universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„LeMATo - LexicoMetric Analysis TOol“

verfasst von / submitted by

Martin Albin Perdacher, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree

Diplom-Ingenieur (Dipl-Ing.)

Wien, 2016 / Vienna, 2016

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet	A 066 940
Studienrichtung lt. Studienblatt:/ degree programme as it appears on the student record sheet	Masterstudium Scientific Computing
Betreut von / Supervisor:	ao. Univ.-Prof. MMag. Dr. Werner Winiwarter

Danksagungen

Ich möchte meinem Betreuer Herrn Prof. MMag. Dr. Winiwarter danken, dass er mir sehr viel Raum für meine eigenen Ideen gegeben hat und dass ich diese auch verwirklichen durfte. Seine genaue und zügige Arbeitsweise, sowie seine wissenschaftliche Beratung haben zum Gelingen dieser Arbeit beigetragen.

Besonderen Dank möchte ich auch meinen Eltern Monika und Albin Perdacher für Ihre Unterstützung in jeglicher Hinsicht durch mein gesamtes Leben ausprechen. Viele Dinge hätte ich ohne sie nicht erreichen können.

Auch meiner Frau Eva Perdacher möchte ich hier auf diesem Weg danken. Ihre Geduld, ihr Vertrauen und ihr unglaublich großes Herz haben mir unzählige Male die nötige Ruhe gegeben, die für ein Studium notwendig ist.

Eine Arbeit beginnt mit der Idee. Hierfür danke ich meinem Freund Alexander Böckmann.

Zu guter Letzt möchte ich auch meiner Großmutter Theresia Rasinger danken. Ohne Ihre Unterstützung hätte ich diese Arbeit nicht fertigstellen können.

Contents

I Introduction and preliminaries

1 Introduction **1**

2 Comparison of tools applied in lexicometrics **3**

II Theory **7**

3 Lexicometrics, support for discourse analysis **9**
3.1 Notes on lexicometrics . . . 10
3.1.1 Corpus compilation . . . 10
3.1.2 Corpus-based vs corpus-driven . . . 10
3.1.3 Linguistic preprocessing . . . 11
3.2 Frequency analysis . . . 12
3.3 Concordance analysis . . . 14
3.4 Analysis of characteristics in sub-corpora . . . 14
3.5 Co-occurrence analysis . . . 15
3.6 Grouping support . . . 16
3.7 Micro analysis . . . 16

4 Discourse theory **21**
4.1 Post-structuralism . . . 21
4.1.1 Structuralism . . . 21
4.1.2 Language as a representational system of society . . . 22
4.1.3 French discourse analysis . . . 22
4.2 Discourse methodology with lexicometrics . . . 23
4.2.1 Lexicometrics as macro analysis . . . 24

5 Finding themes **27**
5.1 What is a theme? . . . 27
5.2 The word space model . . . 29
5.2.1 Related vector spaces . . . 29
5.2.2 Hyperspace analogue to language . . . 30
5.3 Clustering . . . 31
5.3.1 Related clustering of word-context matrices . . . 32
5.3.2 Hierarchical agglomerative clustering . . . 32

6 Calculations in the vector space **35**
6.1 Co-occurrence measures . . . 35
6.2 Adjusting the weights . . . 37
6.2.1 Tf-idf . . . 37
6.2.2 Point-wise mutual information . . . 37
6.3 Smoothing the matrix . . . 38
6.4 Similarities and dissimilarities . . . 39
6.4.1 Similarities . . . 39
6.4.2 Dissimilarities . . . 40
6.4.3 Implementation notes . . . 41
6.5 Linkage criteria . . . 41
6.5.1 Maximum or complete-linkage clustering . . . 41
6.5.2 Minimum or single linkage clustering . . . 41
6.5.3 UPGMA . . . 42
6.5.4 WPGMC . . . 42
6.5.5 Implementation notes . . . 42

7 Significant words **43**
7.1 Comparing words using reference corpora with test statistics . . . 44
7.1.1 Chi-square test . . . 44
7.1.2 Log-likelihood ratio test . . . 45
7.2 Comparing words using reference corpora with frequency adjustment . . . 46
7.2.1 Characteristic Element Diagnostic - ced . . . 46
7.3 Notes on corpus comparison . . . 48

III Implementation **51**

8 Requirements on LeMATo **53**

9 Software Stack **57**
9.1 Groovy and Grails . . . 57
9.2 Elasticsearch . . . 58
9.3 The S-Space Package . . . 60
9.4 Other software dependencies . . . 61
9.4.1 Querying and using Elasticsearch . . . 61
9.4.2 Layout and appearance . . . 62

10 Design of LeMATo **65**
10.1 Grails MVC(S) . . . 65
10.2 Elasticsearch . . . 66
10.2.1 Relational Elasticsearch . . . 66
10.2.2 Obtaining an Elasticsearch client . . . 67
10.3 Frequency analysis . . . 70
10.4 Concordance analysis . . . 71
10.5 Significance analysis . . . 71
10.5.1 Significance measure . . . 71
10.5.2 Grouping words according to their similarity . . . 71
10.6 Latency . . . 73

IV Evaluation and conclusion **75**

11 Evaluation of results in LeMATo **77**
11.1 Frequency analysis 77
11.2 Significance analysis 81
11.3 Themes 82

12 Concluding remarks **89**
12.1 Lessons learned 90
12.2 Outlook 90

Appendices **93**

A Abstract **95**
A.1 English abstract 95
A.2 Deutsche Zusammenfassung 96

B User Guide **97**
B.1 Corpus definition 97
B.2 Querying in LeMATo 98
B.3 Frequency analysis 100
B.4 Concordance analysis 102
B.5 Characteristics analysis of sub-corpora 102

C Supplementary material **109**

List of Figures

3.1 Preprocessing of a text stream. The text stream gets tokenized and filtered to process the stream into terms. Terms are the basic unit in the vocabulary of a corpus, which gets counted and leads to term frequencies. 12
3.2 Web-flow for the frequency analysis. The frequency table (a) gives a good overview of the most frequent words. They could be further explored in the diachronic view (b) as well as the word frequencies for each period (c). 13
3.3 Word tree for the concordance analysis of the word "dance" in the following three sentences: "how they dance in the courtyard", "some dance to remember" and "some dance to forget". The words "some" and "to" have a larger font, because they appear two times in the text. 14

4.1 The Saussurian sign. 22
4.2 Workflow performed in "Die neoliberale Stadt" [78]. 24
4.3 Selected collocations with "Köln". The closer a word is placed in the centre, the higher the ced-value. The font-size corresponds to the word frequency. Figure taken from [78]. Mattissek permits us to distribute the figure through this master thesis. 25
4.4 Collocations of keywords referring to the city image marketing. The thicker the arrows, the more often terms co-occur. This figure is made by hand. Figure taken from [78]. Mattissek permits us to distribute the figure through this master thesis. 26

5.1 The aim of this grouping is to increase the legibility for the reader. There are media related terms on the left and social themes to the right. Terms are the more significant (CED value, see Section 7.2.1), the more they are placed to the center. Figure taken from [78]. Mattissek permits us to distribute the figure through this master thesis. 28
5.2 Enumeration describing the theme finding process. 29
5.3 A simplified representation of theme finding. Six words are applied to the word vector space according to their spatial proximity within the corpus. A clustering algorithm finds two groups (a fire fighting group marked in green and a media related group marked in red). 29
5.4 Dendrogram for the sentence in the Hyperspace Analogue to Language (HAL)-matrix (see Table 5.1). 34

6.1 L_1 metric (red dashed line) and L_2 metric (green dashed line) in a two dimensional plot. 40

7.1 Calculation of the ced-value: Hyper-geometric distribution *(corpus size (N): 160000, size of sub-corpus (n): 20000, word frequency in the whole corpus (M): 36).* 47

8.1 Use case diagram for LexicoMetric Analysis TOol (LeMATo). Each analysis requires a corpus definition. 56

9.1 Depicting Grails architecture. 57

10.1 Grails Model-View-Controller-Service (MVCS) paradigm as it is used in LeMATo 65
10.2 Sequence diagram of MVCS archtecture. 66
10.3 Class diagram. 68
10.4 Obtaining an Elasticsearch client using Spring factory. 69
10.5 Web flow for the frequency analysis. 70
10.6 For the calculation of significant terms in the subset, we are using the superset of the corpus as a background reference. 72
10.7 Steps to perform the significance analysis. 72
10.8 Latency for the frequency analysis. See Table 10.1 for details to our testing environment. 74
10.9 Latency for the significance pipeline. See Table 10.1 for details to our testing environment. 74

11.1 Comparing frequencies observed in LeMATo (built with stemming and stopwords) with frequencies observed in AntConc (built with lemmatisation). . . . 78
11.2 Term frequencies of the sub-corpora. 81
11.3 Comparison of the significance calculation in AntConc and in LeMATo for the sub-corpus labelled with the tag “Frankfurt”. 84
11.4 Comparison of the significance calculation in AntConc and in LeMATo for the sub-corpus labelled with the tag “Leipzig”. 85
11.5 Comparison of the significance calculation in AntConc and in LeMATo for the sub-corpus labelled with the tag “Köln”. 86
11.6 Dendrogram on the discourse of the expansion of the airport in Frankfurt. . . 87

B.1 Creating a new corpus with name and description. 98
B.2 Importing a LexisNexis file and adding it to the corpus. 99
B.3 Parameter selection for frequency analysis. 100
B.4 Frequency distributions for the overall corpus. From left to right: term frequencies, document frequencies, paragraph frequencies and sentence frequencies. 101
B.5 Table of words with top frequencies within the corpus. The columns from left to right: rank (order of highest document frequency, keyword (term), document frequency, term frequency and kendall's τ). 102
B.6 Diachronic view for selected terms. From left to right: term frequencies, document frequencies, paragraph frequencies and sentence frequencies. 103
B.7 List of documents with the occurrence of the term “berlin” in the year 2001. The columns of the table from left to right: tags, date, starts with (the first 100 characters of the document), keyword frequency (how often does the term occur in the document) 104
B.8 Parameter selection for the concordance analysis. 104
B.9 Double word tree for all occurrences of the term “Berlin”. 104

B.10 Concordance analysis table for all occurrences of the term “Berlin”. The columns from left to right: date, tags and the text fragment, where the word occurs. 105
B.11 Table for the characteristics analysis of subcorpora for the tag: “Frankfurt” . 105
B.12 Scatterchart of document frequencies against the score (Chi-square significance). 106
B.13 Part of dendrogram for the tag “Frankfurt” (size: 50 terms). 107
B.14 All significant terms of a subbranch. Obtained through a right-click on a node in the dendrogram. 108

List of Tables

2.1 Related programs. 6

5.1 Example matrix (Hyperspace Analogue to Language (HAL)) for the sentence "I saw the man with the binoculars" and a window width of five words. Words are weighted inversely proportional to the distance of the focus word. 31
5.2 Similarities based on the city-block metric between all rows of the HAL-matrix (see Table 5.1 on page 31). 33
5.3 Similarities are joined with the maximum linkage criterion. 33

7.1 An example of a contingency table. The frequencies of the word "foo" are queried in two different corpora. In the wikipedia corpus (en) the word "foo" occurs 4500 times and in the BNC the word "foo" occurs 40 times. The size of the wiki corpus is 1.9 billion words and the size of the BNC is 100 million words. The label "not foo" refers to all other words. 43
7.2 Contingency table with observed and expected values calculated with frequencies of Table 7.1. The expected frequencies are calculated with a simple cross-multiplication. The expected frequency for "foo" in the wiki corpus is calculated as follows: $c * \frac{a+b}{c+d}$. We assume an equal distribution of the words for both corpora. 44

10.1 Hardware used for testing the latency. 73

11.1 Querying the top three terms in Elasticsearch yields a wrong term frequency of 10 for the term "foo". This is due to top aggregation on each shard. 79
11.2 Stop-words in Lucene 4.10.4. These stop-words are obtained through the following code snippet: `GermanAnalyzer.getDefaultStopSet()`. The stop-words marked in red have a apparent difference in their frequency (see Table 11.3). . 79
11.3 Top fourty term frequencies of our reference corpus with the frequencies observed in LexicoMetric Analysis TOol (LeMATo) and AntConc. Stop-words are marked in red. 80

C.1 Top 500 frequencies of our reference corpus with the term frequencies observed in LeMATo and AntConc (ordered by Elasticsearch document frequencies). Higher term frequency scores are marked for LeMATo in blue and AntConc in red. 109

Part I

Introduction and preliminaries

Chapter 1

Introduction

Discourse analysis is a general term in various disciplines, including linguistics, sociology, anthropology, human geography and communication studies, to name just a few. The focus of discourse analysis is to build up meaning in larger communicative rather than grammatical units. Furthermore, discourse analysis aims to reveal connections of text and speech to institutional structures. Discourse analysis connects text to a historical or social context. The analysis of the text differs among several disciplines. In social sciences and humanities, content analysis has become the state of the art approach to analyse written text of various types, including writings, images, recordings and cultural artefacts. In content analysis, one starts with inferences about the antecedents and the characteristics of the communication. These assumptions about the content enable the content analyst to reveal the meaning of the text within a category system. Discourse analysis criticises the supposition about the inferences and especially French discourse analysts emphasise, that such a category system increases the risk to generate tautologies [14, as cited in 33].

The methods of lexicometrics and corpus linguistics have their roots in Saussure's structural linguistics [108] and enable the distinction of words with different meanings in discursive formations. Corpus linguistics is a methodology in linguistics to develop theories about languages based on statistical data gained from textual corpora. Corpus linguistics assumes that the analysis of a large amount of text explores the language in a meaningful way. In classical corpus linguistics, general statements could be made about the properties of a particular language. There are databases for language-specific reference corpora (e.g. Brown corpus [41] in the USA, British National Corpus [1] in Great Britain, Frantext [76] in France or the Russian Reference Corpus [115] in Russia) These databases enable the measurement of the characteristics with the help of computer programs.

Lexicometrics could be seen as a part of corpus linguistics, but unlike corpus linguistics, which deals mainly with language-specific reference corpora, the corpora in lexicometrics are compiled concerning the research question. Lexicometrics has its tradition especially in the French discourse analysis. The first contribution to lexicometrics goes back to Michel Pêcheux at the end of the 60s. He published a work on automated discourse analysis [92]. This work is an attempt to provide a scientific instrument for discourse analysis that serves as means for the researcher to get rid of the subjective readings of texts. His work is a cornerstone in the modern lexicometric analysis.

Lexicometric methods investigate the quantitative relationships between lexical items in closed corpora. A closed corpus is fixed in definition and compilation, which is not changed during the research process. In the context of discourse-oriented approaches, lexicometric methods are used to infer from discursive structures and their differences between different contexts. The goal of lexicometric methods is to extract a large-scale meaning of the textual corpora. The focus of lexicometrics is the relationship of lexical elements (e.g. words) within the corpus, and the interpretation happens at the end of the research process [33]. Lexicometric analysis has been applied in social science and human geographic studies (e.g. [78], [109], [32] and [121]).

In this thesis, we introduce LexicoMetric Analysis TOol (LeMATo), a lexicometric web application, which enables the analysis of lexical elements in a closed corpus. LeMATo analyses the corpus in a four stage process, and each step is based on the promising ideas, published in [33], for lexicometric analysis as a methodological approach for discourse analysis. First, the *frequency analysis* constructs a list with terms and their frequencies and examines the frequency development in time for a subset of words. Second, the *concordance analysis* provides a Key-Word In Context (KWIC) analysis, which reveals the context of a queried term (i.e. the keyword). Third, the *analysis of characteristics in sub-corpora* groups the most frequent lexical elements into families based on annotations from a reference corpus. Fourth, the *co-occurrence analysis* uses the most frequent terms of the concordance analysis to measure their distance in document, section and sentence to extract the most significant words. At the end of each stage LeMATo shows a meaningful visualisation of the results.

There are many programs for corpus linguistics and the analysis of language-specific features in texts, but there is no program, designed as a tool for discourse analysis to perform a lexicometric analysis. In Chapter 2 we list different corpus linguistic programs and argue for the development of LeMATo. At next, in Chapter 3 we introduce lexicometrics and the concept of the four anlyses in LeMATo. In Chapter 4 we introduce the background of discourse analysis and outline the methodological approach of a human geographic study (i.e., [78]), which is used in this thesis as a reference study. Therefore, we are using the same corpus here. In two stages of our four stage analysis process, we want to put words gained from different statistical filterings into groups. Therefore, we introduce our approach to the problem of "theme finding" as a transformation of the corpus into a vector space model, which is presented in Chapter 5. We explain the mathematical formulations of the vector space in Chapter 6. In the *analysis of characteristics in sub-corpora* and in the *co-occurrence analysis* we need to examine significant words. What significance in the context of corpus comparison means, we explain in Chapter 7.

In the next part of this thesis, we present the implementation of LeMATo. In Chapter 8 we start with a summary of the needs in lexicometrics and formulate the requirements for LeMATo. At next in Chapter 9 we give an overview in the different pieces of frameworks and databases we use. The interaction of the different software pieces and how they are used describes the design of LeMATo in Chapter 10.

In the final part we evaluate LeMATo. We compare the results of LeMATo with the results gained from AntConc [3] in Chapter 11. We give some concluding remarks in Chapter 12. In the appendices we provide an user guide and supplementary material.

Chapter 2

Comparison of tools applied in lexicometrics

There are various programs for different operating systems, which aid a lexicometric analysis. These programs are designed to address particular linguistic problems and come with a large number of different features. For example, AntConc [3] is intended to perform a concordance analysis and Ngram Statistic Package (NSP) [8] [86] is designed to perform n-gram statistics on words and characters. Wordsmith [112] and Lexico3 [67] are two programs, which have been already used in socio-scientific studies (cf. [33, p. 250]). The strength of these two programs is that they offer essential lexicometric features like frequency analysis, concordance and co-occurrence analysis as well as n-gram statistics. Table 2.1 summarises key issues for lexicometrics.

AntConc [3] started as a relatively simple concordance program, but it grew into a useful text analysis software. A concordance plot visualises the concordances and outlines the occurrences of a word within a file. Each context could also get looked up in its origin (i.e. the text file). The concordance plot enables the researcher to look at every use of a word. This feature, the easy use, and the availability across the most common operating systems makes AntConc a comprehensive lexicometric tool.

The current version of **Lexico3** [67] was published in 2001 and provides lexicometric features such as text segmentation, frequency analysis, concordance analysis and comparison of subcorpora based on the Characteristic Element Diagnostic (CED) statistics. In the case of a multivariate analysis, Lexico3 offers a factor analysis, but only for six dimensions. The program is free of charge for temporary personal work, but for commercial and university use a license is required. Unfortunately, it was not possible to make Lexico3 (version 3.45) run with the current Windows operating system (version 8.1), not even in the Windows95 or WindowsME compatibility mode.

WordSmith Tools [112]) can be used in 80 different languages and the core software package includes three modules: Concord, WordList and KeyWord. Concord is a concordancer; WordList displays the frequencies of words with some statistics and KeyWord helps to find significant words with different word forms within the text. Different corpora can be compared with word lists using statistical tests and the Sørenson-Dice coefficient (see Section 6.1) to compare vocabularies of two texts. Wordsmith has been used to investigate the size of

reference corpora [11].

Programs performing multivariate analysis like cluster or factor analysis are rare and offer limited possibilities. Two exemplary programs for this purpose are SenseClusters [87] and HyperBase [124] [17].

SenseClusters [87] is a bundle of Perl scripts, which allows a user to cluster similar contexts using unsupervised learning approaches. There are three different native applications for SenseClusters: Word-sense discrimination, context clustering and word clustering. *Word-sense discrimination* discovers different meanings of a target word. *Context clustering* groups headless contexts (e.g. documents, short messages or emails) based on their topic. *Word clustering* builds up a word-by-word matrix based on co-occurrences (or bigrams) in a common context and clusters according to the vector similarity. The input for SenseClusters is an XML file with a SENSEVAL-2 format (see `http://www.senseval.org/`). The use of Perl scripts and the conversion to the specific format raise some challenges to social scientists.

HyperBase [124] [17] in its current version is free of charge. It enables basic lexicometric approaches like frequency analysis, concordance analysis, as well as multivariate analysis like factor analysis. HyperBase generates some nice histograms to analyse the significance in corpora and a tree view, which's reminiscent of a dendrogram that reveals distances of text fragments [1]. The documentation to HyperBase is written in French and is almost exclusively used in France and Canada.

There are also online databases for corpora, like **Cosmas II** [51], **DWDS** [60] or **CCDB** [22] which allow querying without the use of a program. They provide some basic lexicometric analysis, but it is not possible to upload and use a customised corpus compilation. All these three do not offer any multivariate analysis.

