be as thorough an examination as that of the medoids themselves: it suffices to check if the random sample is not too different and seems to share the characteristic of interest. [add example]

In general terms, for the characteristics identified in the case studies that make up this investigation, we can be quite confident that the medoids are representative of the models in their clusters. However, depending on the concreteness of the phenomena, the variation across models, the clarity of the visualization and the wishful thinking that might lurk in the researchers' minds, it might be the case that something found or assessed in a medoid is not shared by the models in its cluster. The comparison needed with the random sample should be fast and honest and is strongly recommended: if the medoids are representative, you can see it in an instant; if they are not, it just takes a bit longer to admit it. It is *not* the same as actually studying and comparing 64 different models.

Chapter 5

NephoVis

In this chapter we will learn how to use the visualization tool to explore and compare token-level vector space models.

As of this moment, the tool can be found at a Github Page, that is, a Github repository that can be rendered as a static website (**montes.qlvl_2021**). It obtains its data from a submodule; an interested user could clone the repository and just modify the path to the data.

The code for the visualization is written in Javascript, making heavy use of the D3.js library, which was designed for beautiful web-based **d**ata-**d**riven visualization. While it is known of its steep learning curve, it can be useful to think of it in terms of R's vectorized approach: it links DOM elements to arrays and manipulates them based on the items' properties.

The main rationale and framework for this visualization tool was developed by Thomas Wielfaert (**wielfaert.etal_2019**); that code can be found here.

The current implementation would not exist without this foundational setup. However, a number of the available features were added later.

The description of the tool will not immediately follow the expected workflow of an user. Instead, we will start with the lowest level, Level 3, which represents individual token-level clouds, zoom out into Level 2, which shows multiple token-level clouds simultaneously, which will make the most abstract level, Level 1, more clear. Afterwards (Section **??**) we will briefly simulate the path an user would take from Level 1 to Level 3. This perspective was also taken in **montes.heylen_Submitted**. Finally, Section **??** goes into the Beta features that require better development and testing, as well as ideas that we might want to implement in the future. In any case, it must be noted that from July 2019 to 2021-07-07 the user and developer have been the same person, with occasional, valuable input from other members of the Nephological Semantics project. The project could certainly benefit from a wider input of suggestions.

5.1 Level 3

Level 3 of the visualization tool shows a zoomable scatterplot in which each glyph represents a token, i.e. an instance of the target lexical item. The name of the model, coding the parameter settings as described in , is indicated on the top. It is possible to map colors and shapes to categorical variables (such as sense labels) and sizes to numerical variables (such as number of available context words) and to select tokens with a given value by clicking on the corresponding legend key.

5.2 Level 2

Level 2 of the visualization is *not* a scatterplot matrix, although it looks like one and the code was inspired by Mike Bostock's example. Instead, it is just an array of small plots next to each other and wrap of easier readability.

Each of them represents a different model and the same basic features from Level 3 are available: color, shape and size coding, selection by clicking and brushing, and finding the context by hovering over the tokens.

Because they are model-dependent, highlighted context and searching tokens by context word are meaningless in this level, where multiple models are being shown simultaneously. The key contribution of this level, next to the superficial visual comparison of the shape of each plot, is the ability to select one or more tokens in a plot and highlighting them in the rest of the plots as well. Thanks to this functionality, the user can compare the relative position of a group of tokens in a model against that in a different model.

5.3 Level 1

Level 1 shows one zoomable scatterplot, similar to Level 3, but with each glyph representing one model, instead of one token. As a reminder of the difference, the default shape in Level 1 is a wye ("Y"), while that in the other levels is a circle. The data represented by this scatterplot is not the distance between tokens anymore, but that between models, as described at the beginning of Section 3. This scatterplot aims to represent the similarity between models and allows the user to select the models to inspect according to different criteria. Categorical variables (e.g. whether sentence boundaries are used) can be mapped to colors and shapes, as shown in Figure 5, and numerical variables (e.g. number of tokens in the model) can be mapped to size. A selection of buttons on the left panel, as well as the legends for color and shape, can be used to filter models with a certain parameter setting. Otherwise, models can be selected by clicking on the glyphs that represent them.

5.4 The full story

The increasing granularity from Level 1 to Level 3 and the manner of access to different functionalities respect the mantra "Overview first, zoom and filter, then details-on-demand" (**shneiderman_1996**). The individual plots in Levels 1 and 3 are literally zoomable; and in all cases it is possible to select items (either models, in Level 1, or tokens, in the other two), for more detailed inspection. Finally, a number of features show details on demand, such as the names of the models in Level 1 and the context of the tokens in the other two levels.

