Cloudspotting: Visual analytics for distributional semantics

Mariana Montes

2021-04-30

Contents

Introduction

Here I'm starting the first draft ever of my PhD dissertation. There are reports scattered all over the place, but here I will try to write things already thinking of a Final Product. I need a layout (briefly discussed at the end of February) and I will slowly start building the final text.

The original title (in any case, the title of my project) was *Methodological triangulation in corpus-based distributional semantics*, but the triangulation part was lost somewhere along the way.

What I really did do was develop the visualization, based on Thomas Wielfaert's original code, analyze the parameter settings in multiple ways, checking different combinations and exploring different avenues, and lead, check, study and compare manual annotation of 32 lemmas.

That is what I'm going to write about.

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.

Part I

Visualization tool

Chapter 1

An interface to the world of clouds

In this part (which will have who knows how many chapters) I mean to describe the visualization tool. It was originally created by Thomas Wielfart, but around July 2019 I started to play around with the code and learn Javascript and D3, 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. It will be more or less redundant with the paper I wrote with Kris in December (Montes & Heylen, n.d.), more or less like vignettes for documentation (as of now, it is still not documented).

The high level of automatization of vector space models makes it possible to analyze semantic patterns in huge amounts of corpus data. While this has the advantage of making semantic analyses more data-driven, this also implies a lower level of control for the linguist, who is confronted with advanced statistical output that is not straightforwardly interpretable in terms of the linguistic phenomena under investigation. In the case of vector space models, and especially an approach that combines multiple parameter settings to generate a multitude of models, the output is an array of distance matrices that "somehow" models different aspects of probabilistic semantic structure. To interpret and understand statistical modelling from a linguistic perspective, we turn to visual analytics.

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 et al., 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 modelling outputs.

Lexical semantic research with vector space models would benefit from a tool that aids i) the visualization of distance matrices based on high-dimensional data, ii) the comparison of multiple models and iii) more detailed examination of the input to these models.

Chapter 2

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.

2.1 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 will be described in [appropriate section].

2.2 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 (1) and take *herhalen* 'to repeat' as the target.

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

`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 (1) 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 (1) 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 (1) 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 (1).

`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 2.1.

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

2.2.1 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.

2.2.2 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.

2.2.2.1 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.

2.3 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 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 the path needed to take to generate clouds from the raw, seemingly indomitable ocean that is a corpus.

First, we will describe token-level vector space models are created. Section 3.1 will explain count-based models, but this is by no means the only viable path. Other techniques, such as BERT (Devlin et al., 2019), that can generate vectors for individual instances of a word, can be used for the first stage of this workflow.

Once we have token-level vectors, 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 3.2 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 into HDBSCAN in

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" (1954, p. 156). In other words, 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. 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 relationship (Sahlgren, 2006, p. 19).

Distributional models operationalize 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 (PMI, see 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 3.1 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	speak/v
linguistics/n	4.37	0.99		3.16	0.41
lexicography/n	3.51	2.18		2.19	2.09
computational/j	1.60	0.08		-1.00	-1.80
research/n	0.20	-0.84	0.04	-0.50	-0.38
chocolate/n	-1.72	-0.53	1.28	-0.73	-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. As we can see in this example, words with similar vectors (e.g. *linguistics* and *lexicography*) are semantically similar, while words with different vectors (e.g. *linguistics* and *chocolate*) are semantically different.

The vectors in Table 3.1 are type-level vectors: each of them aggregates over all the instances of a given word, 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 (2) through (4), where the target item is in italics.

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

(4) 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 Schütze (1998), we replace the context words around the token with their respective type-level vectors (De Pascale, 2019; Heylen et al., 2015).

For example, we could represent example (2) with the vector for its context word *lexicography*, that is, the second row in Table 3.1; example (3) with the sum of the vectors for *linguistics* (row 1) and *computational* (row 3); and example (4) 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 (2) and (3) 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 and visualization techniques based on dimensionality reduction. However, in order to obtain this result we need to make a number of decisions involved, mostly, in defining the context that will be used to represent an item (Cf. Bolognesi, 2020, p. 83).

3.2 Dimensionality reduction

Dimensionality reduction refers to algorithms that 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), which may be run on R with the function `metaMDS()` of the {vegan} package (Oksanen et al., 2020) and `Rtsne()` of the homonymous package (Krijthe, 2018), respectively.

MDS is an ordination technique, like principal components analysis (PCA). It tries out different low-dimensional configurations and tries 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. Nonethelesss,

the R implementation rotates the plot so that the horizontal axis represents the maximum variation. In cognitive linguistics literature both metric (Hilpert & Correia Saavedra, 2017; Hilpert & Flach, 2020; Koptjevskaja-Tamm & Sahlgren, 2014) and nonmetric MDS (De Pascale, 2019; Heylen et al., 2012; Heylen et al., 2015; Perek, 2016) have been used.