Tools for qualitative data analysis like **MAXQDA** [118] or **Atlas.ti** [6] are powerful in systematically evaluating and interpreting texts, but do not provide any corpus linguistic features. For discourse oriented analysis in social science, it is desirable to add corpus linguistic features to these tools.

There are a lot of command line tools for certain tasks in sociolinguistic studies. One example is TinyCC [13], which calculates co-occurrences of words and provides a list of significant neighbour co-occurrences, but lacks in lemmatisation and legibility of the results.

Finally, there is no program designed for the use in a socio-scientific and discourse oriented context. Programs like Lexico3 and WordSmith lack the visualisation of frequencies and co-occurrences on a flexible document compilation, e.g. characteristics in sub-corpora. There is no program which aids the analysis of the frequency development in newspaper articles over several years (i.e. diachronic frequencies). For every distinction in the analysis, a new corpus definition has to be made, which is cumbersome in standard corpus linguistic tool suites. None of the programs for corpus linguistics are concurrent and take advantage of today's multi-core architectures.

Our tool LexicoMetric Analysis TOol (LeMATo) analyzes the corpus in a four stage process. Each stage of LeMATo is based upon the promising ideas of theoretical discourse analysis published in [33] and includes a diachronic *frequency analysis* and a *concordance analysis* which reveals the context of a queried word. LeMATo provides two multivariate analyses

[1] see `http://fr.wikipedia.org/wiki/Hyperbase`, last visit at 30th of May 2015.

with the *analysis of characteristics in sub-corpora* and the *co-occurrence analysis*. These four analysis steps are introduced and explained in Chapter 3.

name	concordancer	platform	comp. corpora	multiv. analysis	price
AntConc	mutual information	Linux Mac Windows	no	no	free
Lexico3	yes	Windows	yes	no	$\geq$ 150 €
TinyCC	no	Linux Mac Windows	no	no	free
WordSmith Tools	yes	Windows	yes	no	$\geq$ 50 £
SenseClusters	no	perl scripts	no	vector space	free
HyperBase	yes	Windows	yes	factor analysis	free

Table 2.1: Related programs.

Part II

Theory

Chapter 3

Lexicometrics, support for discourse analysis

> All intelligent thoughts have already been thought; what is necessary is only to try to think them again.
>
> Johann Wolfgang von Goethe in Journeyman Years, 1821-1829

The methods of lexicometrics and corpus linguistics have their basis in de Saussure's structural linguistics and enable the distinction of words with different meanings in discursive formations [108]. Corpus linguistics is the study of language based on large corpora samples. In classical corpus linguistics, general statements could be made about the properties of a particular language. Furthermore, corpus linguistics develops theories about how language is governed by rules and the relatedness to another language [79]. As in corpus linguistics, lexicometrics also deals with quantitative relationships of lexical elements, such as frequencies and distances of words, therefore lexicometrics could be seen as a part of corpus linguistics.

Content analysis is a family of a large set of diverse research approaches and techniques in social science and humanities to identify methods for studying and retrieving meaningful information from text. Content analysis usually has an assumption about the documents or the corpus itself [62]. Lexicometrics is not a replacement for content analysis. In lexicometrics, the process of interpretation shifted to the end of the study, which makes it, in contrast to content analysis, a suitable tool for discourse analysis. Lexicometrics is a computational approach to capturing the quantitative properties of a corpus and revealing linguistic structures of large corpora. Lexicometrics is used to explore the corpus and formulate impartial assumptions, which could be employed in a follow-up content analysis study [33].

Lexicometric methods and their use in linguistics go back to Zelig Harris in 1954 [47]. His intention was to take into account the structure of the text for his linguistic transformation. Several studies used lexicometrics until now [1]. Lexicometric analysis has been used to analyse the discourse on the Arab Spring [121]. The frequency of keywords, collocations and a

[1] [121], [78], [32] and [109]

concordance analysis, calculated by AntConc [3], is used to visualize positions, developments over time, breaks and shifts in the discourse (see [121]).

The aim of lexicometric approaches in discourse studies is to identify significant semantic structures in digital corpora and carve out similarities and differences in sub-corpora. In the following Section 3.1, we introduce the lexicometric vocabulary and outline some considerations, before doing a lexicometric study. LexicoMetric Analysis TOol (LeMATo) follows the concepts and ideas published in [33]. These concepts are explained in the following. LeMATo analyses the corpus in a four stage process. First, the *frequency analysis* constructs a list with terms and their frequencies and works out the diachronic frequency development for a subset of words (see Section 10.3). Second, the *concordance analysis* provides a Key-Word In Context (KWIC) analysis for one selectable word (see Section 10.4). Third, the *analysis of characteristics in sub-corpora* brings out the most frequent and the most specific lexical elements (see Section 3.4) of a sub-corpus. Fourth, the *co-occurrence analysis* reveals the most significant words in text units, where a queried word co-occurs (see Section 3.5). We provide a meaningful visualization to each of the four analyses in LeMATo.

3.1 Notes on lexicometrics

3.1.1 Corpus compilation

A linguistic analysis is performed on a corpus, a large digital collection of text. The corpus contains written language, spoken language or both. In lexicometrics, the corpus is fixed in the definition (i.e. the corpus is *closed*), this means that the corpus will not change during the investigation. A closed corpus claims to contain all or nearly all of the data from a particular field (all texts of a particular newspaper, periodic reports of an organisation, all speeches of Presidents). Often it is not possible to fulfil this criterion. The alternative approach is to filter the whole corpus with certain keywords or to filter based on thematic groups [33]. In addition to that, the corpus is split in several sub-corpora. This annotation of sub-corpora enables the distinction of different communication channels, speaker positions or text genres. In our reference study [78] the corpus got filtered on city names to define three sub-corpora (see Section 4.2).

Speaker positions (according to [40]) are most often important and representative persons of organisations, which occur in scientific texts or newspaper articles. For bigger studies there are often a series of publications published by an organisation, which could be used to perform a lexicometric analysis [33].

It is far beyond this thesis to give a detailed description of the compilation of the corpus. For a more detailed description see [7].

3.1.2 Corpus-based vs corpus-driven

The methods for lexicometrics fall in two different approaches: corpus-based and corpus-driven. In the first one, the researcher has a particular hypothesis about the linguistic pattern of the corpus. The researcher queries the corpus to explore this pattern and examines the distribution of the lexical items. This is the *corpus-based* approach because the researcher

validates his theory using the corpus data. The second approach is called *corpus-driven*. Here the researcher uses the corpus itself for the source of his hypothesis about language. It is the corpus-driven method, which

> "... builds up the theory step by step in the presence of the evidence, the observation of certain patterns leads to a hypothesis, which in turns leads to the generalisation in terms of rules of usage and finally finds unification in a theoretical statement." [116]

This quotation means that the corpus-driven method itself embodies the structures of the corpus and is the source of the hypothesis and thus makes the corpus-driven method more inductive. This enables the researcher to make findings of the structures of the text, which have not been subject at the beginning of the investigation [116]. In LeMATo, we provide two analyses, namely the *analysis of characteristics in sub-corpora* (see Section 3.4) and the *co-occurrence analysis* (see Section 3.5), which are nearly corpus-driven. These analyses analyse the corpus without further dictionaries but use a query to distinguish between subcorpora. This query does not infer on the corpus itself, it only distinguishes between different subcorpora.

3.1.3 Linguistic preprocessing

Here we describe the substantive linguistic issues of tokenization, stemming and filtering, which determine the vocabulary of terms which LeMATo uses. Tokenization is the process of chopping a stream into meaningful elements called tokens (see Figure 3.1). This also includes the removal of punctuation marks.

Stop words occur very frequently in language and do not support to find any characteristics within text units. Stop words include function words, such as "the", "is", "at", "which" or "on" and are filtered out in the linguistic preprocessing (see "filtering" in Figure 3.1). Filtering of stop words could be an issue for the particular search on names that include stop words, such as "The Who" or "Take That".

After the filtering processes, the filtered tokens are reduced to a basic form. This generalisation of tokens to terms is necessary, due to the nature of text with their different forms of words. Verbs occur in various conjugation forms, as well as nouns, adjectives or pronouns occur in various declension forms. For most of the cases, these different inflexions have to be grouped together according to the identification rules of a single form. In other words, the lexical items need to be *lemmatised* [69].

The lemmatisation of the vocabulary from a corpus could lead to some ambiguities of lemmas in different forms, also called homographs (e.g. *match* from the competitive sports event or *match* in referring to the equivalence of two or more items with the same characteristics). That is not a great deal as long as it does not violate the theoretical assumptions or hypothesis. For example, in a German gender specific study, it might be necessary to distinguish between the female ("Lehrerin") or male form ("Lehrer") or even the gendered form ("LehrerIn") of a word.

As well as lemmatisation, *stemming* is also focused on reducing the inflectional form or derivative-like form to a common base form. Stemming is a naive algorithm, where the ending of the word is chopped off in the hope of doing it most of the time correctly. Lemmatisation

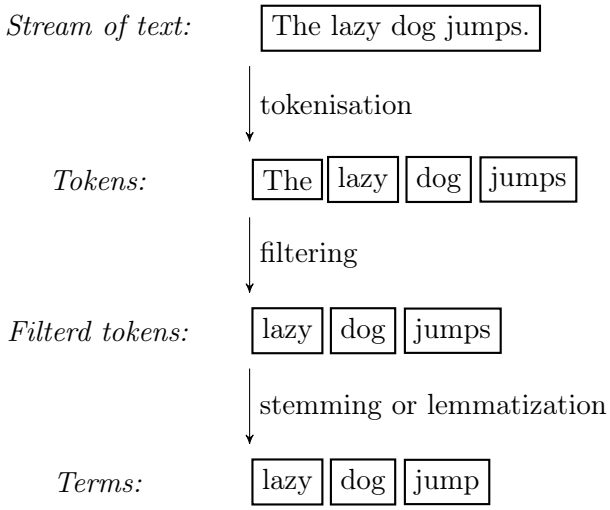


Figure 3.1: Preprocessing of a text stream. The text stream gets tokenized and filtered to process the stream into terms. Terms are the basic unit in the vocabulary of a corpus, which gets counted and leads to term frequencies.

applies a full morphological analysis to accurately identify the lemma of a word, but it comes at a cost of computation time [75]. Unfortunately, lemmatisation was not possible with the current software stack, especially with Elasticsearch. There are some attempts for lemmatisation in Elasticsearch[2], but at the time of writing they do not apply to our Elasticsearch version.

In this section we have only scratched the surface of the issues in linguistic preprocessing. For a more in depth discussion on this topic we refer to [75].

3.2 Frequency analysis

The frequency analysis enables the discourse analysts to have a detailed view on absolute and relative frequencies (relative to the total term frequency) of lexical items. Often the analysts are also interested in consecutive words or a sequence of lexical items, also called *N-grams*.

Diachronic corpora are wide-spread in historical linguistics and become more attractive for discourse analysis. With diachronic studies, it is possible to track the effect on the discourse of social, cultural or political changes over time. In analysing diachronic corpora, it is necessary to characterise the development of words over time, which raises the question of the trend of the data. The easiest approach to this is to look at the rank-order correlation. Does the sequential order of relative frequencies correlate with their ranking? A measure for correlation are coefficients such as the Pearsons coefficient or Kendalls-τ [56]. Since the former is more sensitive to outliers than the latter, Kendalls-τ would be more appropriate to indicate a trend or more precisely whether the frequency of a word is increasing or decreasing over time.

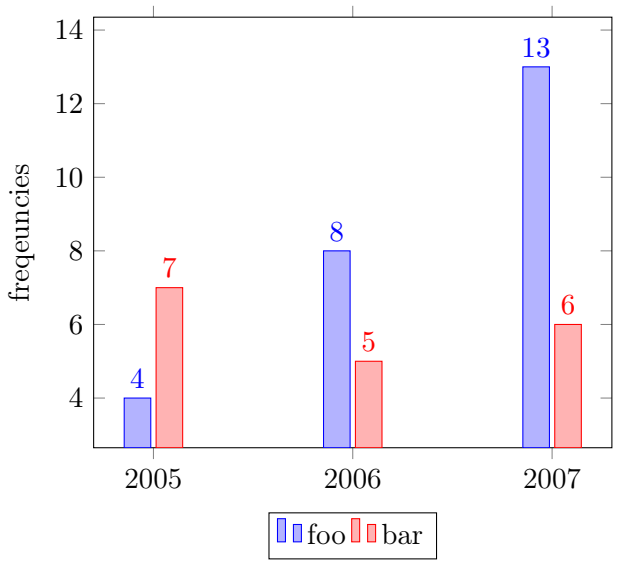
Visualizations on the effect of Kendalls-τ for diachronic corpora have been already performed, as well as two more powerful analysis methods for diachronic corpora ("Variability-based

[2] e.g. `https://github.com/jprante/elasticsearch-analysis-baseform`

neighbor clustering" and "Iterative sequential interval estimation") [49], but the last two go far beyond this thesis.

rank	keyword	frequency	Kendall's τ
1	foo	25	1.00
2	bar	18	-0.33
3	foobar	12	0.00
4	fubar	4	0.82

(a) The frequency table displays the keyword and their frequencies and their correlation coefficient.



(b) Diachronic view of the terms "foo" and "bar".

document	keyword	frequency
article 4	foo	3
article 5	foo	1

(c) Frequencies of terms "foo" in year 2005.

Figure 3.2: Web-flow for the frequency analysis. The frequency table (a) gives a good overview of the most frequent words. They could be further explored in the diachronic view (b) as well as the word frequencies for each period (c).

In LeMATo, the frequency analysis reveals the frequency distributions of words (see Table 3.2a). LeMATo also shows trends in diachronic corpora for every use of a word with the use of Kendalls-τ. Furthermore, the results are supplemented with a visualization (i.e. bar chart) on relative frequencies of selected words in a diachronic manner (see Figure 3.2b). In a table, we show every use of a selected term. The table provides a link to the term occurrence (i.e. the document) and shows the term frequency (see Table 3.2c). LeMATo also highlights the word of interest within the text.

3.3 Concordance analysis

The concordance analysis, sometimes also called KWIC analysis, reveals every use of a keyword and its particular context. KWIC programs just search a text for every use of a particular word or phrase and print out all of the hits within their contexts. This approach is helpful in the preparation for the qualitative interpretation [33]. There, some example keywords are used to query the corpus and examine categories, which serve as a starting point for grounded theory, content analysis or analytic induction.

An article entitled "Deconstructing Development Theory: Feminism, the Public/Private Dichotomy and the Mexican Maquiladoras" by Joanne Wright (1997) [122] demonstrates an example for the KWIC analysis. There, the meaning of the word "deconstruction" as used by the author is examined. Wright found the word "deconstruction" and its relating meaning to a tool, a process of analysis, the results of an analysis and a theory. The different meanings enable the qualitative researcher to make a comparison of the term with its use by others. It is also possible to ask others to sort the hits of the KWIC analysis into piles to verify whether an interpretation is idiosyncratic or not.

There are some nice visualizations to KWIC analysis, e.g. the word tree [119] or the double tree [21] an improved version of the word tree, but they lack the simplicity of integration in other projects. Google Charts offers a double word tree[3] that visualizes one direction and can be easily integrated in other projects. The word tree visualizes the bodies of the text in branches. A big font size within a branch indicates a frequent use of the highlighted term (see Figure 3.3). At the time of writing we found a bug for nested phrases[4], but due to the rare occurrence of nested phrases, it will not affect design decisions. The visualization lacks in showing terms which are characteristic of the environment of the keyword, but these terms are revealed in the co-occurrence analysis (see Section 3.5).

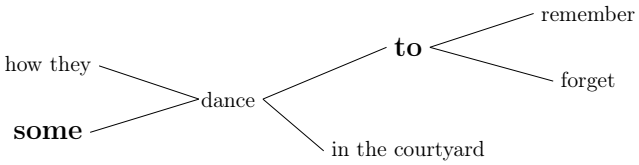


Figure 3.3: Word tree for the concordance analysis of the word "dance" in the following three sentences: "how they dance in the courtyard", "some dance to remember" and "some dance to forget". The words "some" and "to" have a larger font, because they appear two times in the text.

LeMATo provides a KWIC analysis where every use of a queried word is listed with its specific context. We use Google Charts' double word tree for a supplementary visualization.

3.4 Analysis of characteristics in sub-corpora

The analysis of characteristics in sub-corpora tests whether lexical items are significant within a sub-corpus in contrast to the rest of the corpus. Therefore, we consider those words which are

[3] `https://developers.google.com/chart/interactive/docs/gallery/wordtree`
[4] `https://code.google.com/p/google-visualization-api-issues/issues/detail?id=1836`

overrepresented in each sub-corpus. Currently, the existing programs use different diagnostics to achieve this. All these diagnostics calculate on the ratio of the absolute frequencies in different sub-corpora in contrast to the rest of the corpus. Lexico3 [67] uses the Characteristic Element Diagnostic (CED) statistics (see Section 7.2.1 for details and references), which calculates the specificity on hypergeometric assumptions. WordSmith [112] uses a statistic test for the calculation of its keyness value by either chi-square test with Yates correction or Ted Dunnings log likelihood (see Chapter 7 for details and references).

The social scientist is also interested in the frequencies of the significant words in the sub-corpora. In considering visualization issues, we have a list of words with two features, the frequency and the significance. In addition to that, such word lists are often grouped manually into thematic clusters to provide a better clarity on the terms [78] [16] [109]. We can assist this grouping with a computational approach. Therefore, we make use of the co-occurrence to perform a clustering algorithm on the significant terms. Clustering as an unsupervised learning algorithm uses the corpus itself for the source of its hypothesis and is, therefore, an entirely *corpus-driven* approach. We describe our clustering approach, where we group the words into thematic groups, in Chapter 5.

The studies mentioned above, often make use of a circular depiction to visualize the results of such an analysis. As an example take Figure 4.3 on Page 25. The results contain term frequencies significance and a grouping into thematic groups.

LeMATo uses a simpler visualization technique (i.e. scatter chart), where one axis indicates the raw frequency counts and the other axis displays the significance of the terms. It is also possible to zoom in a region of interest, to increase the legibility. We discuss the issue on grouping and its visualization in Section 3.6.

3.5 Co-occurrence analysis

Previously in Section 10.4, we introduced the concordance analysis, which reveals every queried word in its context. Unfortunately, the concordance analysis does not show significant terms of the context to a queried word. The co-occurrence analysis addresses this particular issue. In our reference study [78] each occurrence of a queried term and its words within a word window (e.g. ten words to the left and ten words to the right of the queried word) defines the context or environment of the word. Here, in LeMATo, we take the same text unit of a queried word. We form a sub-corpus from the terms in the environment and compare them to the rest of the corpus to calculate their significance. LeMATo uses text unit frequencies (e.g. document or sentence frequencies) instead of term frequencies, we compare these two approaches in Section 11.2.

The result of the co-occurrence analysis has the same features as the previous analysis, the analysis of characteristics in sub-corpora, which are the frequencies of text units, their significance and a grouping of these terms. Therefore, we are using the same visualization technique here. In the literature, we also find a circular depiction for the presentation of such features (see our reference study at Figure 5.1 on page 28).

3.6 Grouping support

This thesis is also an attempt to the approach of grouping terms into thematic clusters. Grouping of words according to their topic is indeed a subjective approach, and there is no accepted rule how to put such words into thematic clusters. Therefore, we do not perform a tight grouping. Instead, our approach is focussed on the support of manually clustering of words. We are doing this with a dendrogram, where the co-occurring words are neighbours.

We are building a dendrogram in six consecutive steps:

1. Define sub-corpus
2. Calculate significant terms
3. Take top N significant terms
4. Span a vector-space with these
5. Cluster the vector-space
6. Visualize the dendrogram

In LeMATo we have three different text entities: documents, paragraphs and sentences. We have enriched the text units with the publish date gained from LexisNexis [3]. In addition to that, the user defines his own tags to distinguish the text units in different sub-corpora. Through that, we enable a *sub-corpus definition* on certain periods, manually defined labels (i.e. tags) or text entities containing a queried word. The *calculation of significant terms* is based on the comparison of word lists in the sub-corpus to the rest of the corpus. To prevent an overloaded dendrogram, we *filter for top N significant terms*, where N is either 10, 20, 50 or 100. The *vector-space* is a multidimensional space, which is build by the co-occurrence of the words (see Section 5.2). To observe closely related terms, we *cluster* (see Section 5.3) the terms and build a *dendrogram* (see Section 5.3.2).