In practice, the user will start with Level 1, the scatterplot of models, and can look for structure in the distribution of the parameters on the plot. For example, color coding may reveal that models with nouns, adjectives, verbs and adverbs as first-order context words are very different from those without strong filters for part-of-speech, while the use of sentence boundaries makes little difference. Depending on whether the user wants to compare models similar or different to each other, or which parameters they would like to keep fixed, they will use individual selection or the buttons to choose models for Level 2. In our case, we click on "Select medoids", which selects the 8 models returned by a partitioning algorithm, which offers a wide range of variation in a manageable number of plots.

In Level 2 the user can already compare the shapes that the models take in their respective plots, the distribution of categories like sense labels, and the number of lost tokens. In addition, the "distance matrix" button offers a heatmap of the pairwise distances between the selected models. In the case of heffen, the restrictive collocational patterns it presents lead to crisp clusters in the visualization and consistent organization across models. However, models with less clearly defined structure may prove harder to understand. In both cases, the brushing and linking functionality highlights whether tokens that are grouped in one model are also grouped in a different model. From here, the user might switch back and forth between Level 2 and Level 3 for a more detailed inspection of the models.

5.4.1 Examining context words

While it is possible to look at the individual context of each token by hovering over them, it loses track of the larger patterns we want to understand. That is the purpose of the frequency tables in levels 2 and 3.

In any given model, tokens might be close together because they share a context word, and/or because their context words are (based on the second-order modelling) similar to each other. First-order parameters are, by definition, directly responsible for the selection of context words that will be used to model each token. Therefore, when inspecting a model, we might want to know which context word(s) pull certain tokens together, or why tokens that we expect to be together are far apart instead. In other words, if each model offers a different perspective on the distributional behavior of a token, we want to understand what informs said perspective.

In Level 3, individual tokens and groups of them may be selected in different ways. Given such a selection, clicking on "Frequency table" will open a table with one row per context word, a column indicating in how many of the selected tokens it occurs, and more columns with pre-computed information (e.g. PMI values).

The following five columns include pre-computed frequency information, such as the raw co-occurrence frequency and PMI value between the context word and the target based on windows of 10 and 4, and raw frequency in the corpus. These values can be interesting if we would like to strengthen or weaken filters for a smarter selection of context words. This particular model uses dependency-based information as well as a PMI threshold of 0 to select context words.

In Level 2, while comparing different models, the frequency table takes a different form. There is still one context word per row, but the number of tokens with which it co-occurs will depend on the model. The columns in this table are all computed by the visualization based on the lists of context words per token per model. Next to the column with the name of the context word, the default table shows one column called "total" and one per model, headed by the corresponding number. The columns for each model match the second column in their Level 3 frequency table: they indicate with how many of the selected tokens the context word co-occurs. The "total" column, in contrast, reveals the union of this selection: with how many of the selected tokens the context word co-occurs in at least one model.

The default table counts how many of the selected tokens co-occur with each of the context words, but it does not use information from other tokens outside the selection, i.e. the cue validity or association strength of the context words for the selected group. For that purpose, a dropdown button in the top left corner of the frequency table offers a small range of transformations, such as odds ratio, Fisher Exact, cue validity, etc. One such option shows the absolute frequencies within and outside the selection, where the green columns count the number of selected tokens that co-occur with each context word, and the white columns count the number of tokens outside of the selection co-occurring with those context words.

5.5 Wishlist

Chapter 6

HDBSCAN

The visual exploration is extremely useful for a thorough, qualitative description of the vector space models. However, such an application can also become an obstacle to a truly systematic, scientific description. I would avoid talking about objectivity: neither of us, individually, can truly be objective, and we should instead strive for a humble admission of our own partiality and a fruitful combination of all our partialities.

When describing a cloud —in particular these clouds that refuse to show clear images, perfect sense disambiguation, distinct clusters—, how can we ensure that what we see will be found by other researchers? How can we make our observations, if not inherently valid, at least reproducible? This is, after all, the goal we strive for when we embark on quantitative methods.

One tool that can help us systematize our observations, such as the tightness or at least existEnce of distinct islands on a plot, is Hierarchical Density-Based Spatial Clustering of Applications (HDBSCAN) (**campello.etal_2013**). This algorithm basically tries to distinguish dense areas separated by less dense areas[1] and allows for noisy data. In other words, unlike traditional hierarchical clustering, it will not try to cluster all of the points in the dataset, but instead may discard those that are too far from everything else. Moreover, in comparison to its non hierarchical counterpart, DBSCAN, it requires only one parameter to be set *a priori*, namely *minPts*.