The second technique, t-SNE (van der Maaten, 2014; van der Maaten & Hinton, 2008), has also been incorporated in cognitive distributional semantics (De Pascale, 2019; Perek, 2018). The algorithm is quite different from MDS, but 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. T-SNE was the state-of-the-art visualization technique for word vectors in computational linguistics (Smilkov et al., 2016) but is now generally being replaced by UMAP.

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 et al., 1999, p. 18; Wielfaert et al., 2019, p. 222). In addition, t-SNE requires setting a parameter called perplexity, which basically sets how many neighbors the preserved local structure should cover.

Chapter 4

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 et al., 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 4.4) we will briefly simulate the path an user would take from Level 1 to Level 3. This perspective was also taken in Montes and Heylen (n.d.). Finally, Section 4.5 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 2021-04-30 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.

4.1 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.

4.2 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.

4.3 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.

4.4 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, p. 97). 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.

4.4.1 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.

4.5 Wishlist

Chapter 5

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 et al., 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 et al. (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 (Hahsler & Piekenbrock, 2021). 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.

Part II

The language of clouds

Chapter 6

Nonsense or no senses?

In this part, which will have who knows how many chapters, I will delve into the theroetico-methodological insights –the theoretical impact, if you will– of my analyses.

I will have to describe the annotation procedure (probably) but, more importantly, discuss the main theoretical observations derived from my studies.

The main points, for now, are:

- From cloud interpretation, what we see are not necessarily senses, but rather "common/shared/similar" contexts of usage.
- There is no single optimal solution for every item.
- Because of the second point, what do the different parameter settings do, for specific items? (What kind of info do they pick up?)
    - This is also material for the first part of the project's monograph

Currently (but there is still much research to do) I think that different parameters offer different perspectives, but picking out different aspects of the context. Those perspectives won't be relevant for all lemmas –prepositions, if dependency-informed, are relevant for *hoop* but not for other nouns. Even that relevance is gradual. We might be able to generalize, classifying items depending on which parameter settings are relevant to them, or depending on what they would look like with certain or all kinds of parameters.

Say, *horde* and *heffen* look quite good with just any setting, while *haten* looks awful always. Still, this is based on the current way of computing vectors, which adds the type-level vectors of the tokens instead of averaging over them (which is what I thought they did). And I still have to look at the adjectives and revise nouns and verbs using the medoids based on euclidean distances.

Will these analyses change my current conclusions? Chan chan chaaaannn...

In any case, I will try to look at all of that in the last two weeks of February and then discuss with DG the content for this part .

Chapter 7

The nature of clouds

Clouds don't necessarily show senses but usage patterns –the senses are well distinguished insofar they match those usage patterns. But we could have one usage pattern for multiple senses, multiple usage patterns for one sense, and, more importantly, different degrees of definition of each usage pattern.

It is not a matter of frequency we can have infrequent senses/patterns that are nonetheless strong enough to form a clear group; and
extremely frequent senses can also be grouped separately, they don't necessarily absorb the minor senses/patterns, if their pattern is characteristic enough.

Bibliography

Bolognesi, M. (2020). *Where Words Get their Meaning: Cognitive processing and distributional modelling of word meaning in first and second language* (Vol. 23). John Benjamins Publishing Company. https://doi.org/10.1075/celcr.23

Campello, R. J. G. B., Moulavi, D., & Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. In J. Pei, V. S. Tseng, L. Cao, H. Motoda, & G. Xu (Eds.), *Advances in Knowledge Discovery and Data Mining* (pp. 160–172). Springer. https://doi.org/10.1007/978-3-642-37456-2_14

Card, S. K., Mackinlay, J. D., & Shneiderman, B. (1999). *Readings in information visualization: Using vision to think*. Morgan Kaufmann Publishers.

Church, K. W., & Hanks, P. (1989). Word association norms, mutual information, and lexicography. *ACL '89: Proceedings of the 27th annual meeting on Association for Computational Linguistic*, 76–83. https://doi.org/10.3115/981623.981633

De Pascale, S. (2019). *Token-based vector space models as semantic control in lexical lectometry* (PhD Dissertation). KU Leuven. Leuven. Retrieved November 8, 2019, from https://lirias.kuleuven.be/retrieve/549451

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [arXiv: 1810.04805]. *arXiv:1810.04805 [cs]*. Retrieved April 27, 2021, from http://arxiv.org/abs/1810.04805

Hahsler, M., & Piekenbrock, M. (2021). *Dbscan: Density based clustering of applications with noise (dbscan) and related algorithms* [R package version 1.1-6]. https://github.com/mhahsler/dbscan

Harris, Z. S. (1954). Distributional structure [Publisher: Taylor & Francis]. *Word*, *10*(2-3), 146–162.