3.7 Micro analysis

The lexicometric approach (i.e. macro analysis) establishes the relationship, the regular use of lexical items and their distribution in the text. This corse grained view gives an idea of the meaning within the text, but the quality of the relationship among these lexical items is still unclear or can only be revealed with an enormous effort of looking at every single relationship. Therefore, lexicometric approaches (i.e. macro analysis) are often combined with methods of qualitative research to do the statement analysis (micro analysis). The fundamental approach for the macro analysis consists of the following four steps [33]: (1) frequency analysis, (2) concordance analysis, (3) analysis of characteristics in sub-corpora and (4) the co-occurrence analysis. With these four steps, it is possible to examine the occurrence, distribution or even the co-occurrence of words.

The macro analysis does not examine the connection between the words in any way. For example, if we know, that “elderly” and “internet” are very frequently co-occurring, then we neither could say the elderly use the internet, nor they lack access to the internet, nor they abuse the internet. The micro analysis addresses this problem and reveals the heterogeneity

of the discourse and has its origin in the French discourse analysis and enonciative linguistics (from the French word énoncé, see Section 4.1.3 for details and references).

The micro analysis enables the researcher to establish a connection between single comments and discursive contexts. The main focus here is the pattern of the statements within their linguistic form, which guide the interpretation. Such enonciative markings help the social scientist to connect the utterance in the text with a context to reveal its discursive structure. One statement can be interpreted differently by each speaker and, therefore, the micro analysis is not a method to extract the meaning. It is a method to describe the characteristics of an utterance. There are three different enonciative markings, which analyse "the utterance" on a different level. *Deictic words* mark the utterance in a temporal or spatial way. The reference to different speaker positions is the target of the *polyphony* and the argumentative connections of suppositions which form the understanding refers to the *preconstruct* [78, p. 130-] [48].

Deictic words are words which produce subjectivity in the text[5]. Examples of deictic words are words like: "I", "now" or "here". These words refer to persons, locations or time in their discursive context. In our reference study [78] the analysis of deixis enables the disambiguation between "own" and "foreign" in analysing words like: "I"/ "you" and "we"/"you". Furthermore, there is a distinction between primary, secondary and tertiary deictics [78, p. 135-]:

- *Primary deictics*: "I", "here", "now",
- *secondary deictics*: "we", "you", "us", "he", "she", ...
- *tertiary deictics*: proper nouns of persons or places.

Temporal adverbs like "yesterday", "last year" or "soon", or even spatial adverbs like "near", "far", "close to", "left" are also tertiary deictics, because they refer to a certain time or place. Temporal discourse models could detect these temporal adverbs [73].

It is noteworthy that some of the words, like "I" or "we", could occur in the text, without affecting any discourse relation, especially in newspaper articles where they refer to a smaller context without any speaker position. Therefore, it is not easy to detect such words within an automated process and requires a discourse analyst in most of the cases.

Preconstruct is a concept described and developed by Pêucheux [92]. This concept refers to social and institutional conditions in which a discourse emerges. In other words, an utterance uses knowledge whose origin is elsewhere and this knowledge is called the preconstruct. As an example, we have the sentence "The expansion of the airport in Frankfurt is necessary to be internationally competitive.". This utterance includes some normatives, which form the understanding of the sentence. One is that there is a competition between airports and the necessity for expansion.

A characteristic for the detection of a preconstruct is a certain ending like "-ism" (e.g. capitalism, liberalism) or relative clauses, however the preconstruct plays no central role anymore in the practice of discourse analysis today [2, p. 51].

[5]Not to be confused with discourse deictics [120]. Discourse deictics refer to a previous or upcoming term or text segment (e.g. anaphora or co-reference resolution).

Polyphones date back to the original work of Oswald Ducrot in 1972 [29], who is well known in critical French discourse analysis. The general assumption is that texts convey several perspectives from different sources. In other words, in the same text there are several speakers and therefore, the text is polyphonous. These polyphones enable the discourse analyst to detect contradictions or breaks within a discourse. There are some indicator words for such breaks (e.g. "no", "however", "because", "but", "true", "perhaps", ...). These indicator words are also called sentence connectors in linguistics, but not all sentence connectors induce a contradiction or break in a discourse. Part-of-Speech tagging might detect sentence indicators, but there is no discussion break for sentence connectors like "in addition", "furthermore" or "moreover". The sentence connectors put an utterance in an argumentative relation to another utterance. Finally, there are also adjectives or verbs which include a negation: "indecendant", "unavoidable", "undoubtedly" or "counter". For more details on polyphones and their exact definition we refer to [78].

Issues in detecting enonciative markings

There are some known standard solutions in Natural Language Processing (NLP), which might help to address the problems which arise in micro analysis (see Section 3.7). Such standard solutions typically serve a well-defined problem setting, a standard metric for evaluating the task, or even standard corpora, on which the task can be assessed. In the following, we provide a list of standard NLP candidate problems with a brief explanation and evaluate their use to detect enonciative markings.

Discourse analysis identifies the discourse structure of a connected text by textual coherence relational structures. These coherence relations, also known as discourse relations, specify the relationship between sentences, as an example, in the two sentences: "The bartender hid John keys." and "He was drunk.". The coherence relation between these two sentences is an explanation. Examples of other coherence relations are elaboration, result, parallel and occasion. By the text flow, it is desirable to build up a tree, which has the sentences as leaves, and the branches are the coherence relations.

The relation structure in the tree does not necessarily depend on the coherence structure, and it is also possible to describe the time dependent flow of events (by connector words) or even a hierarchical coherence structure (cf. Rhetorical Structure Theory (RST) [74]).

Another possible task in discourse analysis is recognizing and classifying speech acts, within a piece of text. Speech acts here, refer to content questions, statements or assertions within a chunk of text, but these are not the same speech acts defined by Foucault.

There is definitely a potential to apply RST on the text, to highlight contradictory breaks or polyphone structures within the raw text. A discourse tree is a nice visualization aid, but it is not always practical to abstract the raw text into another view, because the reader needs to be accustomed to it.

Named entity recognition (NER) determines which entities of the text map to proper names and also gives inference about their types, whether it is a person, organisation, location or temporal expression. The detection of temporal expressions could be also seen as a separate task since there are corpora for the evaluation of Temporal Expression Recognition

and Normalisation (TERN). NER could be used to detect the proper names in the tertiary deictic expressions (see Section 3.7).

Part-of-speech (POS) tagging annotates a sentence with part of speech tags (e.g. noun, verb, adjective, adverb, modal verb, connector word or pronoun). There is some potential for deictic expressions in detecting personal pronouns ("I", "we"), or adverbs ("here", "now") but there are a lot of terms within a POS-tag, which do not refer to any discoursive structure ("he", "it", "soon"). There is also no POS-tag which marks any negation in the case of polyphonous structures, whether to indicate negating connector words nor for detecting negating adjectives or verbs.

Sentiment analysis (SA) extracts subjective information to determine polarity about specific objects. It is especially useful for identifying trends of public opinion in social media for the purpose of marketing. The polarity in a given text could be classified in emotional categories (e.g. "happy", "sad" or "angry") or ratings (e.g. guessing the star rating for restaurants on the basis of a textual description). Another research direction in sentiment analysis is the subjectivity/objectivity identification. This identification technique classifies a text whether it is subjective or objective based on context and individual sentences (e.g. news article quoting the opinion of people). There are some potentials for lexicometrics, but in an out-of-the-box classification there is too little freedom for the researcher. For the fine-grained analysis an annotation scheme is more desirable.

Final notes on enonciative markings Finally, there is no standard NLP-method to detect enonciative markings, but there is some potential in coherence structures or even in sentiment analysis, however it is far beyond this thesis to create a simple colour marking.

This thesis is more focussed on the lexicometric statistics rather than the enonciative markings, therefore we do not support enonciative markings in LeMATo. Nethertheless, enonciative markings are necessary to perform a critical discourse analytic study.

Chapter 4

Discourse theory

> The meaning of a word is its use in the language
>
> Wittgenstein, 1953

Discourse analysis is a general term to analyse written, vocal, or sign language. Discourse analysis here refers to Critical Discourse Analysis (CDA). CDA is an interdisciplinary approach which views language as a form of social practice. Moreover, CDA connects text to historical or social context. The focus of discourse analysis is to *build up meaning* in larger communicative rather than grammatical units. Furthermore, discourse analysis aims to reveal connections of text and speech to institutional structures [25].

4.1 Post-structuralism

4.1.1 Structuralism

Discourse analytical approaches from writings of Michel Foucault are known as social constructionism or post-structuralistic theories. These principles evolved among other things from the linguistic turn, a significant development in Western philosophy during the early 20th century. The linguistic turn focused primarily on the relationship between the construction of knowledge and language [99]. Decisive for the linguistic turn in humanities were the works of the structuralist Ferdinand de Saussure. He is regarded as the founder of modern linguistics. His lectures about important principles of language description were collected and published by his pupils posthumously in *Course the linguistique générale* in 1916 [108]. In this work, he lays out the foundations for a particular view on language. He sees language as a system of signs, where a sign is an inseparable dichotomy between a *signifier* and a *signified* (Figure 4.1). The signifier (French *significant*) is the "shape" of the word represented by a sequence of graphemes (letters) or phonemes (speech, sound). The signified (French *signifié*) is the mental concept, ideation or object that appears in our minds when we hear or read the signifier.

There is no natural or internal connection between signifier and signified, the relation between

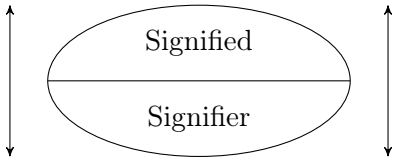


Figure 4.1: The Saussurian sign.

those two is purely arbitrary [108]. The arbitrariness is an important principle in structuralist theory since it guarantees that no extralinguistic factor influences the constitution of signs.

> « . . . la langue est un système dont tous les termes sont solidaires et où la valeur de l'un ne résulte que de la présence simultanée des autres. . . »
> '. . . language is a system in which all terms are fixed, and in which the value of one is only the result of the simultaneous presence of the others. . . ' [108]

According to Saussure's quote the meaning of a word is purely relational and defined through differences to all other words. For this master thesis, this finding is important for two things: Firstly it sets the basis for the understanding of language in discourse analysis (see Section 4.1.2). The relation between language (signifier) and reality (signified) is arbitrary, and it implies that language as a system is not determined by the reality to which it refers. Secondly, the Saussureian view on language has been further developed by Zellig Harris and is the foundation in his distributional theory (see [47]).

4.1.2 Language as a representational system of society

The finding that the context defines the meaning of a word has important implications for discourse analysis. Discourse analysis analyses signs and their relation to other signs, in other words it analyses a representational system. The most important representational system is language, but every other sign system could also form a representational system (e.g. architecture or pictograms). The rule of the arbitrariness of a sign states that the relation between sign (signifier) and its meaning (signified) is neither natural nor a priori given. The relation between these two, as well as the representational system, is fixed by a social convention. In analysing the representational system, then discourse analysis is able to argue in society, which has built the representational system [46, as cited in 32]. Michael Foucault calls the structure of the text, where the order of society is evident, as "the order of discourse" [39, as cited in 32]. This concept enables discourse analysis to reflect with text about social ideologies and their relationships.

4.1.3 French discourse analysis

The French discourse analysis has its origin in post-structuralism, an intellectual movement in 1960 which has its roots in structuralism (see Section 4.1.1). Post-structuralism is characterized by the work of a relatively small number of writers. The writers were leading experts in different disciplines: Jacques Derrida (philosopher), Michel Foucault (social theorist), Gilles Deleuze (philosopher), Jacques Lacan (psychoanalyst and psychiatrist) and Jean Baudrillard (sociologist and philosopher) to name the most formative ones of post-structuralism.

A central element for discourse analysis in post-structuralism is Foucault's work "Archeology of knowledge" [40]. In this work, he describes a concept to uncover the rules of formation of discourses or discursive systems. He defines discourse as "an entity of sequences, of signs, in that they are enouncements" (from the French word énoncés, meaning "the statement" or "utterance" between and among objects). The terms "enonciative linguistics" or "enonciative markings" are often used in this thesis. These terms refer to the term énoncés, which was used by Focault and introduced by Pêcheux [92].

Hence, a discourse[1] is composed of a sequence of semiotic signs, which describe the relation between and among objects, subjects and statements [40]. In other words, discourse must no longer be considered as signs referring to representations, but as social practices that shape the objects which are mentioned in the discourse.

Foucault's influence on discourse analysis is more indirect, the greater influence on French discourse analysis comes from Michel Pêcheux. His most important contribution to discourse analysis consisted of the development of tools for conducting empirical studies of discourses, which he called automatic discourse analysis (AAD, "Analyse automatique du discours") [92]. The true innovation in AAD was the theory of "interdiscourse" (see [92] for details), which makes him an important contributor to discourse theory and discourse analysis. In summary his work is an attempt to provide a scientific instrument for discourse analysis, that serves as means for the researcher to get rid of the subjective readings of texts. Therefore, Pêcheux can be considered as a cornerstone in modern lexicometrics [48].

4.2 Discourse methodology with lexicometrics

Discourse analysis with lexicometrics has been performed in several studies[2] until now. To sketch the methodology of a lexicometric analysis applied in a human geographic study, we outline the approach of "Die neoliberale Stadt" [78]. In this study lexicometrics explores the great structures of language in society to infer about urban development and city marketing. In a first step, the hegemonic pattern of speech for each city is captured using a lexicometric analysis based on national print media. In other words, the most common terms and themes which are associated with each city are examined. Therefore, they used the database *Lexis-Nexis*[3] to extract all articles of four newspapers[4] in the years of 1999 to 2005, where the word "Stadt" [5] and one of three German city names[6] co-occur within a window of ten words. This results in a corpus size of 2.3 million words. The corpus contains three sub-corpora, one for each city. The split into sub-corpora makes it possible to examine which terms and themes tend to co-occur significantly more often than in other cities.

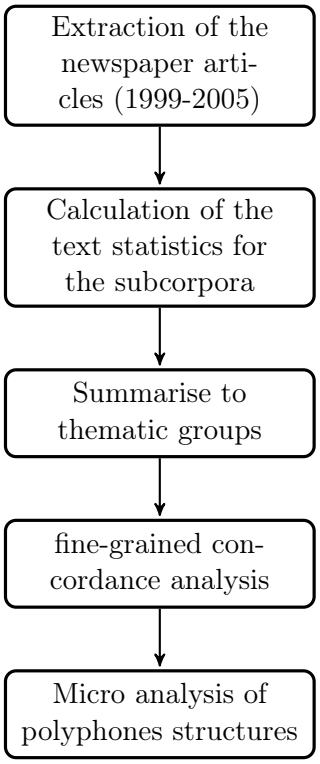


Figure 4.2: Workflow performed in "Die neoliberale Stadt" [78].

4.2.1 Lexicometrics as macro analysis

They used *Lexico3*[67] to calculate co-occurrences, which uses the the ced-value (explained in Section 7.2.1). The ced-value is used to determine all terms which have a significantly higher frequency in contrast to the other two corpora. The terms got filtered by the ced-value between 5 and 20 in the sub-corpus, to reduce the bias of single results. These lists are saved as Excel sheets and each sheet contains between 461 terms ("Köln") and 486 terms ("Leipzig"). Some non-relevant occurrences (proper nouns, single events, etc.) were deleted.

The rest of the words are summarised to thematic groups according to "heuristic cooccurrence fields" (in German: "heuristische Kookkurrenzfelder") [10]. A famous quotation of John R. Firth is put here in a slightly different context.

> "You should know a word by the company it keeps" [10].

In this approach [78] looks for shared characteristics of all occurrences. The formation of the groups depends on presumptions of the analysis (e.g. analysis interest, linguistic model, application context, etc.) [10]. Mattissek emphasises that no predefined categories are used and all the groups evolve from the empirical material, namely the text itself, and that is why it does not violate the structuralism or post-structuralistic paradigm [78].

[1] Unlike semantics, where a discourse is defined as a conceptual generalization of conversation.
[2] [121], [78], [32] and [109]
[3] Online database LexisNexis `http://e-solution.lexisnexis.de/KSH/de/index.html`
[4] "Süddeutsche Zeitung", "die taz", "Der Spiegel" and "Der Stern"
[5] "Stadt" is the German word for city
[6] The analysis is done for three cities: Frankfurt, Köln and Leipzig

This lexicometric approach reveals which terms are in rare and which terms are in abundant use, but it lacks providing insights to the relationship of terms used in a corpus. Even if one would know that these terms are used in the same context with one of the cities, it is not possible to say how they are connected. This connection should be revealed in the following *micro analysis*, which is also cross reading for the heuristic cooccurrence fields. In this step, the most important heuristic cooccurrence fields and their terms are retraced in the text using the *concordance analysis* (see Section 10.4). The text passages are written down to exemplify their use and for further examination of the polyphonic structures of the text. Polyphonic structures reveal breaks in discourses (see Section 3.7).

It should be pointed out, that it is not the intention of the study to examine the authors and their opinion on the different city, it is more about finding patterns of language when talking about one of the cities. For the analysis of the newspaper articles, the micro analysis is used. This enables to describe heterogeneities and contradictions of text passages and their use in a discoursive context.

Lexicometric results

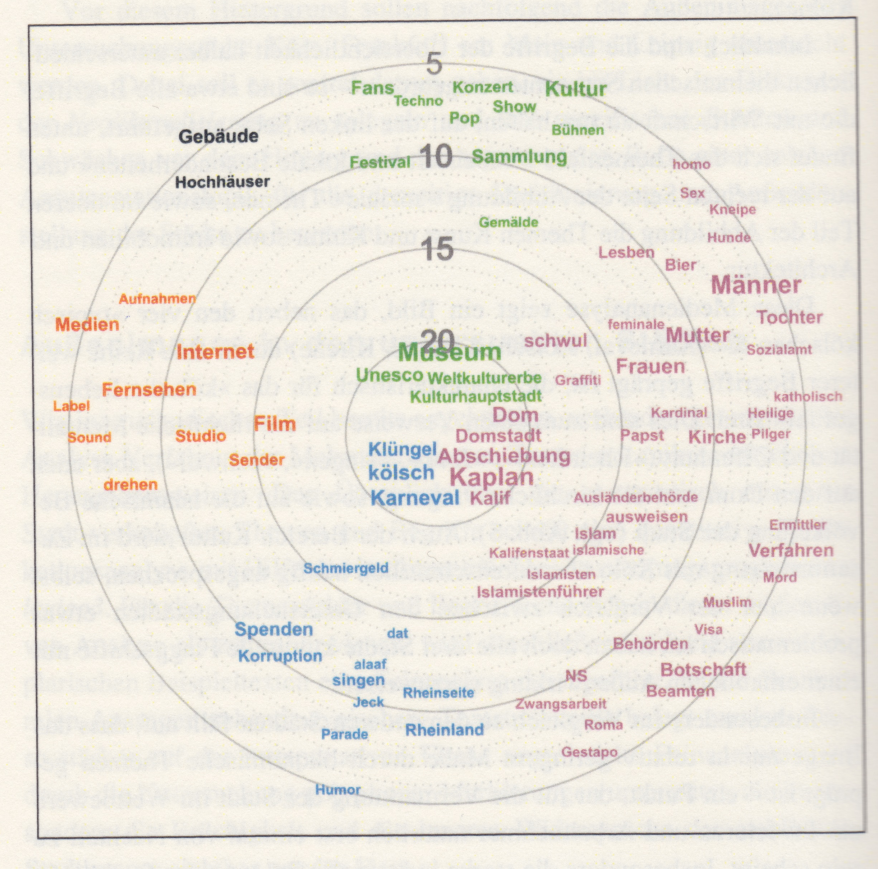


Figure 4.3: Selected collocations with "Köln". The closer a word is placed in the centre, the higher the ced-value. The font-size corresponds to the word frequency. Figure taken from [78]. Mattissek permits us to distribute the figure through this master thesis.

There are two fundamental questions in this reference study we are looking at. First, which

themes and terms are used in the context of each city and second, which keywords and reference structures form the neo-liberal discourse in which the city marketing discourse is embedded [78]. The approach to the first question is outlined in the preceding Section 4.2 and depicted for the city "Köln" in Figure 4.3. To answer the second question, it is inevitable to use a much larger corpus, in this case the DeReKo is used [64]. The DeReKo is with two billion words[7] a representative sample of the German language. The goal is to extract typical collocations and usages of the words in the neo-liberal discourse. These collocations are a good indicator for discursive connections [78, p. 126]. The main way to access the DeReKo is to use COSMAS II, the *COrpus Search Management and Analysis System*. Collocations of the heuristic co-occurrence fields are performed using COSMAS II. The collocations in COSMAS II calculate a log-likelihood ratio (LLR), which makes the collocations comparable. Figure 4.4 shows collocations of words, estimated with the LLR.