The *minPts* parameter indicates the minimum size of a dense group of points to be considered a cluster. An isolated dense group of points for which $n < minPts$ will be considered noise. In the case studies described here we have fixed *minPts* to 8, which seems a reasonable size for the smallest clusters, but it would be interested to look more systematically into the effect of lowering this threshold.

[1]For a friendly description of how the algorithm works, the reader is directed to **mcinnes.etal_2016**; for an even friendlier explanation, find the authors on YouTube.

Rising the threshold, on the other hand, would increase the proportion of points that are considered noise, which is already very high.

Like other clustering algorithms and the dimensionality reduction techniques, HDBSCAN can take a token-context matrix as input or a distance matrix. We have used the transformed distance matrix, that is, the same input fed into the t-SNE algorithm, for the `hdbscan()` function of the {dbscan} R package (**R-dbscan**). The output includes, among other things, the cluster assignment, with noise points assigned to a cluster 0, and epsilon values, which can be used as an estimate of density.

HDBSCAN estimates the density of [the area in which we find] a point a by calculating its core distance $core_k(a)$, which is the distance to its k nearest neighbor, k being $minPts - 1$. Then it recalculates the distance matrix by defining a new distance measure, called *mutual reachability*, which is defined as the maximum between the distance between the items $d(a, b)$ and each of their core distances.

$$d_{mreach-k}(a, b) = max(core_k(a), core_k(b), d(a, b))$$

Once the algorithm obtains these distances, it uses a single linkage method to create hierarchical clusters, then using again $minPts$ and other calculations to merge them into the final selection. The `eps` (epsilon) values returned by `hdbscan()` indicate the height, in the single linkeage tree, at which each point was joined to a cluster, and can thus be used as a proxy for its "density".

This is intuitively more clear if we map the clustering solution to colors and the `eps` value to transparency in a t-SNE plot with perplexity 30. For the most part, the results converge, which has two main upsides. In the first place, we have independent confirmation of the structure found by t-SNE, as a different algorithm processing the same input returns compatible output. Second, insofar the HDBSCAN output matches our visual assessment, it can systematize it and render it reproducible.

This wonders notwithstanding, the compatibility between the HDBSCAN output and visual examination is not guaranteed. We might find interesting tokens that are discarded as noise, or structure within a single HDBSCAN cluster. However, it must be noted that this match (or lack thereof) has been assessed only between `dbscan::hdbscan()` with $minPts = 8$ and `Rtsne::Rtsne()` with $perplexity = 30$. It would be an interesting avenue for further research to experiment with other combinations, and of course UMAP output.

Chapter 7

Annotation schema

For these case studies, we selected 34 Dutch lemmas to annotate and model with token level vector spaces, two of which (*herkennen* and *spoor*) were discarded. The selection process will be presented in Section **??**, with a description of the selected items and what we expected their annotation and clouds to look like. A short description of the corpus (**depascale_2019**) will follow. The annotation procedure will be the focus of Section **??**. Bachelor students of Linguistics at KU Leuven (later called *annotators*) were recruited and hired to manually annotate samples of the selected lemmas, and while the administrative procedure itself is not of great interest to the project, a number of practical issues will be discussed: the distribution of tokens, the assignment of tasks (in particular, the graphic interface provided) and the processing/analysis of the data.

7.1 Selection of items

For this case study 34 Dutch lexical items were selected. We aimed to cover a variety of polysemy phenomena, which will be addressed in the specific sections for each part of speech: section **??** for nouns, section **??** for adjectives, and section **??** for verbs. By modeling different parts of speech and different kinds of polysemy, we expected to develop more robust generalizations regarding the parameter settings that best model specific phenomena.

The selection procedure mixed some introspection (thinking of words that could be interesting), looking up lexical resources (going through a tentative list of dictionary entries to figure out what kind of polysemy we could expect) and corpus data (surveying a sample of concordances for evidence of the expected polysemy). While dictionaries were an essential resource to sketch the sense labels the annotators would have to choose from, we also adjusted them to a more manageable granularity. The concordances were also crucial to estimate sense distribution and adjust the granularity of the definitions. We didn't want an overwhelmingly frequent sense to affect the annotators judgement, and very infrequent senses might be hard to model or at least to visualise in the 2D representations. Still, we did allow for some complexity and subtlety in some cases.

In a number of cases, the corpus survey (reading a concordance of 40-50 randomly selected instances) invalidated options that intuitively or according to the dictionary definitions would have conformed to our requirements. When judging such a discrepancy, it is important to take into account the composition of the corpus. The topics addressed in newspapers and the terms used to talk about them are certainly not representative of everyday life or the entirety of language.