Heylen, K., Speelman, D., & Geeraerts, D. (2012). Looking at word meaning. An interactive visualization of Semantic Vector Spaces for Dutch synsets. *Proceedings of the eacl 2012 Joint Workshop of LINGVIS & UNCLH*, 16–24.

Heylen, K., Wielfaert, T., Speelman, D., & Geeraerts, D. (2015). Monitoring polysemy: Word space models as a tool for large-scale lexical semantic

analysis. *Lingua*, *157*, 153–172. https://doi.org/10.1016/j.lingua.2014.12.001

Hilpert, M., & Correia Saavedra, D. (2017). Using token-based semantic vector spaces for corpus-linguistic analyses: From practical applications to tests of theoretical claims. *Corpus Linguistics and Linguistic Theory*, *16*(2). https://doi.org/10.1515/cllt-2017-0009

Hilpert, M., & Flach, S. (2020). Disentangling modal meanings with distributional semantics. *Digital Scholarship in the Humanities*, (fqaa014). https://doi.org/10.1093/llc/fqaa014

Kiela, D., & Clark, S. (2014). A Systematic Study of Semantic Vector Space Model Parameters. *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality*, 21–30.

Koptjevskaja-Tamm, M., & Sahlgren, M. (2014). Temperature in the word space: Sense exploration of temperature expressions using word-space modelling. In B. Szmrecsanyi & B. Wälchli (Eds.), *Aggregating Dialectology, Typology, and Register Analysis* (pp. 231–267). DE GRUYTER. https://doi.org/10.1515/9783110317558.231

Krijthe, J. (2018). *Rtsne: T-distributed stochastic neighbor embedding using a barnes-hut implementation* [R package version 0.15]. https://github.com/jkrijthe/Rtsne

Levshina, N. (2015). *How to do linguistics with R: Data exploration and statistical analysis*. John Benjamins Publishing Company.

McInnes, L., Healy, J., & Astels, S. (2016). How HDBSCAN Works — hdbscan 0.8.1 documentation. Retrieved April 30, 2021, from https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

Montes, M., & Heylen, K. (n.d.). Visualizing Distributional Semantics. In D. Tay & M. X. Pan (Eds.). Mouton De Gruyter.

Montes, M., & QLVL. (2021). QLVL/NephoVis: Stratocumulus. https://doi.org/10.5281/ZENODO.4726926

Oksanen, J., Blanchet, F. G., Friendly, M., Kindt, R., Legendre, P., McGlinn, D., Minchin, P. R., O'Hara, R. B., Simpson, G. L., Solymos, P., Stevens, M. H. H., Szoecs, E., & Wagner, H. (2020). *Vegan: Community ecology package* [R package version 2.5-7]. https://CRAN.R-project.org/package=vegan

Perek, F. (2016). Using distributional semantics to study syntactic productivity in diachrony: A case study [Publisher: De Gruyter Mouton Section: Linguistics]. *Linguistics*, *54*(1), 149–188. https://doi.org/10.1515/ling-2015-0043

Perek, F. (2018). Recent change in the productivity and schematicity of the way-construction: A distributional semantic analysis [Publisher: De Gruyter Mouton Section: Corpus Linguistics and Linguistic Theory]. *Corpus Linguistics and Linguistic Theory*, *14*(1), 65–97. https://doi.org/10.1515/cllt-2016-0014

Sahlgren, M. (2006). *The word-space model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-*

dimensional vector spaces (Doctoral dissertation) [OCLC: 255579201]. Dep. of Linguistics, Stockholm Univ. [u.a.] Stockholm.

Schütze, H. (1998). Automatic Word Sense Discrimination. *Computational Linguistics*, *24*(1), 97–123.

Shneiderman, B. (1996). The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. *IEEE Visual Languages*, 96–13.

Smilkov, D., Thorat, N., Nicholson, C., Reif, E., Viégas, F. B., & Wattenberg, M. (2016). Embedding Projector: Interactive Visualization and Interpretation of Embeddings [_eprint: 1611.05469]. arXiv:1611.05469

van der Maaten, L. (2014). Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, *15*, 3221–3245.

van der Maaten, L., & Hinton, G. (2008). Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, *9*, 2579–2605.

Wielfaert, T., Heylen, K., Speelman, D., & Geeraerts, D. (2019). Visual Analytics for Parameter Tuning of Semantic Vector Space Models. In M. Butt, A. Hautli-Janisz, & V. Lyding (Eds.), *LingVis: Visual analytics for linguistics* (pp. 215–245). CSLI Publications, Center for the Study of Language; Information.