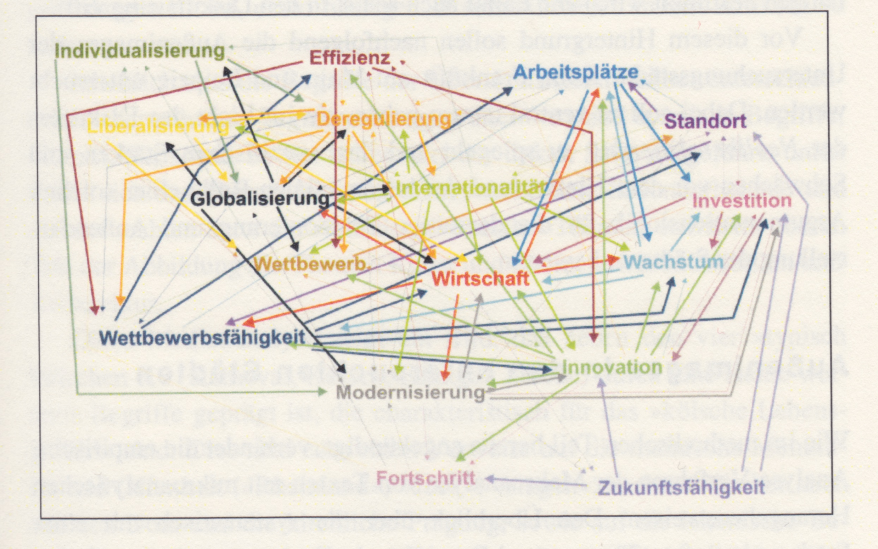


Figure 4.4: Collocations of keywords referring to the city image marketing. The thicker the arrows, the more often terms co-occur. This figure is made by hand. Figure taken from [78]. Mattissek permits us to distribute the figure through this master thesis.

[7]The study is performed with two billion words, but the corpus size of DeReKo grows rapidly. In 2014, the DeReKo contains over 24 billion words [65]

Chapter 5

Finding themes

> You shall know a word by the company it keeps
>
> John R. Firth, 1957

There are four different analyses implemented in LexicoMetric Analysis TOol (LeMATo). Two of these methods, namely the *analysis of characteristics in sub-corpora* and the *co-occurrence analysis* contain a grouping of words into thematic groups. Until now, this grouping was done manually for both analyses in two different studies. "Die neoliberale Stadt" [78] performs a co-occurrence analysis for three different sub-corpora, one for each German city. The manual grouping is performed in the following way: the context of each term is examined with the concordance analysis and then grouped intuitively according to the thematics by the social science researcher (see Figure 5.1) and "Die diskursive Konstitution von Großwohnsiedlungen in Frankreich, Deutschland und Polen" [16] analyses characteristics of a sub-corpus and categorises each significant term manually into thematic groups (see [16]).

In this thesis, we provide a visualisation aid to support the manual grouping of significant words. It is not our goal to force a tight grouping into categories, since this form of grouping is a purely objective task which is performed in different ways by different persons. With our visualisation it is possible to provide a bird perspective of the significant words within the articles of interest and how these significant terms co-occur.

5.1 What is a theme?

Social scientists use some different names to put words into categories. Grounded theorists use terms like "categories", "codes", "labels", "expressions" or "thematic units". We call them "themes", because these groups arise from the context of the words used and form a kind of thematic category. Sometimes we use the word "clusters" interchangeably because these themes are outcomes of a clustering algorithm.

In qualitative data analysis, there are several approaches and methods to find topics. We want to emphasise here, that it is not our goal to replace any method in qualitative data analysis, which is focussed on finding themes. This approach is intended to support different methods of

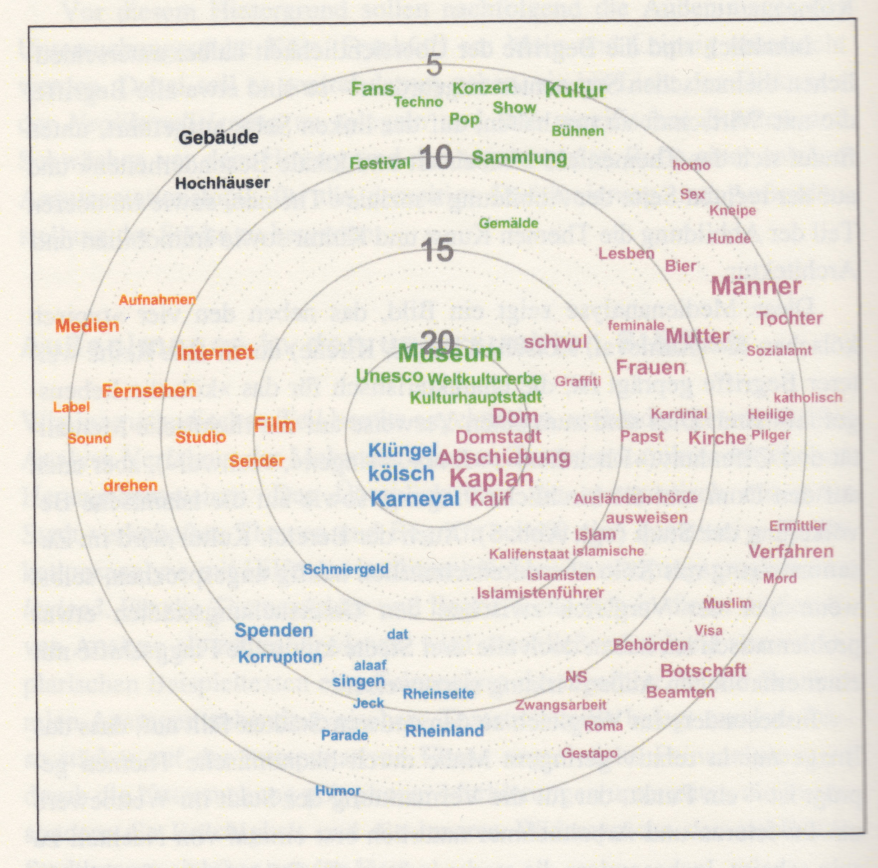


Figure 5.1: The aim of this grouping is to increase the legibility for the reader. There are media related terms on the left and social themes to the right. Terms are the more significant (CED value, see Section 7.2.1), the more they are placed to the center. Figure taken from [78]. Mattissek permits us to distribute the figure through this master thesis.

qualitative data analysis. In a first step, we take the top N significant terms as our candidate words. Our theme finding approach uses the vector space model [106], which is built with a co-occurrence matrix of the candidate words. The vector space model allows us to search the context of each candidate word for other candidate words and form a multidimensional space (see Figure 5.2).

The candidate words for the theme finding are derived by one of two possible analysis steps, the *analysis of characteristics in sub-corpora* or the *co-occurrence analysis*. These two analyses differ in their approach. Where the former examines the overrepresented words of the sub-corpus, the latter examines the particular environment of a queried word (e.g. words within the same sentence or paragraph). Both approaches use different statistics, which are state of the art for their purpose. For more details on these two methods, we refer here to Section 3.4 *analysis of characteristics in sub-corpora* and Section 3.5 *co-occurrence analysis*.

1. Selection of a sub-corpus by tag (*analysis of characteristics in sub-corpora*), or querying (*co-occurrence analysis*) the sub-corpus.
2. Calculate significance in contrast to the rest of the corpus.
3. Take the top n words.
4. Build a vector space.
5. Perform a clustering algorithm (i.e. hierarchical agglomerative clustering).
6. Display a dendrogram.

Figure 5.2: Enumeration describing the theme finding process.

5.2 The word space model

The word space model or Vector Space Model (VSM)[1] is an algebraic model for representing text objects as vectors which form a spatial representation of meaning in a high dimensional vector space [101, p. 17-]. In particular, we filter the corpus for our n words of our interest and transform their distribution to an n-dimensional space, namely the word space. The word space mirrors the spatial proximity of the terms hence it reflects the context to all other n-words. Then we perform a clustering algorithm to group the words according to their context (see Figure 5.3).

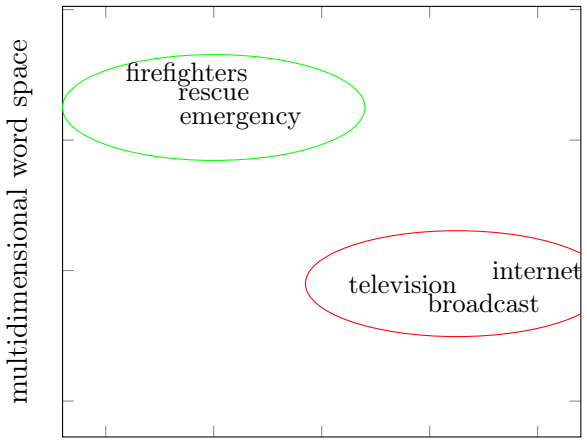


Figure 5.3: A simplified representation of theme finding. Six words are applied to the word vector space according to their spatial proximity within the corpus. A clustering algorithm finds two groups (a fire fighting group marked in green and a media related group marked in red).

5.2.1 Related vector spaces

VSM [106] was developed for the SMART (System for the Mechanical Analysis and Retrieval of Text [103]) information retrieval system and since then it has been applied to text processing in a number of applications. The applications can be divided into three broad classes of

[1] also known as the computational model of meaning [111]

VSMs, according to their matrix form based on term-document, word-context and pair-pattern matrices [117]. *Term-document matrices* allow querying a large collection of documents within a term-document space. Each document is represented by the totality of words and their frequencies (according to the bag-of-words hypothesis [47]), and a document, or even a query, is a dot in a term frequency space. The documents which are closest to the query are the most relevant ones.

The theme finding in LeMATo uses *word-context matrices*. In word-context matrices, the context is given by words, phrases, paragraphs, chapters, documents or other text segments. Word-context matrices are often used in applications where the semantic similarity of words is of interest. There are a lot of different approaches. One is to measure the semantic similarity between two words by the cosine of the angle between their row-vectors [23]. The roots of word-context matrices go back to Zellig Harris. Harris' idea in his distributional hypothesis was that linguistic items with similar distributions can be grouped according to their distributional behaviour, and, therefore, words that occur in similar contexts tend to have similar meanings [47]. One could also see the distributional hypothesis as a justification for the VSM. The context in word-context matrices is occurrences of words within various contexts, such as word-windows [72] or grammatical dependencies [71] [82]. There are different applications for the VSM, most notably the detection of word similarity [23] by comparing row vectors of documents. Other applications are word clustering, word classification, automatic thesaurus generation, word sense disambiguation, context-sensitive spelling correction, semantic role labelling, query expansion, textual advertising, or Named Entity Recognition (NER). For a detailed overview see [117].

The row vectors in *pair-pattern matrices* correspond to pairs of words (e.g. mason: stone, carpenter: wood) and column vectors correspond to patterns in which the pairs co-occur, such as "X cuts Y" and "X works with Y". The purpose is to identify the semantic similarity of the patterns, i.e. the similarity of the column vectors.

5.2.2 Hyperspace analogue to language

In this section we demonstrate how to build a vector space according to the representational model of semantic memory, the Hyperspace Analogue to Language (HAL) [72]. In this approach, the authors read the corpus word by word and record each other word of n-words to the left and n-words to the right as co-occurring. For each co-occurrence, they increment the corresponding cell in the matrix. The matrix is direction sensitive, that means that the co-occurrences to the left are registered in the rows and the co-occurrences to the right are recorded in the columns (see an example in Table 5.1). Finally, they end up with a matrix of roughly 70 000 words.

70 000 words lead to a very high dimensionality (described in HAL as $2n$ high dimensional space for pre-occurrence and post-occurrence to the focus word) for the word-space. The VSM as a statistical methodology relies on statistical evidence, and therefore, the accuracy of the word space is directly proportional to the size of the data that is used. At the same time, the co-occurrence matrix gets enormous for any reasonably amount of data, which makes it difficult to design an algorithm which is scalable and efficient.

Another characteristic of the VSM-matrix is that the majority of the cells is zero, because only a fraction of the terms co-occur and the majority of words occur in a small number of

-	binoculars	I	man	saw	the	with
binoculars	0	0	3	1	7	4
I	0	0	0	0	0	0
man	0	3	0	4	5	0
saw	0	5	0	0	0	0
the	0	5	4	7	3	5
with	0	2	5	3	4	0

Table 5.1: Example matrix (HAL) for the sentence "I saw the man with the binoculars" and a window width of five words. Words are weighted inversely proportional to the distance of the focus word.

contexts, regardless of the size of the data.

The problem of the very high dimensionality is usually solved by representing the high dimensional data in a low dimensional space. This technique is called *dimensionality reduction* (see Section 6.3) and reduces both, the sparseness of the data and the dimensionality. The simplest form of it is simply to filter out words and documents by linguistic (e.g. part-of-speech filtering) or statistical criteria.

Once the matrix in HAL is constructed, similarity measures of the Minkowski family (Euclidean and city-block) are applied to the normalized vectors. HAL detects the similarity of word meanings to patterns of vector elements. The correlation between vector similarity and the cognitive effects relies on the 100 to 200 most various vector elements [72]. The matrix used in HAL is also applied to a classification task, where three categories (i.e. animals, body parts and geographical locations) are identified. Their classification is based on inter-vector distances and was able to separate all the words into their distinct classes.

5.3 Clustering

Clustering or cluster analysis is the general task of grouping a set of objects according to their similarity. Each group of similar data objects is called a cluster. Objects within the same cluster are more similar to each other than to objects in other clusters. It is a common technique in statistical data analysis and has been applied in various fields, including information retrieval, image analysis, pattern recognition or bioinformatics.

Clustering is the most common algorithm in *unsupervised learning* and has the purpose of finding hidden structures in unlabelled data. Whereas in supervised learning, (e.g. classification tasks) a form of human supervision is needed to impose on the data, clustering algorithms as a form of unsupervised learning come without such human guidance.

One of the most critical inputs in clustering algorithms is the distance measure. There are several different distance measures for word-context matrices (see Chapter 6). Various distance measures result in a different similarity and, therefore, generate different clusters.

Clustering algorithms could be distinguished according to two properties. The first property is the assignment of a data point to its cluster. *Hard clustering* assigns every data point to exactly one group, whereas *soft clustering* algorithms assign a fractional membership to

each cluster. This assignment is represented by a probability distribution for each data point to every cluster. The second property is how the cluster relates to each other cluster. *Flat clustering* generates a flat set of clusters without any relation to other clusters. *Hierarchical clustering* creates a hierarchy of data points which form each cluster.

5.3.1 Related clustering of word-context matrices

The following algorithms detect the sense of polysemous words[2] by generating different clusters for each sense of the word. The application of these algorithms refers to *word sense induction*. These algorithms use the same matrix form (i.e. word-context matrix) as LeMATo does, but they perform their algorithms in a much higher dimensional space.

- The first approach uses soft hierarchical clustering to row-vectors in word-context matrices, where noun-verb and verb-noun matrices [89] are used. Furthermore, relative entropy (see Section 6.4.2) is used for measuring the similarity of verb-noun and noun-verb row-vectors [89].
- The seminal work of the word sense discrimination model [110] used word hard-flat clustering to row-vectors, where the context was given by a ± 25 word window to the centered word.
- Another approach uses soft flat clustering to word-context clustering, where the context was given by parsed text [83].
- Clustering by Committee (CBC) is a general purpose clustering algorithm. The authors present the algorithm specifically for the automatic clustering of documents and automatic induction for concepts and word senses [84].

5.3.2 Hierarchical agglomerative clustering

In Chapter 6 we define several different ways to measure the similarities (or dissimilarities) between the rows or columns of a data matrix. We want to emphasise again that different similarities build different matrices, which in fact result in different cluster affiliations of the data objects. Similarities of word-context matrices could be visualised in several different ways. The graphical representation of a matrix of distances, which is the easiest to understand, is the dendrogram. A dendrogram is a tree, where the objects are joined in a hierarchical fashion according to their similarities. Here, we describe the algorithm of hierarchical cluster analysis on the step by step guidance of building a dendrogram.

We use the HAL-matrix (see Table 5.1) as a baseline for the hierarchical agglomerative clustering. In reality, it does not make sense to perform clustering on a single sentence. Clustering is usually performed on a massive dataset, but here it is an illustrative example. In a first step, the clustering algorithm compares each row with each other row by the city block distance because it is quick to set up and therefore it is easy to retrace. For this, we need to calculate the sum of the differences for each row and end up with a similarity matrix (see Table 5.2). This city block distance is a symmetric metric, and therefore, the matrix is also symmetric, and we need to look at entries below or above the main diagonal.

[2] A polysemous word has multiple meanings (e.g. a "crane", could be a bird or a construction equipment).

-	binoculars	I	man	saw	the	with
binoculars	0	15	15	20	17	12
I	15	0	12	5	24	14
man	15	12	0	11	16	8
saw	20	5	11	0	19	15
the	17	24	16	19	0	14
with	12	14	8	15	14	0

Table 5.2: Similarities based on the city-block metric between all rows of the HAL-matrix (see Table 5.1 on page 31).

-	binoculars	I/saw	man/with	the
binoculars	0	20	15	17
I/saw	20	0	15	24
man/with	15	15	0	16
the	17	24	16	0

Table 5.3: Similarities are joined with the maximum linkage criterion.

To build up the dendrogram, we need to look for the pair of samples that are the most similar. In our case, these are the entries with the closest similarity, which are the words "saw" and "I" with a value of 5, followed by the word "with" and "man" with a value of 8. These two pairs are joined according to the linkage criteria (see Section 6.5). We are using the *maximum*, or *complete linkage* method to compute the next level of similarities. In this method two pairs are joined according to the maximum similarity, this is the highest value of the similarity matrix of the corresponding cells.

Since there are new pairs "saw"/"I" and "man"/"with" we need to sum their similarity up, calculate the new matrix (see Table 5.3) and find the most similar ones again.

Each agglomeration to a cluster, up the hierarchy level, has a greater distance between the clusters than the previous one in the hierarchy level. A stopping criterion for the clustering is either when the clusters are too far apart to be merged *(distance criterion)*, which automatically finds k (i.e. the number of clusters), or when there are a sufficient number of clusters *(number criterion)*, where k must be given.

Here we continue clustering until all words are clustered. For the complete dendrogram see Figure 5.4.

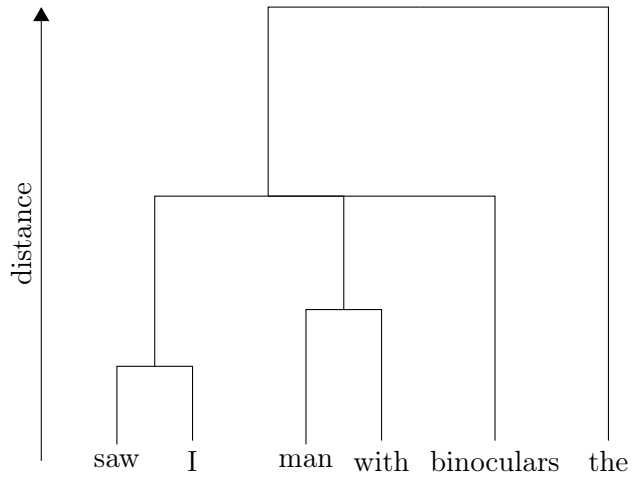


Figure 5.4: Dendrogram for the sentence in the HAL-matrix (see Table 5.1).

Chapter 6

Calculations in the vector space

After the text in LexicoMetric Analysis TOol (LeMATo) has been tokenized, stored in an inverted index database and filtered for the most significant terms, we need to build up the vector space. The general mathematical processing for the vector space contains four steps [117]. The first step is to *generate a matrix* filled with raw frequencies or any other co-occurrence measure (see Section 6.1). The measures within the *matrix* are *adjusted* (see 6.2) because common words might have high frequencies. Hence they are less informative than rare words. Third, the matrix gets smoothed by reducing the random noise in the matrix through filling zero elements in a sparse matrix (see Section 6.3). Finally, in the fourth step, there are many different ways to measure the similarity (or dissimilarity) between two vectors (see Section 6.4).

In addition to that, the vector space in LeMATo gets clustered. The hierarchical agglomerative clustering uses a linkage criteria to aggregate two clusters to one. At the end of this chapter we show the most common linkage criteria (see Section 6.5), which determines the distance between sets of observations as a function of pairwise distances.

6.1 Co-occurrence measures

Co-occurrences are often used as a way to represent the global contexts of words. There are several measures to observe the co-occurrence of words. These measures are often used as a baseline for further post processing (e.g. collocation extraction in LeMATo). To calculate the measures we need to observe the co-occurrence frequencies n_{AB} of two words A and B, as well as the individual frequencies n_A, n_B and the corpus size n.