We also had cases of adjectives that could be used in adverbial form and were not always properly tagged for part-of-speech, so we had to discard them. We made a difference between cases such as *hoopvol* 'hopeful', which often occurs in predicative contexts with a verb that is not copula (e.g. *ik ben hoopvol gestemd* 'it makes me hopeful', *hij kijkt hoopvol omhoog* 'he looks up hopeful(ly)') but still predicates over an entity, and cases such as *gemiddeld* 'average', which could either predicate over an entity, as in *gemiddelde student* 'average student', or a predicate, as in *zij eet gemiddeld 3 koekjes elke dag* 'she eats in average 3 cookies per day'. Sometimes the incorrectly tagged cases (adverbs tagged as adjectives) were infrequent enough to be dismissed, but in some other cases they were so many we had to discard the lemma as candidate. While the only direct consequence is that a certain potentially interesting lemma couldn't be investigated, this also should be taken into account when relying on the part-of-speech tagger in other steps of the workflow.

The next subsections describe the selected nouns, adjectives and verbs, the QLVLNewsCorpus and the sampling method. This includes a general description of the polysemy phenomena and hypotheses, but the specific definitions, examples and translations can be found in Appendix XX.

7.1.1 The nouns

The 8 nouns all exhibit homonymy and polysemy in at least one of the homonyms.

Three nouns have one frequent, monosemous homonym and a less frequent, polysemous one: *hoop* 'hope/bunch', *spot* 'ridicule/show or spotlight' and *horde* 'horde/hurdle'. The polysemy phenomena are varied: metaphor for *horde* 'hurdle', metaphor/generalization for *hoop* 'heap' and metonymy for *spot* ('short video/spotlight'). In this latter case, the 'spotlight' sense can further be used literally or metaphorically, but this distinction was not included among the definitions given to the annotators.

Four nouns have two polysemous homonyms: *schaal* 'scale/dish or shell', *blik* 'look/tin', *stof* 'substance or fabric or topic/dust', *staal* 'steal/sample'. For *blik*, the frequent homonym ('look') has a concrete sense with a metonymic and a metaphoric extension, while the infrequent one can refer to a material ('tin'), an object made of that material or its content: the distinction is quite clear but might depend on the specificity of the context and be very infrequent. Similarly, *stof* presents one frequent homonym with two concrete, referentially distinct senses ('substance' and 'fabric') andd an abstract one ('topic'), and another with a subtle, context-specificity dependent difference ('dust (in the air)' or 'reducing something to dust, pulverize'). *Schaal* exhibits subtle perspective shifts in one homonym ('scale', e.g. "scale of Celsius" as opposed to "large scale") and refers to different concrete objects with the second ('shell', 'dish', 'scale dish'). Finally, *staal* 'steal' could refer, like *blik* 'tin', to either the material or an object made of it, while the 'sample' homonym is sensitive to construal (sample as evidence, or in general): it's likely to present high confusion and/or a very skewed distribution in both homonyms separately.

- Finally, *spoor* has three homonyms, of which two polysemous: 'footprint or trace/train(line, rail, company)/spur'. This noun was later discarded because it proved too complicated, but the data is available for reanalysis.

7.1.2 The adjectives

The selection of adjectives includes 13 lemmas presenting different kinds of polysemy phenomena.

Three adjectives have a metonymic reading: *hoopvol* 'hopeful', *geestig* 'witty' and *hachelijk* 'dangerous/critical'. For *geestig* and *hoopvol*, one of the senses is anthropocentric, i.e. it's mainly or exclusively applied to people, although such distinction is not made explicit in the definitions of *geestig* but only suggested in the example. The expected frequency of the anthropocentric sense is in both cases much higher than the other one. In *hachelijk*'s case, the difference is a matter of temporal or telic perspective, so probably harder to distinguish, and it's probably more likely that annotators suggest the second sense as an alternative to the first one (assigning the 'critical' interpretation to something potentially dangerous) than the other way around.

Four adjectives have metaphoric readings: *hoekig* 'angulous/clumsy', *dof* 'dull', *heilzaam* 'healthy/beneficial' and *gekleurd* 'colorful, person of color, tainted'. *Heilzaam* has two distinctions, somewhere between metaphoric and specialization: one refers to something specifically/literally healthy, and the other one is broader and less concrete. *Hoekig* and *gekleurd* present three sense distinctions, one of which is particularly concrete and the most frequent and another one explicitly anthropocentric. The third sense distinction has a different quality: synaesthetic for *hoekig* and very much metaphoric for *gekleurd*. Finally, *dof* has all four kinds of senses: concrete, synaesthetic, anthropocentric and abstract.

Three adjectives present some other form of similarity between the readings: *geldig* 'valid', *hemels* 'heavenly' and *gemeen* 'shared/public/mean/serious'. *Geldig* 'valid' and *hemels* 'heavenly' offer two options, one restricted to a specific context and one much broader. The relation between the relative frequencies of those senses are inverted: the specific sense of *geldig* is less frequent than the general one, while in *hemels* it's the other way around. We would expect that the specific sense would not be offered as alternative to the general sense as much as the other way around. The case of *gemeen* is quite complex, involving a number of rather subtle distinctions. The limits between the first and the second one and between the third and the fifth are hard to establish; the fourth sense seems more clear but if the context isn't specific enough it could be easily confused with the fifth. In addition, the senses are not always mutually exclusive, and a certain instance could very well conflate or be ambiguous between two senses.

Finally, we have three more complex adjectives: *heet* 'hot' for different entities and metaphorically, *grijs* 'gray', with metaphorical and metonymical extensions and *goedkoop* 'cheap', with different entities and metaphorically.

Heet 'hot' presents, first, three very concrete senses that differ in perspective: temperatures of different kinds of things. The second half is metaphorical, of which one synaesthetic, one anthropocentric and very specific, and one more abstract and also quite specific. Crucially, there is no exclusive sense tag for idiomatic expressions, which are quite frequent; they are expected to be tagged with the concrete senses (and maybe a comment on their figurative interpretation), but annotators might also use the *geen* tag for those cases.

Grijs presents a very frequent, concrete sense, two specific metonymic extensions, one anthropocentric sense, one rather abstract and another very specific metaphor.

Goedkoop, on the other hand, presents a modest set of 4 sense distinctions: a concrete, prototypical and frequent sense, two perspectival shifts and a clear metaphor.

7.1.3 The verbs

For the verbs, we selected a range of combinations of syntactic and semantic variation:

Four verbs are always trasitive, and the sense distinction is related to the objects they can take: *haten* 'hate', *huldigen* 'honor/hold (attitudes, opinions, stances)', *heffen* 'raise', *herroepen* 'annul (a law)/retract (statement)') *Haten* and *heffen* are probably more easy to distinguish, the former having an anthropocentric distinction (basically, hating people against disliking things, but with possible gray areas in between, depending on how that object is construed) and the latter presenting a rather clear and common metaphor, between physical objects and abstract entities such as taxes being levied. *Huldigen* and *herroepen* instead have slightly more subtle differences, but the former (between honoring someone/something and holding and opinion) is probably stronger and easier to distinguish than the latter, between retracting a statement or annuling a decree (which again could be interpreted differently depending on how the entity is construed, how prototypical it is).

Two of the verbs can be transitive, with a distinction based on the object, or intransitive: *helpen* 'help', *herstructureren* 'restructure'. These verbs are quite subtle and might present a lot of confusion, particularly because the intransitive uses are semantically very similar to one of the transitive cases. For *herstructureren*, one transitive sense and the intransitive one (exemplified with a reflexive...) are more specific, regarding companies and with the connotation that the personnel is being reduced, while the other transitive sense is broader and might be selected in contexts with less specificity. It might also depend on world knowledge (whether the annotators know or can guess that a certain object -or the subject in the intransitive construction- is a company) and how prominent the implication of personnel reduction is. For *helpen*, the distinction between the transitive uses is rather subtle (the "collaboration" sense is exclusive of animate subjects, but that's not explicit in the definitions), so there might be some disagreement in their annotation, but if the intransitive sense is confused with the transitive ones, it should only be with the first one.

Three verbs can be transitive, with a distinction based on the object, or reflexive: *diskwalificeren* 'disqualify', *herhalen* 'repeat', *herinneren* 'remember/remind'. This category initially included *herkennen* 'recognize' but it was discarded. For some verbs this opposition can be interpreted as a specific situation where the object and the subject coincide. This is particularly the case with *diskwalificeren*, where the reflexive argument structure pretty much replicates the transitive senses, in a particular case where someone disqualifies themselves. The possibility to distinguish between the transitive cases, which differ in specificity (sport context against more general, mostly political context), relies instead on the clarity of the context. for *herhalen*, what could be an object in the transitive senses (but probably wouldn't) is the subject in the reflexive, so the distinction should be very clear, while the transitive uses differ in the kind of objects that they take, with certain prototypical nouns (and the possibility of clauses for the second sense) and maybe some borderline cases. Finally, *herinneren* shows both a clear distinction between reflexive and transitive uses and a further argument structure distinction between transitive uses, either with or without an *aan* complement (which might be absent in the restricted context).