Raw frequency counts: One possibility is to view each word as a distinct entity within sentences or other easily observable linguistic units, applying the raw co-occurrence frequency counts n_{AB} in a simple vector space model, where each word defines a new dimension. Several variations of this kind of vector space model were proposed. The most famous one is the Latent Semantic Indexing (LSI). It is a well-known technique for document categorization based on the raw co-occurrence frequency counts of words [30]. Dimensionality reduction (e.g. part of speech filtering, Singular Value Decomposition) leads to a performance gain in computational efficiency.

The Dice coefficient compares two different samples [26, as cited in [15]] (see Equation 6.1). It is commonly used in information retrieval [95, as cited in [15]]. The coefficient is defined as twice the shared information (e.g. co-occurrences n_{AB}) over the sum of cardinalities ($n_A + n_B$):

$$s_{dice} = \frac{2n_{AB}}{n_A + n_B} \tag{6.1}$$

The Mutual Information measure [19, as cited in [15]] has been applied to lexicography with a slight modification to its original formula [59, as cited in [15]]. Regarding the preference of low frequent words the *Lexicographers Mutual Information* s_{LMI} added a factor for co-occurrence words n_{AB} (see Equation 6.2).

$$s_{LMI} = n_{AB} \log_2 \left(\frac{n \cdot n_{AB}}{n_A \cdot n_B} \right) \tag{6.2}$$

The log-likelihood test [31, as cited in [15]] compares two binomial distributions with each other using the generalized log likelihood ratio λ, where λ in our case is defined as:

$$\lambda = \left[\begin{array}{l} n \log(n) - n_A \log(n_A) - n_B \log(n_B) + n_{AB} \log(n_{AB}) \\ + (n - n_A - n_B + n_{AB}) \log(n - n_A - n_B + n_{AB}) \\ + (n_A - n_{AB}) \log(n_A - n_{AB}) + (n_B - n_{AB}) \log(n_B - n_{AB}) \\ - (n - n_A) \log(n - n_A) - (n - n_B) \log(n - n_B) \end{array} \right] \tag{6.3}$$

The significance is then computed as:

$$s_{LL} = -2 \log \lambda \tag{6.4}$$

The log-likelihood test is only one-sided, this means that it does not distinguish between significant co-occurrence and non-significant co-occurrence. To correct this, a second significance can be defined:

$$s_{LL2} = \begin{cases} -2 \log \lambda \text{ if } n_{AB} < \dfrac{n_A \cdot n_B}{n} \\ 2 \log \lambda \text{ else} \end{cases} \tag{6.5}$$

Notes on co-occurrence measures: This is not an exhaustive list of all co-occurrence measures. There are also other measures, e.g. the **Poisson significance measure** [50, as cited in [15]], which is favorable in cases where the frequency of the words is much smaller than the corpus size. The **z-score** and the **t-score** (t-test) are also two commonly used measures [38, as cited in [15]].

In LeMATo, currently, the only measure which is implemented is the Dice coefficient because its value is better directly comparable (see Section 6.2). LeMATo is designed to add easily other measures.

6.2 Adjusting the weights

Unfortunately, all co-occurrence measures except the Dice coefficient, are not directly comparable to each other. A higher log-likelihood ratio does not necessarily mean a more significant co-occurrence. If the frequency of two words is very high, they are also likely to co-occur, but they are less important than the co-occurrence of two words which are less frequent.

For example, in measuring the similarity between words like "Leipzig" and "Frankfurt", the contexts "financial" or "football" are more discriminative than the contexts "have" or "like".

Currently, LeMATo does not support an adjustment of the weights, but the point-wise mutual information transformation is also implemented in the s-space package. This means LeMATo could be easily extended for an adjustment of the weights.

6.2.1 Tf-idf

The most popular way to adjust the frequencies is to use the Term Frequency-Inverse Document Frequency (tf-idf) [52]. It is a commonly used weighting scheme for ranking and scoring documents. It increases proportionally to the word frequency within a document, but drops off proportionally to the overall word frequency and, therefore, it has certain advantages over raw frequencies [105]. It has been applied to stop words filtering in various applications like text summarization or text classification. Usually tf-idf is used with document normalisation (see [114]), because the raw tf-idf favours long documents.

The tf-idf is the product of two statistics, the term frequency tf and the inverse document frequency idf. There are various ways to determine the exact values for both statistics. For example, the tf value could be raw frequencies of a document, a binary value or even a log-normalised value. The idf value is usually defined as $\log(\frac{N}{n_i})$ where N is the total number of documents and n_i is the number of documents containing the term i, but there are also several variations.

6.2.2 Point-wise mutual information

We are not focused on document weighting, therefore tf-idf is not appropriate. A better fit for the solution to our problem, of adjusting the frequencies for word context matrices, is the Pointwise Mutual Information (PMI) [19, as cited in [117]]. For the context matrix $\mathbf{F}$, with the entries $f_{i,j}$, the row total sum $\sum_{i=1}^{n_r} f_{i,j}$ (n_r number of rows), the column total sum $\sum_{j=1}^{n_c} f_{i,j}$ (n_c number of columns) and the matrix total sum $\sum_{i=1}^{n_r}\sum_{j=1}^{n_c} f_{i,j}$ the PMI $pmi_{i,j}$ is defined as follows:

$$p_{i,:} = \frac{\sum_{j=1}^{n_c} f_{i,j}}{\sum_{i=1}^{n_r}\sum_{j=1}^{n_c} f_{i,j}} \tag{6.6}$$

$$p_{i,j} = \frac{f_{i,j}}{\sum_{i=1}^{n_r}\sum_{j=1}^{n_c} f_{i,j}} \tag{6.7}$$

$$p_{:,j} = \frac{\sum_{i=1}^{n_r} f_{i,j}}{\sum_{i=1}^{n_r}\sum_{j=1}^{n_c} f_{i,j}} \tag{6.8}$$

$$\begin{aligned} pmi_{i,j} &= \log\left(\frac{p_{i,j}}{p_{i,:}\cdot p_{:,j}}\right) \\ &= \log\left(\frac{f_{i,j}\cdot\sum_{i=1}^{nc}\sum_{j=1}^{n_c} f_{i,j}}{\sum_{j=1}^{n_c} f_{i,j}\cdot\sum_{i=1}^{n_r} f_{i,j}}\right) \end{aligned} \tag{6.9}$$

Here, we have the estimated probability of a word that occurs in a context $p_{i,j}$, the estimated probability $p_{i,:}$ of the word w_i and the estimated probability $p_{:,j}$ of the context c_j.

There are some variations to the PMI. One is the Positive Pointwise Mutual Information (PPMI) [81], where all entries less than zero are replaced with zero. The PPMI performs better than a variety of other weighting approaches [18].

Other variations address the problem that PMI is biased towards infrequent events, particularly in cases where the word w_i and the context c_j are statistically dependent (i.e. the word always co-occurs with its context) (see [117] for details).

6.3 Smoothing the matrix

In statistics, to smooth a data set is to create an approximation that captures the essential attributes and patterns in the data, while leaving out the noise and other fine-scale phenomena. When the Vector Space Model (VSM) is applied to a large corpus, this inevitably leads to a large amount of features[1] and also to a large word-context matrix of the same size.

The simplest way to reduce this high dimensional space is to filter the components with no or little information and to keep only the most frequent words. Common words like "the" or "is" carry little discriminative power. Statistical filtering based on the significance gained from the comparison to a reference corpus or any other homogeneous corpus also reduce the large dimension of the feature space (see Chapter 7).

Computing the similarity between all pairs of vectors in the VSM is a computationally intensive task. Only those vectors which share non-zero candidates are compared, otherwise they are dissimilar to each other. Filtering based on the PMI decreases the number of comparisons greatly, while losing little precision for the most 200 similar words [71].

There are also other, very elegant, dimensionality reduction techniques like Singular Value Decomposition (SVD) or Random Indexing (RI). SVD has been performed in Latent Semantic Analysis (LSA) [23]. SVD is a matrix factorization technique that decomposes the matrix into three smaller matrices, which contain the linearly independent factors of the original matrix.

Instead of building the co-occurrence matrix and then extracting vectors from it, RI incrementally accumulates context vectors in a two-step approach. First, each context (e.g. document) in the text, is assigned to a randomly generated representation called the index vector. Second, context vectors are produced by scanning through the text and every time a word appears in a context, the corresponding index vector is added to the context vector. Each word is

[1] 70 000 features for Hyperspace Analogue to Language (HAL), see Section 5.2.2

represented by a context vector, which is a combination of all index vectors of the documents in which the word appears. For more information on RI we refer to [100].

The vector space in LeMATo is relatively small (≤ 100 terms), and these terms also have discriminative power, because the words originate from the significance filtering step. Therefore we go completely without smoothing in LeMATo.

6.4 Similarities and dissimilarities

An accurate measure of distance obeys three properties: symmetry, the identity of indiscernibles and the triangle inequality. *Symmetry* is given when the distance d between two objects a and b is the same as the distance between b and a (see Equation 6.10). The *identity of indiscernibles* says that the distance d of two objects a and b is always positive, with the only exception, that is, iff the two objects are equal, then the distance d is zero (see Equation 6.11). The third axiom is called the *triangle inequality* that states that the distance $\mathrm{d}(a, b)$ is shorter or equal than the distance $\mathrm{d}(a, c) + \mathrm{d}(c, b)$ in the triangle of three objects a, b and c. Equal only, iff object c is on the "route" from a to b (see Equation 6.12).

$$\mathrm{d}(a, b) = \mathrm{d}(b, a) \tag{6.10}$$

$$\mathrm{d}(a, b) \geq 0 \tag{6.11}$$

$$\mathrm{d}(a, b) \leq \mathrm{d}(a, c) + \mathrm{d}(c, b) \tag{6.12}$$

There are some measures, which do not satisfy all the axioms for metrics, especially the triangle inequality is often not satisfied (e.g. Bray-Curtis dissimilarity). These measures are called dissimilarity measures or *dissimilarities*.

6.4.1 Similarities

L_1 distance (city-block)

The name city-block distance[2], considered by Hermann Minkowski, allude to the grid layout of most streets of Manhattan. The taxicab in Manhattan needs to take an angular path to reach the destination on the shortest path. The city-block distance d_1 between two vectors $\mathbf{p}$ and $\mathbf{q}$ is the sum of the differences in the coordinates (see Equation 6.13). For example, in a two-dimensional space, the city-block distance between the point $\mathbf{p}(p1, p2)$ and point $\mathbf{q}(q1, q2)$ is $|p_1 - q_1| + |p_2 - q_2|$ (see Figure 6.1).

$$\mathrm{d}_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i| \tag{6.13}$$

[2]also called Manhattan distance, taxicab distance, rectilinear distance, L_1 distance or ℓ_1 norm

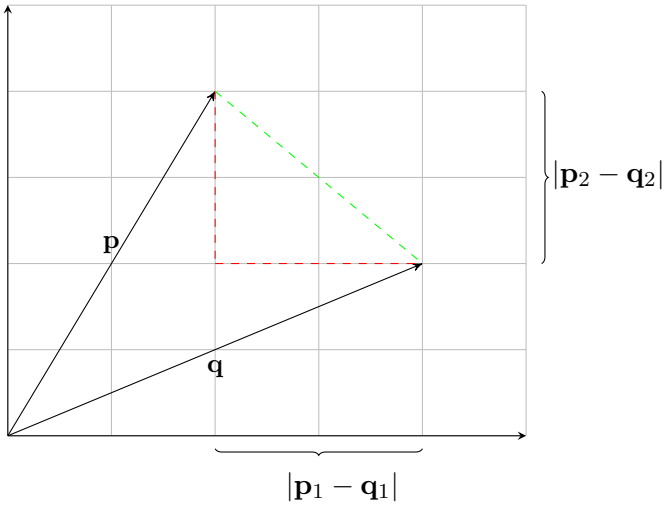


Figure 6.1: L_1 metric (red dashed line) and L_2 metric (green dashed line) in a two dimensional plot.

L_2 distance (Euclidean)

The Euclidean distance, also known as Pythagorean metric, is the shortest distance between two points in the Euclidean space. In an n-dimensional space, the Euclidean distance d_2 between two vectors $\mathbf{p}$ and $\mathbf{q}$ is given by the Pythagorean formula (see Equation 6.14).

$$\begin{aligned} d_2(\mathbf{p}, \mathbf{q}) &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \end{aligned} \tag{6.14}$$

In a two-dimensional space, the Euclidean distance between two points $\mathbf{p}(p1, p2)$ and $\mathbf{q}(q1, q2)$ is given by $d(\mathbf{p}, \mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$ (see Figure 6.1).

6.4.2 Dissimilarities

Relative entropy

The relative entropy[3] is a non-symmetric measure to compare two probability distributions P and Q [63]. The relative entropy of Q from P, denoted as $D_{KL}(P\|Q)$, is a measure of the information loss, when Q is approximated by P (see Equation 6.15). $D_{KL}(P\|Q)$ is zero, if P and Q are equal and it is inverse proportional to the probability of a random sample drawn according to P.

$$D_{KL}(P\|Q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{6.15}$$

[3]also information divergence, information gain, Kullback-Leibler divergence, KLIC or KL divergence

Jaccard index

The Jaccard index, also known as the Jaccard similarity coefficient, is used to compare similarity between finite sample sets, and is defined as the size of the intersection $|A \cap B|$ divided by the union $|A \cup B|$ between two sets (see Equation 6.16).

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{6.16}$$

6.4.3 Implementation notes

The default value for the similarity of two vectors in LeMATo is the L_2 distance (Euclidean), but each other value, which is implemented within the s-space package[4] could also be used. The similarity measure could be changed by modifying the entry in the *application.yml* file.

6.5 Linkage criteria

In the initial phase of hierachical agglomerative clustering (see Section 5.3.2), each observation forms its cluster. Pairs of clusters are merged according to their similarity. Given a distance matrix, linkages between clusters are computed through a criterion, that is the linkage criterion, to compute the distance between groups. In the following, we describe commonly used linkage criteria between two clusters A and B. The distance $D(A, B)$ is the distance between two clusters A and B and the distance $d(a, b)$ is the distance between two elements $a \in A$ and $b \in B$.

These linkage criteria discussed here are not an exhaustive list of all linkage criteria. We only explain the linkage criteria, which are implemented in the S-Space package. For a complete list of linkage criteria for hierarchical agglomerative clustering see [107].

6.5.1 Maximum or complete-linkage clustering

In complete-linkage clustering, also known as the method of farthest neighbour clustering, the link between two clusters is defined through those two elements in the clusters which are farthest away from each other (see Equation 6.17).

$$D(A, B) = \max_{a \in A, b \in B} d(a, b) \tag{6.17}$$

6.5.2 Minimum or single linkage clustering

In minimum or single linkage clustering, also known as nearest neighbour clustering, two clusters are combined according to the nearest elements in different clusters (see Equation 6.18).

[4] see enum edu.ucla.sspace.common.Similarity.SimType in Javadoc `http://fozziethebeat.github.io/S-Space/apidocs/`

$$D(A, B) = \min_{a \in A, b \in B} d(a, b) \tag{6.18}$$

A drawback of single linkage clustering is the chaining phenomenon. Two clusters via minimum linkage clustering may be forced together due to two single elements being close to each other, even if many of the elements in the cluster are far apart. In contrast to that, the complete linkage tends to find compact clusters of approximate equal diameters [37, p. 62-64].

6.5.3 UPGMA

Unweighted Pair Group Method with Arithmetic Mean (UPGMA), also known as mean linkage clustering, is the most popular method in ecology for hierarchical clustering, which takes the average of the pairwise similarities in relevant descriptor variables. The distance between two clusters A and B is taken to be the average of all distances between all possible pairs of objects between $a \in A$ and $b \in B$, that is the mean distance between the elements of each cluster (see Equation 6.19).

$$D(A, B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a, b) \tag{6.19}$$

6.5.4 WPGMC

In Weighted Pair-Group Method using Centroids (WPGMC), also known as median linkage or centroid linkage clustering, two clusters A and B are merged according to their centroids c_A and c_B (see Equation 6.20). The centroid is the positional arithmetic mean (i.e. average) of all objects. It is used only for Euclidean distance. For instance, if the cluster A contains the two objects p and q, the centroid is defined as $c_A = \frac{1}{2}(p + q)$.

$$D(A, B) = \|c_A - c_B\| \tag{6.20}$$

6.5.5 Implementation notes

All four linkage criteria which are listed here are implemented within LeMATo. The default linkage criterion in LeMATo is the mean linkage clustering (UPGMA), because it is the most popular method for hierarchical agglomerative clustering.

Chapter 7

Significant words

A frequency list is a good starting point for studying a text corpus. Frequency-ordered listings give an overview of the most commonly-occurring words in the text. Computers process frequency lists in nearly real time, but the information provided is not easy to read. Therefore, we need a filtering mechanism to reveal significant items, those items which are either over- or under-represented in the text corpus. There are at least two different approaches to this filtering mechanisms: the *frequency adjustment* uses a formula to adjust the frequencies for the distribution of words, and the *statistical test* applies a statistical procedure to find the significant words. One thing which both approaches have in common is the contingency table[1]. This table summarizes the multivariate frequencies of the variables.

	wikipedia corpus (en)	British National Corpus (BNC)	total
foo	4500	40	4540
not foo	$1.9 * 10^9 - 4500$	$10^8 - 40$	$2.9 * 10^9 - 4540$
total	$1.9 * 10^9$	10^8	$2.9 * 10^9$

Table 7.1: An example of a contingency table. The frequencies of the word "foo" are queried in two different corpora. In the wikipedia corpus (en) the word "foo" occurs 4500 times and in the BNC the word "foo" occurs 40 times. The size of the wiki corpus is 1.9 billion words and the size of the BNC is 100 million words. The label "not foo" refers to all other words.

As an example for a contingency table, we queried Wikipedia (en)[2] and the BNC[3] for the word "foo" (see Table 7.1). The wiki corpus is roughly 19 times larger than the BNC. Therefore, we need to adapt the frequencies and apply the cross-multiplication to calculate the expected frequencies (see Table 7.2). This adaptation to the contingency table forms the basis for further calculations (e.g. chi-square test or log-likelihood test).

[1] also known as cross tabulation or cross tab

[2] `http://corpus.byu.edu/wiki/` -2014

[3] `http://corpus.byu.edu/bnc/`

		wiki corpus	BNC	total
foo	observed	4500 (a)	40 (b)	4540
	expected	4313	227	
not foo	observed	$1.9 * 10^9 - 4500$	$10^8 - 40$	$2.9 * 10^9 - 4540$
	expected	$1.9 * 10^9 - 4313$	$10^8 - 227$	
total		$1.9 * 10^9$ (c)	10^8 (d)	$2.9 * 10^9$ (N)

Table 7.2: Contingency table with observed and expected values calculated with frequencies of Table 7.1. The expected frequencies are calculated with a simple cross-multiplication. The expected frequency for "foo" in the wiki corpus is calculated as follows: $c * \frac{a+b}{c+d}$. We assume an equal distribution of the words for both corpora.

7.1 Comparing words using reference corpora with test statistics

7.1.1 Chi-square test

The chi-square test, also referred to χ^2 test is used to determine whether there is a significant difference between observed and expected frequencies in one or more categories. There are some variants of the chi-square test, but if there is no other precluding context, the Pearson's chi-square test [85] (see Equation 7.1) is often meant. The letter "O" stands for observed and "E" for expected values. In our case we have to sum up each of the values using Pearson's equation:

$$\chi^2 = \sum_{i=1}^{N} \frac{(O_i - E_i)^2}{E_i} \tag{7.1}$$

$$\begin{aligned}\chi^2 = \frac{(4500 - 4313)^2}{4313} + \frac{(40 - 227)^2}{227} + \\ \frac{(1.9 * 10^9 - 4500 - 1.9 * 10^9 - 4313)^2}{1.9 * 10^9 + 4313} + \\ \frac{(10^8 - 40 - 10^8 - 227)^2}{10^8 + 227} = 162.16\end{aligned}$$

The χ^2 test statistic can be used to calculate the p-value by comparing the value to a chi-squared distribution. We have 1 degree of freedom, because we have 2 columns and 2 rows, $(2-1)*(2-1) = 1$. Our value is 162.16. After looking up our value with 1 degree of freedom in a chi-square distribution table, we conclude that the observed and expected frequencies are not equal with a certainty of 99.999%.

The test statistic tends to indicate significance even for low frequencies and is therefore not always reliable. For such cases **Yates correction for continuity** [123] is applicable, where the differences of observed and expected values are reduced by 0.5 before squaring:

$$\chi^2_{\text{Yates}} = \sum_{i=1}^{N} \frac{(|O_i - E_i| - 0.5)^2}{E_i} \tag{7.2}$$

7.1.2 Log-likelihood ratio test

Another approach to calculate the significance is the log-likelihood ratio test. There are some issues with the chi-squared test. In a comparison between the LOB Corpus[4] and the Brown Corpus[5], there are many common words marked as having significant chi-squared values, because words are not selected at random in language, so we will always see a large number of differences between two corpora with the chi squared statistics [57].