Two more verbs can be transitive, intransitive or reflexive, with semantic distinctions within the transitive structure: *harden* 'make/become hard, tolerate', *herstellen* 'heal/repair'. In the case of *harden*, the transitive can be concrete, figurative, or concrete with a different sense and in a specific construction, namely *(niet) te harden*; the intransitive structure is similar to the concrete transitive, but taking its object as subject, and the reflexive is similar to the second transitive. If senses of different argument structures were confused, the intransitive would be with the first and the reflexive with the second. The *(niet) te harden* uses should be easy to isolate, with strong agreement between annotators and high confidence. For *herstellen*, the transitive structure presents three possible senses: one concrete, one figurative but not presented as such, and one abstract that is very subtly different from the second one. The reflexive is very close to the figurative sense and the intransitive is more specific to concrete healing (rather than repairing) and should not be confused with the others.

Finally, we included a verb with semantic distinctions within both the transitive and the intransitive structures: *haken*. It was presented with two transitive senses, two intransitive and one transitive/intransitive. The transitive senses can be both concrete and literal and differ in specificity: one sense refers particularly to making somebody trip. Intransitive uses differ in literality and, while both might occur with *blijven*, only the figurative definition mentions it (apparently restricting it). No figurative options are mentioned for the transitive senses, so if they occur annotators might either tag them as transitive concrete, intransitive figurative, or *geen*. Finally, one sense that can occur as transitive or intransitve (ellided object) is that of 'crochet'; it's so specific that it shouldn't be confused with others and would probably have high confidence.

7.2 Expectations for the annotation

Here a first set of expectations for the annotations of the types has been summarized in six points, but discussion and revision are needed. They are formulated as predictions and followed by suggestions on how to confirm them. Some 'technical' terms are:

- **majority sense**, meaning the sense tag that most of the annotators assigned to a given token.
- **alternative sense/annotation**, meaning a sense tag assigned to a given token, different from the majority sense.
- **(be) confuse(d)**, meaning there is disagreement on the annotation.

7.2.1 For all types

1. **Very specific senses will not be confused with more general senses**. When the majority sense is a very specific one, the only alternative annotation will be *geen*. Concretely:

- *haken 5* and *haken 3* will not be confused with each other nor with other senses.

- *heet 4* should not be confused with other senses.

2. **Metaphor will be easier to identify than metonymy/specialization** If the metaphoric sense is an option distinct from the concrete/literal one, it won't often be confused with the literal counterpart; annotators will agree it's figurative. For metonymy and specialisation, there will be more disagreement and less confidence. Concretely:

- *blik 1.1* will be confused with *1.2* more than with *1.3*.
- *grijs 1* will be confused with *2* and *3* more than with *6*.
- *goedkoop 1* will be confused with *2* and *3* more than with *4*
- adjectives with metaphoric distinctions (*hoekig*, *dof*, *gekleurd*, *heilzaam*) will present less confusion than those with metonymic distinctions (*hachelijk*, *hoopvol*, *geestig*)

3. **Anthropocentric senses will be more easily distinguishable**. If the definition explicitly restricts the application to people, it won't be alternative annotation with other, non anthropo-exclusive senses. Borderline cases, due probably to unspecified context, would have low confidence. Concretely:

- There should be low confusion in *haten*, or at least low confidence in borderline cases
- *heet 5* should not be confused with others
- *grijs 4* should not be confused with other senses (except maybe 3, which is derived from it)
- *gekleurd 2* should not be confused with others
- *hoekig 3* should not be confused with others
- *dof 3* should not be confused with others
- *hoopvol* will present less confusion than *geestig* and they both will present less confusion than *hachelijk*

7.2.2 Only for nouns

4. **Homonyms will not be confused with each other**. When a majority sense is from one homonym, alternative annotations will be of the same homonym or *geen*.

7.3 Only for verbs

The next two predictions overlap, and could explain different verb groupings (sometimes, different argument structure *implies* subject distinction, but which is more prominent?).

5. **Argument structure will be easier to identify than semantic differences.** Senses that differ in argument structure will not be confused with each other (won't be each other's alternatives) as much as senses with the same argument structure but different kinds of objects. The distinction will be probably easier to make with reflexive than with intransitive cases. Concretely: in general, transitive senses will be confused with each other but not with senses of a different argument structure (unless that sense is semantically very similar to that transitive sense). Cases that might generate confusion between different argument structures are:

- gral. trans. *haken* and fig. intrans. *haken* in cases of fig. trans. *haken*
- fig. trans. *harden* and refl. *harden*
- fig./abs. trans. *herstellen* and refl. *herstellen*
- spec. trans. *herstructrureren* and intrans. *herstructureren*
- *helpen 1* and intrans. *helpen*

6. **Senses that require different subjects will be easier to identify than senses that require different objects or prepositional arguments** Senses with different subject restrictions won't be each other alternatives. I'm thinking that subject restrictions are often linked to animacy, while object restriction might be more subtle in these cases. Concretely: in general, senses of any argument structure will not be confused with each other if one takes (mostly) animate subjects and the other one (mostly) inanimate subjects.

7.4 The corpus and samples

The exploration of these samples of concordances also served for the calculation of the number of tokens we would have annotate. Regardless of the actual frequency of the items in the corpus, we extracted a minimum 240 tokens of each type (thinking of 6 batches of 40 tokens), and raised the amount to 280 if any of the senses had a relative frequency below 20% in the sample, to 320 if it was below 10%, and 360 if there were many senses and therefore some had a low frequency.

The sample of tokens was selected almost absolutely randomly. First all the instances of each type were extracted from the corpus; then, for each type as many *files* as tokens we wanted to extract were selected, and from each file I randomly selected one token. Therefore, there are no two instances of the same lemma from the same file in the samples. There were, however, a few duplicates, due to repetition of the same fragment on different dates.

The corpus is a selection of the LeNC and TwNC corpora, which include newspapers articles from Flanders and the Netherlands. This selection, performed by **depascale_2019** with an eye on a lectally balanced corpus, contains 4,614,267 types and 519,996,217 tokens (roughly 520M, 260 from Flanders and 260 from the Netherlands). The articles in the subcorpus were published between 1999 and 2004 in both quality and popular newspapers from both countries.

7.5 Annotation procedure

7.5.1 Assigning batches to annotators

In October 2019, 48 students from the General Linguistics course of the 2nd year of the Bachelor in Linguistics in KU Leuven were recruited to work as annotators. Each of them was tasked with annotating 40 tokens of each of 12 types (at least three nouns, four adjectives and four verbs, plus one of either of the categories), a total of 480 tokens, for which we expected them to work an average of 10 hours, spread over 6 weeks. Students had the option of subscribing to double the number of tokens (and hours, and pay). Both the types and the sets of tokens were assigned randomly, while keeping in mind the part-of-speech distribution. It was the intention to shuffle the samples of each lemma before splitting them into batches, but something went wrong with the code and they were ordered by source.

The annotation involved three compulsory tasks and one normally optional. For each of the tokens, the annotators had to:

1. assign a sense from a predefined set of definitions, as shown in Appendix XX. If none of the senses was satisfactory, they may choose a "None of the above" option;
2. express the confidence of their decision in a Likert scale of 6 values;
3. identify the words of the context that helped them assign a sense, comprising 15 tokens to the left and to the right of the target, disregarding sentence boundaries but respecting those of the article;
4. if they couldn't assign a sense, they had to explain why. If they did assign one, they still had the option of adding extra information or thoughts on the annotation process, but it was not compulsory.

7.5.2 Annotation tool/interface

Since entering textual information in a spreadsheet can easily lead to typos and inconsistencies and, furthermore, annotating the relevant context words (cues) is challenging in such a tool, a user-friendly visual interface was designed that transforms button-output into a json file with all the information required.

The interface had a menu of types and, for the selected type, two tabs: an overview of the concordance lines and an annotation workspace. In the annotation workspace, they could read each line individually, click on the button corresponding to the sense they wanted to assign, rate their confidence with star rating, click on the words they found useful and enter any other comments. The overview section didn't only let them see the whole set of tokens to analyze, but the target items changed color once they had been annotated and were themselves links to the Annotation tab for their concordance lines.

The interface was rendered as a webpage via Github Pages, but only processed the information: the annotators had to download (and if necessary upload) their progress as a JSON file and eventually send it by mail.

The goals of the interface were twofold. First, it would reduce typos and inconsistencies for values that should be straightforward and present little variation: it was much faster to design the interface than it would have been to check the typos in 480 tokens times 40 annotators. Second, it would make the annotation experience simpler and even more pleasant, letting the annotators focus their energies on the lexicographical task itself rather than in technicalities.

This is particularly evident for the task of selecting the relevant context words. With a spreadsheet, it would be either necessary to have separate rows per context words and annotate all of them (to make sure you are not forgetting any) or make a list of items in a cell of the row of the relevant concordance. That list would need to have something truly identifying of the context word, therefore not the form or the lemma but the position (since a same item could occur more than once in the context and not always have the same relevance), and counting them reliably would be time consuming and prone to errors. Clicking on the words so that the program itself lists the position of the relevant words, also making it visible which words were selected, solves both the easiness and reliability issues.