Another issue is, that chi squared values become unreliable for extreme values. They tend to indicate significance for low values and overestimate high frequency words, when comparing a small corpus to a much larger one. The assumption of a normal distribution causes this effect, and text is not normally distributed. The parametric analysis based on the binomial or multinomial distribution is more suitable for text analysis. An often suggested alternative to the Pearson's chi-squared test is the log-likelihood ratio test proposed by Ted Dunning [31], but it also remains unreliable for rare bi-grams [88].

The log-likelihood statistic gives an accurate measure of how "surprising" an event is [31]:

$$G^2_{dunning} = -2\ln(\lambda) = 2(H(matrix) - H(rowSums) - H(colSums) + H(N)) \tag{7.3}$$

where H is the Shannon entropy of a finite sample and could be written as:

$$H(X) = -\sum_i P(x_i) \ln P(x_i) \tag{7.4}$$

For our contingency table (Table 7.1), we could compute the G^2 statistics as follows:

$$\begin{aligned} G^2_{dunning} &= 2(a\ln(a) + b\ln(b) + (c-a)\ln(c-a) + (d-b)\ln(d-b) \\ &\quad - (a+b)\ln(a+b) - ((c-a)+(d-b))\ln((c-a)+(d-b)) - (c)\ln(c) - (d)\ln(d) \\ &\quad + (a+b+(c-a)+(d-b))\ln(a+b+(c-a)+(d-b))) \\ &= 243.11 \end{aligned}$$

Many programs rely on an even simpler log-likelihood ratio to indicate the over- or under-use of words. Starting point for the calculation is the contingency table. We can then apply the log-likelihood according to the following formula proposed by Paul Rayson and Roger Garside [93]:

[4] Lancaster-OsloBergen Corpus (often abbreviated as LOB Corpus) is a million-word collection of British English texts

[5] Brown University Standard Corpus of Present-Day American English

$$-2\ln(\lambda) = 2\sum_i O_i \ln\left(\frac{O_i}{E_i}\right) \tag{7.5}$$

In our case N_1

According to our frequencies for the word "foo" and we replace the observed frequencies O_i with our frequencies a and b (see Table 7.2) we can calculate the log-likelihood (LL) as follows:

$$\begin{aligned} G^2_{rayson\&garside} &= 2((a\ln(a/E_1)) + (b\ln(b/E_2))) = \\ &\quad 2((4500\ln(4500/4313)) + (40\ln(40/227))) = \\ &\quad 2(190.99 - 69.44) = \\ &\quad 243.11 \end{aligned}$$

There is also a log-likelihood calculator on the web. This calculator requires the frequencies for one word and the corpus sizes: `http://ucrel.lancs.ac.uk/llwizard.html`.

7.2 Comparing words using reference corpora with frequency adjustment

7.2.1 Characteristic Element Diagnostic - ced

The ced parameter used to filter the relevant terms from non-relevant terms is called the *characteristic element diagnostic (ced)* and identifies *characteristic elements* or *characteristic textual units* [69]. This measure gives information whether a word is over- or under-represented in a sub-corpus in comparison to the rest of the corpus. For the calculation of this diagnostic value, it is assumed that the words are distributed in a hyper-geometric manner. This is not true, because the words typed by a human being are far more complexly distributed. Therefore, the calculated values are no probabilities, but they serve as a powerful and descriptive property.

The ced-value is computed based on four quantities, expressed as simple word counts:

- the corpus size N,
- the size of the sub-corpus n,
- the word frequency in the whole corpus M and
- the observed word frequency k in the sub-corpus.

According to the urn problem without replacement, the entire population N has some marked objects M and we draw a sample which has exactly n objects. Figure 7.1 shows an example of a hyper-geometric distribution for a model corpus. In this example we are looking for a fictitious word "foobar", which has a total word frequency in the overall corpus of 36. The distribution in Figure 7.1 is our discrete probability density function. It can be seen, that the probability of drawing zero marked objects is approximately 1.4%. The mode of the distribution is the most probable value and has a probability of approximately 20.7%.

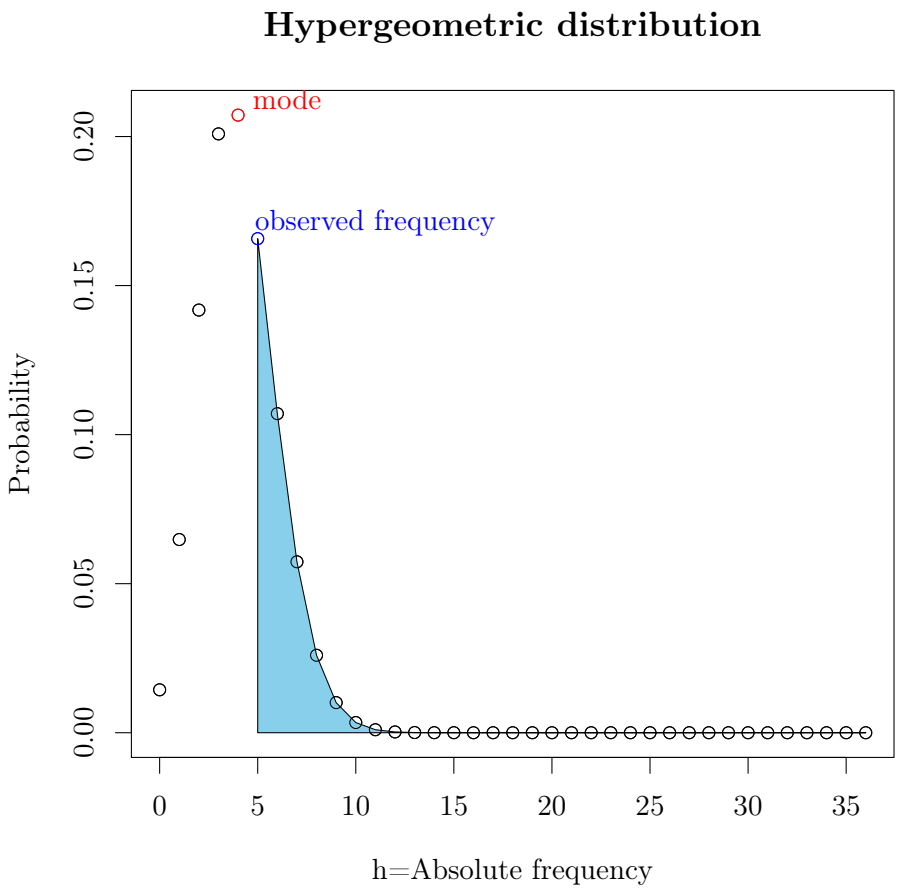


Figure 7.1: Calculation of the ced-value: Hyper-geometric distribution *(corpus size (N): 160000, size of sub-corpus (n): 20000, word frequency in the whole corpus (M): 36).*

Now we place our observed frequency of the sub-corpus in relation to the mode of the distribution. If "foobar" is observed much more often as the mode, then "foobar" is overrepresented in the corpus. To express this in a value, we could calculate the "probability" of having a number of marked objects greater than or equal to k:

$$\mathrm{P}_{\mathrm{inf}}(k) = \sum_{i=k}^{n} \frac{\binom{M}{i}\binom{N-M}{n-i}}{\binom{N}{n}} \tag{7.6}$$

The same holds for under-represented objects, e.g. if "foobar" is observed much less often than the mode. So we calculate the "probability" of having a number of marked objects less than or equal to k:

$$\mathrm{P}_{\sup}(k) = \sum_{i=0}^{k} \frac{\binom{M}{i}\binom{N-M}{n-i}}{\binom{N}{n}} \tag{7.7}$$

The text analyst decides whether an object is over- or under-represented, so he sets up a threshold and if P_{inf} is smaller than the threshold, then the word is poorly represented in the sub-corpus and we have a *negative characteristic* noted as $CED-$.

If P_{sup} is smaller than the threshold, we conclude that the word is abundant in the sub-corpus and we have a *positive characteristic* noted as $CED+$. It could also happen, that neither P_{inf} nor P_{sup} is below this threshold, so the word is *banal* to the corpus.

7.3 Notes on corpus comparison

The question of "Which words are characteristic of a text?" is the dogma of Information Retrieval (IR), and the Term Frequency-Inverse Document Frequency (tf-idf) is a numerical statistics in IR that is intended to reflect how important a word is to a document collection or a corpus. The tf-idf (see Section 6.2.1 and [104] for detailed information and references) adresses the general IR-problem to retrieve and rank relevant documents to a user query from a large document collection or database, but it lacks in providing information to corpus comparison for the following two reasons [58]. First, tf-idf does not normalize for the document length, this is why IR-applications usually normalise for document length (see [114]). But after all, this normalisation is not satisfactory, because a single use of a word within a hundred words corpus is far less noteworthy than ten uses within a thousand word corpus. Secondly, in the case where words are present in all documents, the idf value collapses to zero (see Equation 7.8, where N is the total number of documents in the corpus and n_i are the documents containing the term i). Idf was originally introduced as "term specificity" [52].

$$idf_i = \log \frac{N}{n_i} \tag{7.8}$$

IR applications use stop word lists, which remove common words, and ignore this issue. After all, these two issues show the incompatibility of the tf-idf value for corpus comparison.

Another approach to corpus comparison is the Mutual Information (MI) statistic [**?**].

$$MI_{\omega,X} = \log_2 \left(\frac{a}{c} \cdot \frac{N}{a+b} \right) \tag{7.9}$$

This measure states how much information word ω provides about corpus X (see Equation 7.9 and contingency Table 7.2). MI is not suitable for corpus comparison, because it is known to overemphasize common terms [58].

In comparing corpora, an important question is: "How large has to be the reference corpus

to have meaningful results?". A study ([11]) on the basis of WordSmith Tools KeyWords[6] procedure compares a study corpus with reference corpora of various sizes. The sizes of the reference corpora vary from two to 100 times larger in contrast to the study corpus. The study yields that the reference corpus needs be at least five times larger than the corpus [11] in order to have meaningful statistics.

[6]the Keyness value in WordSmith tools is calculated on basis of a contingency table with Ted Dunning's Log Likelihood Test (see G^2 in Section 7.1.2)

Part III

Implementation

Chapter 8

Requirements on LeMATo

In Chapter 3 we have described what lexicometrics is and how it differs from traditional corpus linguistics. We also introduced the theoretical background of lexicometrics and how it has evolved in the French discourse analysis (see Chapter 4). Here in this chapter, we recap the basic concepts which are published in [33] to formulate requirements for a tool, which supports discourse analysis of text corpora in social sciences.

Corpus compilation: There are a few studies (see references in Chapter 3) which have already applied lexicometrics to text. Most of these studies analyse newspaper articles for significant terms, co-occurrences and frequencies. In our reference study [78] the acquisition of such newspaper articles has been done with the LexisNexis [13] interface. LexisNexis provides the possibility to obtain newspaper articles with certain keywords within a period. It would be the best option to use a REST interface on LexisNexis to query the newspaper articles of our interest within LexicoMetric Analysis TOol (LeMATo). Unfortunately, the University of Vienna does not have access to the REST-API, so we decided to parse plain text files, which are downloaded from the LexisNexis web front-end.

The linguistic preprocessing (outlined in Section 3.1.3) for texts contains tokenisation, stop-word filtering and either lemmatization or stemming. Lemmatization is highly preferable over stemming, but it comes with computational costs, and it is also not easy to incorporate lemmatization within our search-engine (i.e. Elasticsearch), especially for the German language. There are some attempts[1], but at the time of writing, they are not compatible with current Elasticsearch implementations. After all, we decided to continue with stemming instead of lemmatization.

The frequency analysis (see Section 10.3) focuses on particular issues such as N-grams, diachronic corpora and Kendall's τ. N-grams are implemented in various programs (e.g. AntConc [3]), and since LeMATo is a prototype, we decided to leave out N-grams in the initial version. However, a diachronic view on text corpora is not very common in text analysis programs, and it might be very useful to see how the terms evolve in a particular period. We provide an annual view on the development of the frequencies. We also include

[1] see `https://github.com/larsmans/lucene-stanford-lemmatizer`

Kendall's τ, to have a view on the development of a term over the years. It would be beneficial if we could query certain parts of the corpus.

The Concordance analysis (see Section 10.4) performs a Key-Word In Context (KWIC) analysis, which enables the social scientist to query the context of certain keywords, which is something quite important for exploring a corpus and its keywords. Therefore, we include it in our tool.

Significance analysis: Here, in this paragraph, we sum up both kinds of analysis, namely the *analysis of characteristics in sub-corpora* (see Section 3.4) and the *co-occurrence analysis* (see Section 3.5) to a **significance analysis**, since both analyses are very similar in the presentation of their results and their significance calculation. As we have already mentioned, there are some restrictions in the initial version of LeMATo. Here we decided to make a limitation on the calculation of the significance of the terms. Since our reference study [78] is split into three distinct parts, we calculate the significant terms of each part in contrast to the rest of the corpus. The approved methodology here would be to calculate the significant terms in contrast to a much bigger corpus (e.g. a reference corpus, see also Chapter 7).

Furthermore, in the calculation of significance, we are not only interested in document frequencies. Sometimes a much smaller text unit (e.g. paragraph or sentence) is more interesting. Therefore, we decided to split each newspaper article into paragraphs and sentences. This enables us to make a significance calculation on sentences, which is important if we are looking at co-occurrences. Sometimes it is not enough, to look only at the sentences, because the sentence context provides not enough meaning, therefore we include also the calculation of significance on paragraphs into LeMATo.

Since we cluster the most significant terms to groups, we need to visualise the grouping in a dendrogram (see also Section 5.3).

The *analysis of characteristics in sub-corpora* needs the possibility to distinguish between different parts of the corpus, so we provide the opportunity to annotate various parts of the corpus with tags.

Visualisation: There are several visualisation techniques for frequency diagrams, concordance plots, dendrograms and tables. Google Charts[2] offers an excellent collection of visualisation techniques, including all kinds of charts, and more important for us, a double word tree (see Section 10.4). The visualisation of the dendrogram, which is the outcome of our clustering, could be performed with d3js[3]. Finally, we conclude, that the visualisation of the results in LeMATo could be best achieved with web techniques; therefore, we have a necessity to build a web application.

Summary Finally, we are summarising the requirements on LeMATo and, additionally, we want to emphasize here, that we are creating an application, which is modular and easily extensible.

[2] `seehttps://developers.google.com/chart/`
[3] see `http://bl.ocks.org/mbostock/4063570`

- Parsing LexisNexis text files into document, paragraph and sentence entities.
- Linguistic preprocessing includes stemming instead of lemmatization.
- Diachronic view on the corpus including raw frequencies, relative frequencies and Kendall's τ on the most frequent terms.
- KWIC visualisation with a double word tree.
- Calculation of significant terms for all levels of text units (sentence, paragraph and document).
- Providing a dendrogram for the clustering results.
- Annotation of tags for different sub-corpora.
- The necessity to build a Web application to provide best possible visualisation techniques.

"Nice-to-have" features: Finally, we end up with some features which are not necessary, but they offer a quite handy feature for discourse analysts. If we are facing raw frequencies it is sometimes not very clear, how a term is used within the context. Therefore, we provide a possibility to look at the utilization of the word within its document, paragraph or sentence.

Since we have annotated our text with tags, publish date of each newspaper article, and the text itself, it would be great to have a query system, which allows us to query individual texts within a period or with certain tags or even with a containing term. Such a query system is already provided within Apache Lucene [4].

- Allowing to zoom into the document, where the term was used.
- Applying Apache Lucene queries

Use Case: Finally we conclude this chapter with a use case diagram (see Figure 8.1). A LeMATo user could perform four different kinds of analyses, but each analysis requires a corpus, which needs to be defined in advance.

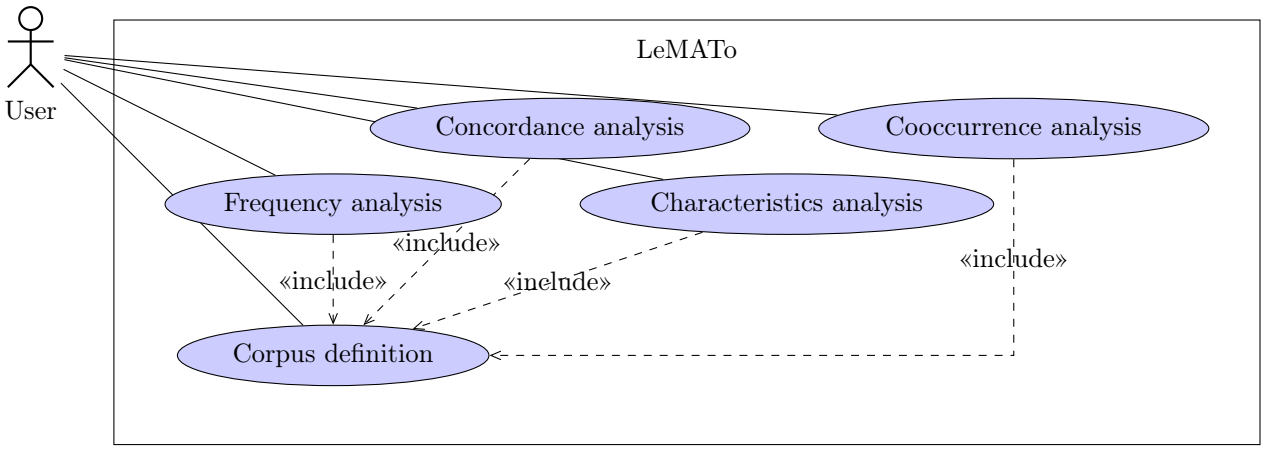


Figure 8.1: Use case diagram for LeMATo. Each analysis requires a corpus definition.

Chapter 9

Software Stack

9.1 Groovy and Grails

LexicoMetric Analysis TOol (LeMATo) is built with the open source web application framework Grails [96]. It is based on the Convention-over-Configuration design paradigm [80], which means a developer only specifies unconventional aspects of the application to reduce the complexity of configuration files. For example, the name of the class corresponds to the name of the table in the database. It behaves as expected without additional configuration files, but could get adapted depending on the needs.

Grails as a dynamic framework embraces the Don't Repeat Yourself (DRY) principle. Grails reduces the complexity of building web applications for the Java platform. In contrast to other dynamic frameworks like Rails or Django it is built on top of already established Java web technologies like Spring [90] and Hibernate [94].

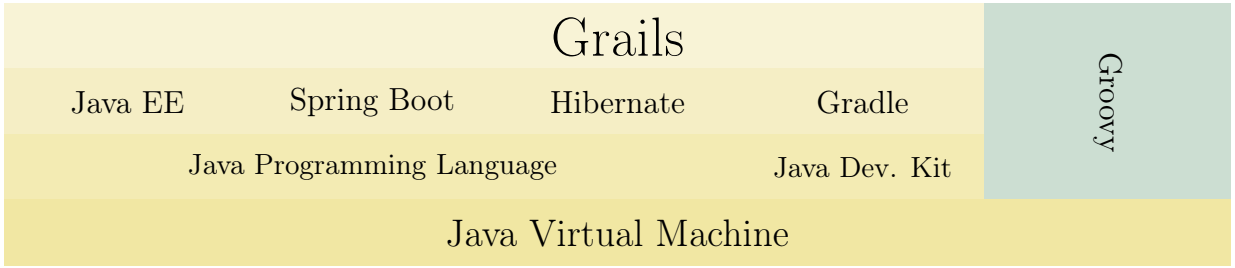


Figure 9.1: Depicting Grails architecture.

Grails is a full stack framework, this means that it supports the full development stack from the user interface till the data store. Grails core technology and its associated plug-ins serve some comfortable "out of the box" features like:

- An easy to use Object Relational Mapping (ORM) layer built on top of Hibernate [94]
- An expressive html view technology called Groovy Server Pages (GSP) (similar like JSP or ASP but far more flexible and intuitive)
- A controller layer built on Spring MVC [90]
- Iterative command line environment and build system based on Gradle [28]
- embedded Tomcat [5] container for on the fly reloading
- Dependency injection with the inbuilt Spring [90] container
- Support for internationalization (i18n) built on Spring's [90] core MessageSource concept
- Transactional service layer built on Spring's [90] transaction abstraction

Grails is intended to be a high-productivity framework, which means that there is no Extensible Markup Language (XML) configuration, a ready to use development environment, and functionality is available through mix-ins[1]. Instead of XML, all of the features listed above, are easy to use through the power of the Groovy language [66] and the extensible use of Domain Specific Language (DSL) [97]. For a detailed introduction into Grails see reference documentation [97].