7.5.2.1 Known issues

The interface did have some issues, the consequences of which affect the output.

One technical issue was a bug in the code of the annotation, for which context words selected by an annotator might be replaced by other context words of the same wordform, but in a previous position. Once that bug was found, the annotators were warned, but not all of them necessarily checked their previous annotation very thoroughly. In any case, this only affects wordforms that occur more than once in the same concordance (which is not very often) and could be cleaned with some reasoning.

Another issue had to do with the format of the corpus, and could have been dealt with better. On the one hand, different sentences in the concordance were indiciated with a `<sentence></sentence>` but had no impact in the rendering of the concordance, they were just replaced by empty spaces. They could've been replaced by `<p></p>` tags. On the other hand, at some point of the corpus processing (before we had access to it), someone must have replaced all *&* with *and*, so that HTML entities like *&quot;*, which would've translated into a quotation mark ", are rendered as *andquot;*, which was extremely confusing for the annotators, especially in already complicated concordances full of them. This issue was identified too late (and in any case, if the corpus already reads it as *andquot;* instead of ", the confusion is for both).

–>

Part II

The language of clouds

Chapter 8

The nature of clouds

When we talk about language, we often talk about the linguistic sign. That term itself is most often linked to Saussure and structuralism, but it is not far from the form-meaning pairing in cognitive linguistics.

The different branches and theoretical frameworks in linguistic differ by how they define each of the elements of the pair. For example, referring to "meaning" or *signifié* aspect: is it linguistic internal? Does it include reference? Does it include *social meaning*? Moreover: should this relationship be studied in relation to an organized system, or discovered from the turbulent chaos of language use?

Cognitive linguistics, the theoretical framework that has guided these studies, accepts a broadly encompassing notion of *meaning* —which certainly does not make it easy to define it— and values the study of language in use. This is wonderfully compatible with the Distributional Hypothesis, the methodological framework that inspires this research, but can lead to disappointing misunderstandings.

The Distributional Hypothesis basically states a correlation between difference in distributional behavior (difference in how forms are used) and semantic difference.

A popular interpretation of this Hypothesis involves using distributional patterns, such as word vectors, as *operationalizations of meaning.* The word space becomes a semantic space, and a type-level model such as Latent Semantic Analysis (LSA) "constructs a single vector for a word meaning" (**bolognesi_2020**). This is the core argument of these models, the selling point of distributional models and word embeddings.

However, as the analysis in this research project can make clear, *usage is not meaning.* Form can be simplified, but both usage and meaning are extremely complex phenomena, and the simplification or abstraction we may make of one will not necessarily correlate with the simplification or abstraction we prefer for the other. Usage correlates with meaning, but that doesn't mean that any specific distributional model will correspond to dictionary senses and the case is, unfortunately, that it doesn't.

Even **sahlgren_2006** makes a point of treating semantic similarity in very vague terms, only distinguishing between paradigmatic and syntagmatic relationships.

A more useful idea might be to think of a triad of form, meaning, and usage. This does not necessarily complexify the picture, since *usage*, in its variety, is very concrete and operationalizable. Even if we cannot equate usage patterns to meaning, the relationship between form and usage patterns can still inform our understanding of meaning. It certainly does not render distributional methods useless. We just have to be careful not to conflate correlated but different phenomena, just like we should not conflate form and meaning.

Chapter 9

Nonsense or no senses?

9.1 Introduction

In linguistic terms, clouds may provide us with different types of information, both at syntagmatic and paradigmatic level. At the syntagmatic level, they may illustrate cases of collocation, colligation, semantic preference or even tendencies towards the open-choice principle. The paradigmatic level, on the other hand, codes the relationship between the clusters and dictionary senses, from heterogeneous clusters to those that represent prototypical contexts of a sense.

Given a naive understanding of the correlation between context and meaning, we would mostly expect, from the paradigmatic perspective, clusters that equal senses: each cluster would cover all the occurrences of a dictionary sense and only the occurrences of that sense. However, even if we relax the requirements, expecting clusters with mostly occurrences of one sense and covering most of the instances that are not discarded as noise, this does not arise often. Instead, even homogeneous clusters only group typical contexts within a sense, which, at the syntagmatic level, tend to correspond to collocations. In any case, as we will see in this chapter, the full picture is much more complex, and we may obtain much richer information than just lexical collocations representing typical contexts within a sense.

In this chapter, we will look into the types of syntagmatic and paradigmatic information that the clouds offer. Section **??** starts with an overview of the different levels in each dimension and mentions a few examples of their interaction in a contingency table. We then elaborate with more detailed examples of each in situation in sections **??** through **??**, and round up with an overall summary in Section **??**.