9.2 Elasticsearch

Elasticsearch [36] is made to query text and to perform statistical analysis on a corpus of text. More specifically, Elasticsearch is a standalone database server, written in Java. Due to its multi-tenant architecture, Elasticsearch provides a distributed and scalable environment [44].

The core of the intelligent search engine in Elasticsearch is Apache Lucene [4]. Lucene implements everything that pertains to the algorithms of matching text and storing optimized indexes of searchable terms. However, Lucene is not easy to use directly in a Java application, so Elasticsearch hides the complexity of Lucene behind a simple and coherent RESTful interface.

Elasticsearch as a full-text search engine has a RESTful web interface with schema-free JSON documents. This means that Elasticsearch is accessible from any programming language over port 9200 with any web client. It is even available from the command line using the `curl` command.

LeMATo uses the "node client" from the Groovy API [34] to access the Elasticsearch server. The Groovy API is a wrapper on top of the Elasticsearch Java API. The Java API comes with two different built-in clients, the "node client" and the "transport client":

[1] A mix-in is a method that is added to a class dynamically, as if the functionality had been compiled into the program.

The node client joins a local cluster without holding any data by itself, but it knows what data lives on which node in the cluster and can forward the request directly to the correct node.

The transport client does not join the cluster itself, but forwards requests to a node in the cluster.

Using Elasticsearch from a client has certain advantages over the RESTful API. All operations are completely asynchronous, and operations on a client may be accumulated and executed as a `bulk`, which takes advantage of streams and multi-threaded environments [44]. For more details on joining Elasticsearch with a node client see Section 10.2.2.

Elasticsearch is distributed by nature and it is designed to hide the complexity that arises with being distributed, this includes operations like:

- Partitioning documents across multiple nodes
- Balancing the data load in the cluster
- Managing redundancy to prevent data loss in case of hardware failure
- Routing requests to the data of interest
- Integration of new nodes and redistributing in the event of data loss

All these operations are happening automatically under the hood and are abstracted by Elasticsearch itself. LeMATo is built on top of Elasticsearch, and this means that LeMATo scales horizontally as well as Elasticsearch does, but LeMATo has not been tested on a distributed system until now.

The essential unit, where the search actually happens is a Lucene index. This index is made up of one or more immutable index segments. Any CRUD-operation here, adds a segment to the index. Some segments are merged to reduce the runtime complexity (for detailed information why and how segments are merged we refer to the documentation[2]).

An index in Elasticsearch is a collection of shards. Each shard is a Lucene index. When Elasticsearch searches an index, it sends the query out to a copy of every shard to see if they have any matching documents. The results from multiple shards must be combined into a single sorted list before the hits are returned. This two-phase process is called "query then fetch" to perform any search queries in Elasticsearch.

[2] `https://www.elastic.co/guide/en/elasticsearch/guide/current/merge-process.html` [44]

9.3 The S-Space Package

The S-Space Package is an open source framework for developing and evaluating word space algorithms. The package comes with well-known word-space algorithms in various application domains:

Document-Based Models
Latent Semantic Analysis (LSA) [68]
Explicit Semantic Analysis (ESA) [42]
Vector Space Model (VSM) [106]

Co-occurrence Models
Hyperspace Analogue to Language (HAL) [72]
Correlated Occurrence Analogue to Lexical Semantic (COALS) [98]

Approximation Models
Random Indexing (RI) [102]
Reflective Random Indexing [20]
Temporal Random Indexing (TRI) [54]
Bound Encoding of the Aggregate Language Environmet (BEAGLE) [53]
Incremenatal Semantic Analysis (ISA) [9]

Word Sense Induction Models
Purandare and Pedersen [91]
HERMIT [55]

The implementations listed here, come with some tuning mechanisms for different screws. As an example, the implementation of HAL[3] offers two distinct options for dimensionality reduction based on the entropy measure. One allows to filter on an entropy threshold, and the other retains n columns with the maximum entropy. Applying the calculation of principal components may further reduce the high dimensionality, but comes only with the expensive cost of computational complexity [72].

The S-Space Package is not only a software stack for the word-space algorithm, it also provides a lot of utilities to develop a customized semantic word space algorithm. We are using a clustering algorithm (i.e. hierarchical agglomerative clustering) from the S-Space Package to develop an algorithm, which allows clustering the words into thematic groups according to the co-occurrence of these words. Instead of providing a tight solution for the clustering of words into thematic clusters, we provide a visualisation aid, which is a dendrogram of the words which co-occur frequently. In the following, we give an overview of the hierarchical agglomerative clustering algorithm, which is implemented in the S-Space Package and used in LeMATo.

Hierarchical Agglomerative Clustering (see Section 5.3.2 for a more detailed explanation) is a clustering algorithm, which builds a hierarchy of clusters in a "bottom up" approach: each observation is its own cluster in the beginning and pairs or merges with the neighbour cluster under certain similarity conditions. In most hierarchical clustering algorithms the similarity is achieved by choosing an appropriate distance metric (see Section 6.4.1) whereas a linkage criterion (see Section 6.5) specifies the dissimilarity between two clusters.

[3] `https://github.com/fozziethebeat/S-Space/wiki/HyperspaceAnalogueToLanguage`

The current implementation in the S-Space Package runs in $\mathcal{O}(n^3)$ time. The method of $\mathcal{O}(n^2 \log n)$ is not implemented in the S-Space, due to the fact that it requires a priority queue of all similarity comparisons, which results in a much higher run time cost for large matrices. For some special cases, optimal efficient ($\mathcal{O}(n^2)$) algorithms are known (e.g. SLINK [113] for single linkage, CLINK [24] for complete linkage), but these are not implemented in the S-Space Package. The implementation offers four different linkage criteria: *a*) complete linkage, *b*) mean linkage, *c*) median linkage and *d*) single linkage (see Section 6.5).

License information

- *Purpose:* Performing Hierarchical Agglomerative Clustering.
- *Author:* David Jurgens and Keith Stevens
- *Url:* `https://github.com/fozziethebeat/S-Space/`.
- *License:* GNU General Public License 2.

9.4 Other software dependencies

In the previous sections we have introduced Grails, Elasticsearch and the S-Space Package. These three frameworks comprise the core of LeMATo, but there are various other dependencies used within LeMATo. Here, we list all the software dependencies with their license and use in LeMATo.

9.4.1 Querying and using Elasticsearch

This API is built on top of the Elasticsearch Java API [35]. We use the Groovy Client API [34] for almost all operations with Elasticsearch. Sometimes we had the problem that the documentation is not sufficient, especially for initializing index templates and indexed scripts. The problem in that case was solved by using the REST interface of Elasticsearch with Httpbuilder [45].

Elasticsearch Groovy Client

- *Purpose:* Querying and indexing Elasticsearch.
- *Author:* Elastic.
- *Url:* `https://github.com/elastic/elasticsearch-groovy`.
- *License:* Apache License 2.0.

Httpbuilder Groovy

- *Purpose:* Initialization of index templates and indexed scripts.
- *Author:* Jason Gritman.
- *Url:* `https://github.com/jgritman/httpbuilder`.
- *License:* Apache License 2.0.

Calculations

- *Purpose:* Calculation of Kendall's τ.
- *Author:* Apache Software Foundation.
- *Url:* `https://commons.apache.org/proper/commons-math/`.
- *License:* Apache License 2.0.

9.4.2 Layout and appearance

Here we give an overview of the software and we have used for designing the layout and the appearance of LeMATo.

Twitter Bootstrap

- *Purpose:* Basic layout.
- *Author:* Twitter Inc.
- *Url:* `http://getbootstrap.com/`.
- *License:* MIT license 2015.

Bootstrap-table

- *Purpose:* Table layout.
- *Author:* `https://github.com/wenzhixin`.
- *Url:* `http://bootstrap-table.wenzhixin.net.cn/`.
- *License:* MIT license 2015.

Bootstrap-tagsinput

- *Purpose:* Tags input.
- *Author:* Tim Schlechter.
- *Url:* `https://github.com/bootstrap-tagsinput/bootstrap-tagsinput`.
- *License:* MIT license 2015.

Typeahead.js

- *Purpose:* Tags input.
- *Author:* Twitter Inc.
- *Url:* `https://github.com/twitter/typeahead.js/`.
- *License:* MIT license 2015.

D3js.org

- *Purpose:* Visualisation of the dendrogram.
- *Author:* Unknown author.
- *Url:* `http://bl.ocks.org/mbostock/4339083`.
- *License:* BSD 3.

Chapter 10

Design of LeMATo

10.1 Grails MVC(S)

LexicoMetric Analysis TOol (LeMATo) is built with the web application framework Grails [96]. Grails architecture enforces like any other web development framework the Model-View-Controller (MVC) pattern [61, 70] to have a clean modularisation of each layer.

Moreover, grails discourages the embedding of core application logic inside controllers as it does not promote reuse and a clean Separation-Of-Concerns (SoC) [27]. Therefore, Grails recommends using service classes to put the majority of the logic in the application, leaving controllers responsible for handling request flow and redirects. This separation of the business logic into service classes is also known as Model-View-Controller-Service (MVCS) paradigm [12] (see Figure 10.1).

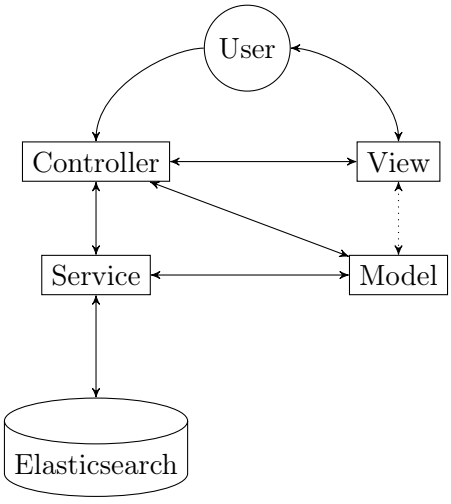


Figure 10.1: Grails MVCS paradigm as it is used in LeMATo

The sequence diagram in Figure 10.2 represents a generalisation of the data flow in the MVCS paradigm used in LeMATo. Each analysis in LeMATo is built very similar to this illustration in Figure 10.2. The user performs an action in the view and triggers with a Representational State Transfer (REST) call the action on the controller. We have simplified this in the

diagram with the "action" call to increase the legibility. Depending on the kind of action a service method is called and looks up relational data in the model if necessary. The service class also performs Elasticsearch queries, which return Elasticsearch objects ("ES object"). The controller delivers the data to the view. The view renders the data into a HyperText Markup Language (HTML) page, which gets displayed to the user.

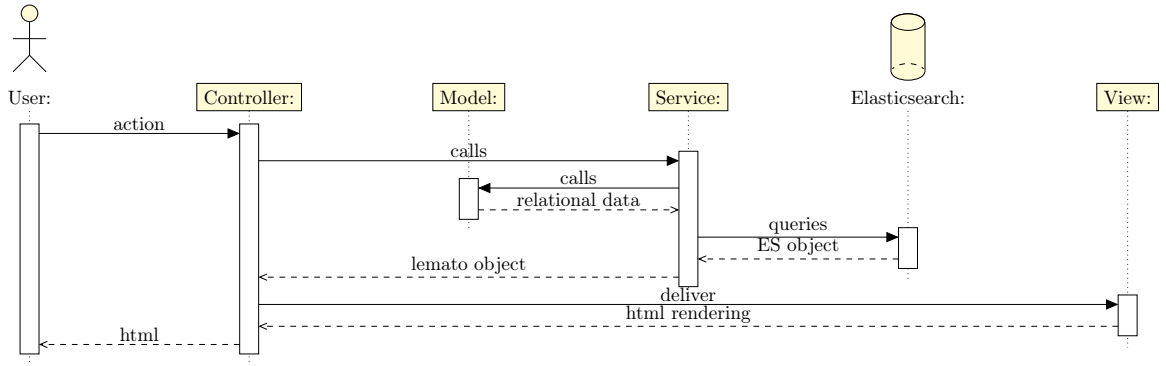


Figure 10.2: Sequence diagram of MVCS archtecture.

For each analysis we perform in LeMATo, we have created a separate controller, view, model and service class. The frequency analysis contains three different views on the Elasticsearch data. Therefore, we use three different objects[1] to display the results. To increase the performance and reduce the response time, we do not persist the results. The result of each query performed on a corpus in LeMATo gets lost and needs a re-run.

10.2 Elasticsearch

Elasticsearch is different from and not comparable to SQL databases. In contrast to SQL databases, there are a lot of benefits in performance, scale, real-time search and analytic functions across massive amounts of data. On the other hand, handling relationships is not that obvious as it is in relational databases and, therefore, the golden rule of thumb to normalise the data does not apply to Elasticsearch. In this section we discuss the three different kinds of relations in Elasticsearch and why they don't apply to our case. Finally, we present our choice of the model which takes advantage of relational databases and the benefits of Elasticsearch.

10.2.1 Relational Elasticsearch

Elasticsearch, like most NoSQL databases, models the world as a flat structure. An index is a flat collection of independent documents, which contains information that decides whether it gets matched to a search request or not. The independence allows a multi-tenant architecture which spreads massive amounts of data across multiple nodes. While changing a single document is ACID, transactions involving multiple documents are not. The lack of the roll-back enables other performance advantages like fast and lock-free indexing and searching.

[1] see `https://github.com/perdacherMartin/LeMATo/tree/master/grails-app/domain/lemato/nopersistence`

Four common techniques manage relations in Elasticsearch:

- Application-side-joins
- Data de-normalisation
- Nested objects
- Parent-child relationships

Most of the time the final solution includes a mixture of a few techniques [44]. Application-side-joins make sense in cases of a one-to-many relationship, where "one" has a small number of "many" with seldom change. In our case we have many paragraphs and even more sentences. The second possibility is data de-normalisation: you append the "one" into your "many" entity. This applies in cases, where speed really matters, but unfortunately this makes the calculation of significance on paragraph and sentence level difficult or even impossible. This statement also holds for nested objects and parent-child relationships, which add a hierarchy layer to the index structure of our data structure and makes querying very complex. With LeMATo we want to provide the full power of Elasticsearch to the user, therefore, we try to simplify our queries and keep a simplified data structure.

To fulfil the requirement of an easy-to-use query system and to query on three different levels, we need to break up the normal form and store the text in three Elasticsearch fields (i.e. document, paragraph and sentence). A field in Elasticsearch is comparable to a separate table in a relational database, so we are storing three-fold redundancy (see also the class diagram in Figure 10.3). The advantage of this approach is to easily calculate significance and apply filtering on the corpus with Apache Lucene queries[2].

Text fields are very expensive for relational databases, so we have left them out in the Hibernate layer by marking them as transient objects. The Hibernate layer keeps track of the relational data, that is the compositions between documents, paragraphs and sentences (see Figure 10.3).

10.2.2 Obtaining an Elasticsearch client

Before we obtain an Elasticsearch client, it is important to set up some prerequisites on the Elasticsearch environment to have a clean index structure with **index templates** and a stable scripting environment with **indexed scripts** to calculate term frequencies. Both settings are done with a HyperText Transfer Protocol (HTTP) query over the REST interface of Elasticsearch.

Indexed templates: In Elasticsearch there is usually no need for creating an index, everything happens with dynamic mappings. This feature is great for testing, but in practice, there is the need to define settings and mappings on the internal structure of the index. In our approach, we set index templates for our index structure. There are two main properties of the index: the settings and the mappings. In the settings section, we have defined filters and analysers, which depend highly on language settings[3]. A static closure contains these

[2] see `https://lucene.apache.org/core/2_9_4/queryparsersyntax.html`

[3] Currently we only provide the use of the German language, because our reference study [78] uses also a German corpus.

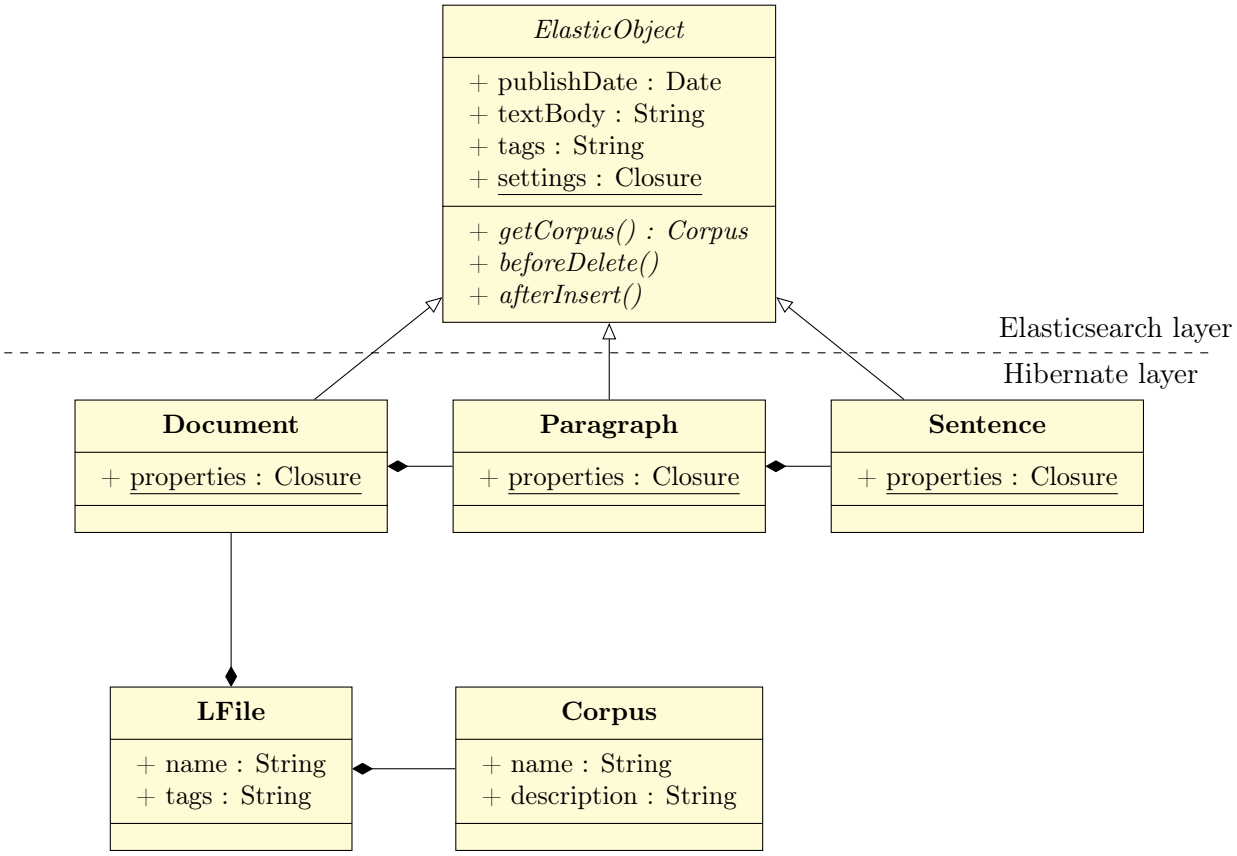


Figure 10.3: Class diagram.

settings[4] in the ElasticObject (see Figure 10.3). The mappings[5] for each Elasticsearch field (document, paragraph or sentence) are stored in separate entities as a static closure. The two abstract methods `beforeDelete()` and `afterInsert()` are used to trigger Elasticsearch on the Hibernate events "before delete" and "after insert". This methods allows us to delete Elasticsearch objects, whenever an Hibernate object gets deleted. This approach enables cascading deletes even for Elasticsearch objects. The same applies for inserts. We are loading the data into Elasticsearch after it has been inserted into the relational database.

Indexed scripts: Elasticsearch is focussed on documents and, therefore, most of the calculation of significant terms or even only counting objects is mostly related to document counts. To observe term frequencies, it is necessary to use the scripting environment[6] of Elasticsearch. There are three different approaches to the scripting environment: *a*) dynamic scripting, *b*) file scripting and *c*) indexed scripts. The dynamic scripting environment of Elasticsearch could be used in every Elasticsearch environment, which has the dynamic scripting

[4] see `https://github.com/perdacherMartin/LeMATo/blob/master/grails-app/domain/lemato/es/ElasticObject.groovy`

[5] see `https://github.com/perdacherMartin/LeMATo/blob/master/grails-app/domain/lemato/es/DocumentRdb.groovy`

[6] see `https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-advanced-scripting.html`

flag enabled. Dynamic scripting is turned off by default from version 1.4.3 to prevent scripts from being accepted as a part of a request to decrease security issues. File scripting allows the execution of scripts which are stored in the `/config/scripts` directory. Unfortunately, scripts cannot be uploaded in cloud environments (i.e. Elastic found[7]), which are free of charge. Premium accounts are permitted to upload their scripts within custom bundles. After all, we applied indexed scripts in LeMATo to observe our term frequencies. Elasticsearch stores the scripts in an internal index `.scripts` and references it by the file name which is also an `id` for the script.

Factory method: The creation of the Elasticsearch client object requires checking the prerequisites of index templates and indexed scripts. There is also additional information, which is required to connect to the Elasticsearch environment. This information includes cluster information, like the name of the cluster, the Uniform Resource Locator (URL) and the port. In LeMATo this information is set through a configuration file[8] in Grails conf directory (i.e. `grails-app/conf`).

Obtaining an Elasticsearch client is, therefore, a complex process, which could be managed best by a factory method [43] pattern, which is a "creational pattern". Since Grails is based on Spring [90], we are using the Spring implementation here to create our factory method (see Figure 10.4). The `afterPropertiesSet()` method initializes the client based on the configuration and calls `createIndexTemplate()` and `createIndexScript()` to set up the index template and the indexed script, which are prerequisites in LeMATo.

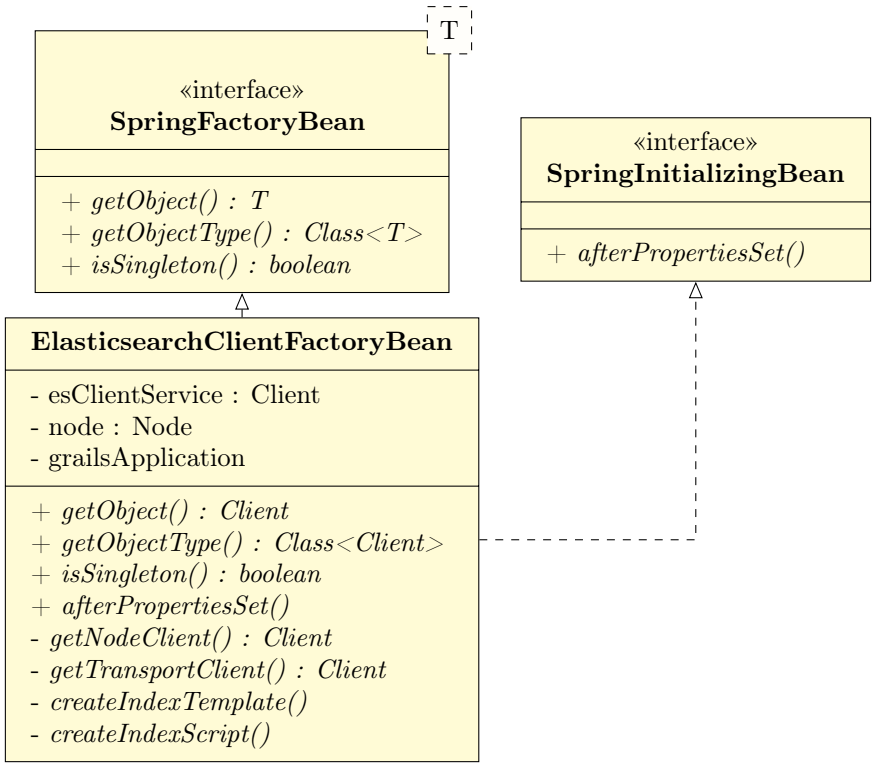


Figure 10.4: Obtaining an Elasticsearch client using Spring factory.

[7] see `https://www.elastic.co/found`
[8] see `https://github.com/perdacherMartin/LeMATo/blob/master/grails-app/conf/application.yml`

10.3 Frequency analysis

The frequency analysis starts with a parameter selection (see "Corpus selection" in Figure 10.5), where the user selects the corpus and could define an additional query (i.e. Lucene query) to filter the corpus on particular parts (e.g. time period, tag or documents with a particular term in it). Furthermore, the user needs to specify a limit of the terms (10, 20, 50, 100, 500). Unfortunately, we cannot display all terms, because the calculation of Kendall's τ needs an additional query for each term we have selected.

In the frequency analysis (see "Show frequencies" in Figure 10.5), we provide diachronic term-, document-, paragraph- and sentence frequency distributions (see Figure B.6) for the currently filtered part of the corpus. In a table, we give an initial overview of the most frequent words with some statistics like document frequencies, absolute and relative term frequencies, and Kendall's τ, which gives an idea of the development of the term frequencies over the years (see Figure B.5 in the user's guide). The user could select interesting words in the table and reveal its diachronic frequencies by clicking on the "Show detail for selected" (see Figure B.6 in user's guide). Furthermore, it is possible to show the documents of a certain year for a specified term by clicking on the bar (e.g. on document frequency distribution for a term) in the detailed view. The web flow of the frequency analysis is visualised in Figure 10.5. It is even possible to look into the documents, where the specified keyword is highlighted.

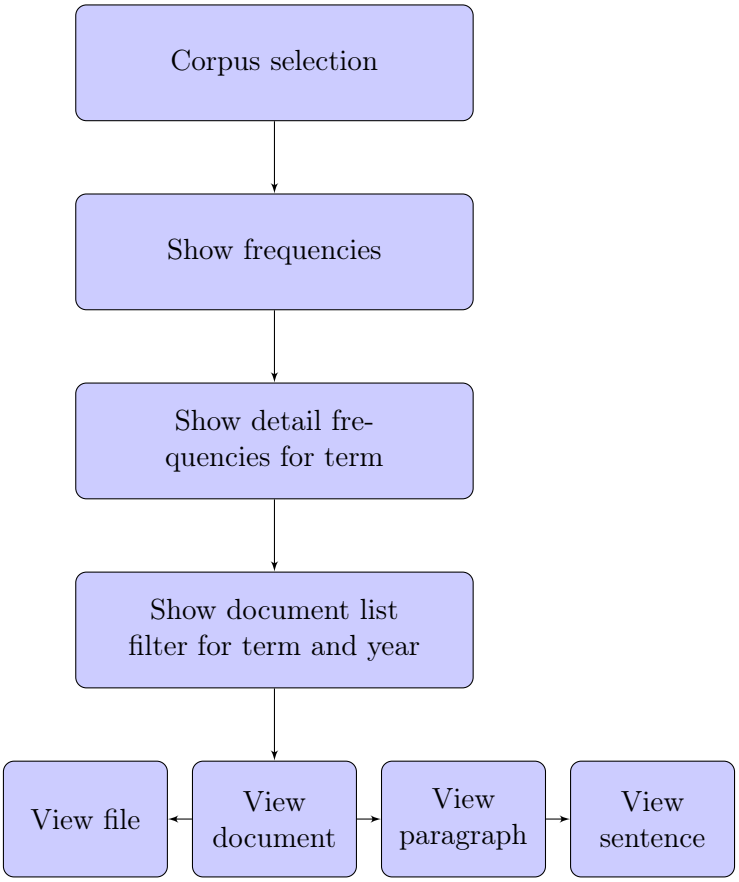


Figure 10.5: Web flow for the frequency analysis.

10.4 Concordance analysis

The concordance analysis starts with the selection of the corpus, a centered word and an optional query to filter the corpus on a particular part (see Figure B.8 in the user's guide). For the visualisation, we are using the double word tree (see Figure B.9 in the user's guide). The concordance analysis is straight forward, and we are showing the context of our centered word with the tag and the publish date (see Figure B.10 in the user's guide).

10.5 Significance analysis

There are two kinds of analyses which measure significant terms in parts of the corpus: the "Analysis of characteristics in sub-corpora" and the "Co-occurrence analysis". In the first analysis, we investigate on text units which are labelled with tags, and LeMATo finds the most significant words for the entered tags. The background set here is the superset of text units in the corpus, which has a different tag than the current tag.

In the second analysis, we investigate on text units which contain a keyword. Here, LeMATo looks for the most significant words, where the background is again the superset of text units in the corpus which do not have this word in their text.

In both analyses, we have the same features to display: a list of words, with their significance measure and their frequency counts. On top of that, we are trying to find some commonalities of words to form a common group. We have decided to merge both analyses in one parametrized analysis. We call this analysis "Significance analysis". From the user interface, it still looks like, as if these two analyses are two different kinds of tools.

10.5.1 Significance measure

To calculate significant terms we are using the chi-square test [75]. This statistic is implemented in the "score" of significant terms. The calculation of the score is based on the count of text units, not on raw term frequency counts. For LeMATo, this means that the selection of "document" counts every document in the subset which fulfils our criteria (e.g. contains the word or labelled with tag), and compares it using the chi-square measure against the same criteria in the superset (see Figure 10.6).

The measurement of close proximity from words to other words, based on this kind of significance measure is in comparison to other approaches (e.g. sliding windows [72]) much faster, but slightly inaccurate, because there is no normalisation for the sentence length.

10.5.2 Grouping words according to their similarity

We have summarised the four steps of our theme finding process in Figure 10.7. After filtering for the most significant terms, we use the co-occurrence of the terms to built a co-occurrence matrix. There are several measures which describe co-occurrence, we have decided to use the Dice coefficient (see Section 6.1).

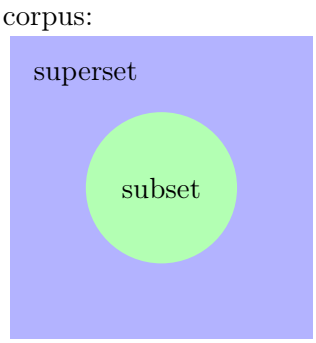


Figure 10.6: For the calculation of significant terms in the subset, we are using the superset of the corpus as a background reference.

We are using the Dice coefficient because other co-occurrence measures are not directly comparable to each other. For example, a higher log-likelihood ratio does not necessarily mean a more significant co-occurrence. If the single occurrence frequency of two words is high, then they also tend to co-occur, but they are less important than the co-occurrence of two words which are less frequent.

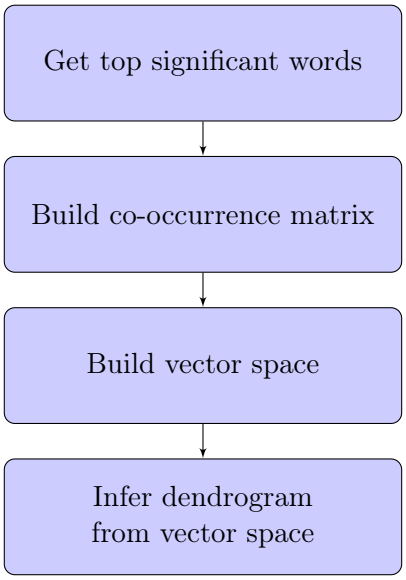


Figure 10.7: Steps to perform the significance analysis.

To measure the co-occurrence of N significant terms, we need to build an $N \times N$ matrix and calculate the co-occurrence for each entry of the matrix. Each co-occurrence of two terms is indeed a query in Elasticsearch. To reduce the complexity, we are using a symmetric matrix. The single occurrences N_A and N_B of all terms are queried with one Elasticsearch query, but the co-occurrence N_{AB} is an additional query. Therefore, if we calculate the co-occurrence matrix for N significant terms, we are using $(N^2)/2 - N$ queries.

We are using the co-occurrence matrix to build the vector space with the S-Space Package. We perform a hierarchical agglomerative clustering in the vector space. To decide which clusters should be combined, a measure of dissimilarity between sets of observations is required. This is achieved by the use of a **Euclidean metric**, and as a linkage criterion which specifies

Operating System	OS X Yosemite (Version 10.10.5)
MacBook Pro	13-inch Early 2011
Processor	2.7 GHz Intel Dual Core i7 (Sandy Bridge)
Memory	8 GB 1333 MHz DDR3
Graphics card	Intel HD Graphics 3000 512 MB

Table 10.1: Hardware used for testing the latency.

the dissimilarity of sets we are using the **mean linkage** criterion, which is the most common linkage criterion. The metric and the measure of dissimilarity could be changed in the configuration file (`grails-app/conf/application.yml`[9]).

For the visualisation of the results in the significance analysis, we are using a table, a scatter chart and a dendrogram. The table lists all significant words with a score (i.e. χ^2-score) with counts of the subset and superset (see Figure B.11 in the user's guide). The scatter chart plots the document frequencies against the χ^2-score (see Figure B.12 in user's guide). The dendrogram supports the researcher in the manual grouping of words (see Figure B.13 in the user's guide).

10.6 Latency

In this section, we give some insights to the response time of our web application. We are focussed on a clean application design with minimal response. We are using "minified" stylesheets and "minified" JavaScripts, but the real bottleneck in LeMATo is querying text with Elasticsearch. Elasticsearch is a "near-real-time" search engine for search. Other operations like create, update, delete or get are performed in real-time [44]. The term "real time", could be understood as the response in the order of milliseconds and sometimes microseconds, this includes receiving the data, processing it, returning and visualize the results at that given time [77]. In this section we are using the term "instantaneously" for some operations in LeMATo, by that we mean reacting in the order of at most 0.5 seconds with no special calculation required, except to display the result.

There are only three calculations in LeMATo, where the user needs to wait more than 20 seconds. The frequency analysis, the "analysis of characteristics for the sub-corpora" and the "co-occurrence analysis".

For the frequency analysis we need to calculate Kendall's τ on the diachronic frequencies, this means we are querying a date histogram for each term in the frequency analysis. In other words, if we are using the size 500, we get the terms with the top 500 document frequencies, and LeMATo applies 500 date histogram queries to calculate the Kendall's τ on each date histogram (see Figure 10.9).

Internally the "analysis of characteristics for the sub-corpora" and the "co-occurrence analysis" perform very similarly. Here, again, the bottle-neck is the Elasticsearch query. Building the co-occurrence matrix, which is necessary for a "co-occurrence analysis" or a single tag in "the analysis of characteristics for the sub-corpora" leads to $(N^2)/2 - N$ queries for Elasticsearch.

[9] see `https://github.com/perdacherMartin/LeMATo/blob/master/grails-app/conf/application.yml`

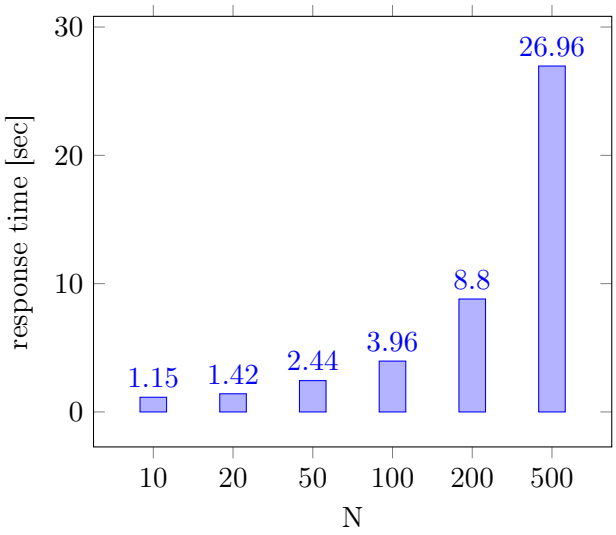


Figure 10.8: Latency for the frequency analysis. See Table 10.1 for details to our testing environment.

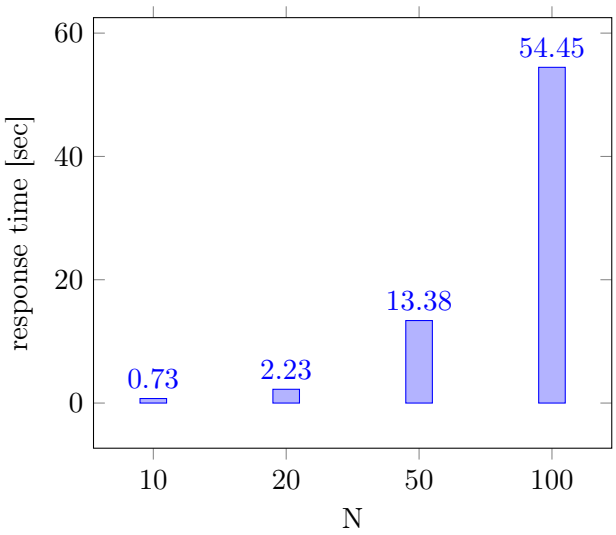


Figure 10.9: Latency for the significance pipeline. See Table 10.1 for details to our testing environment.

Part IV

Evaluation and conclusion

Chapter 11

Evaluation of results in LeMATo

In this chapter, we take a close look at the results we obtain from LexicoMetric Analysis TOol (LeMATo). In LeMATo we are performing four different kinds of analyses: *a*) *frequency analysis*, *b*) *concordance analysis*, *c*) *analysis of characteristics in sub-corpora* and *d*) the *co-occurrence analysis*. Since we have merged the last two analyses into one, namely into the *significance analysis*, we have organized this chapter into three main sections. Each section compares the results with AntConc [3] and discusses the comparison.

We are using the corpus of our reference study [78] (see also Section 4.2).

11.1 Frequency analysis

Before we compare the results obtained from the frequency analysis in LeMATo with the frequency results in AntConc, we have to mention their differences in preprocessing of terms. AntConc uses lemmatization (see also 3.1.3) to summarise different word forms into one term. LeMATo uses the stemming and stop-word filtering of Elasticsearch [36] instead of lemmatization. Because of that, we observe different frequencies for the same term. To make these frequencies comparable, we had to apply our own preprocessing on the text (see Figure 11.1).

We recall on the preprocessing in LeMATo (see also Section 3.1.3). LeMATo parses the text files which we have obtained from the LexisNexis [13] web interface. These text files contain several annotations, which get removed during the parsing process. To have the same piece of text in both programs, we query our corpus from the Elasticsearch database with a Groovy [66] script (see first step in Figure 11.1). With this corpus, we build a word list in AntConc with all words in its base form. We are analysing (i.e. preprocessing) each word in Elasticsearch to reduce the words to its stemmed form and write each word with its stemmed form in a text file. The analysed file from Elasticsearch with all stemmed words is used as a lemmatization file for AntConc, this means we are applying Elasticsearch stemming in AntConc. Next we are querying the top 500 terms from our Elasticsearch database in the same way as LeMATo does it. Finally, we compare the frequencies in Table 11.3.

At a first glance at Table 11.3, it seems that the frequency count in Elasticsearch, and therefore also the frequency count in LeMATo, is slightly reduced in comparison to the frequencies

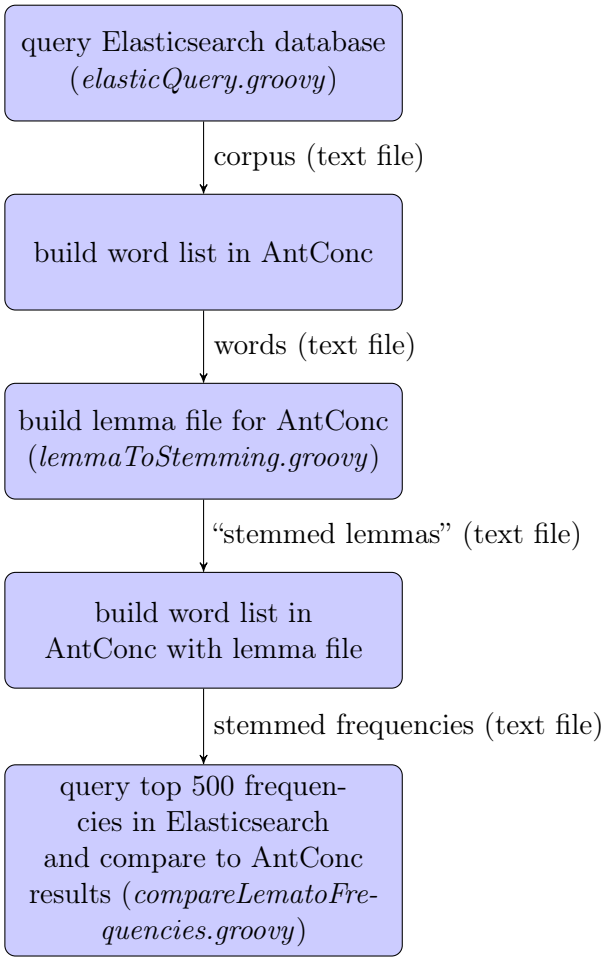


Figure 11.1: Comparing frequencies observed in LeMATo (built with stemming and stopwords) with frequencies observed in AntConc (built with lemmatisation).

in AntConc. Unfortunately, we don't have an explanation for that. Our guess is that it has something to do with the term aggregation[1] in Elasticsearch. The term aggregation in Elasticsearch is not always accurate. This missing accuracy is because each shard has its own view on the document structure and the term frequencies. We provided an example in Table 11.1. Here we have three shards, where each shard contains the word "foo" five times. A query on the top three terms and its frequencies commands each shard to perform a top three filtering on the terms on the shard. Therefore, on shard C we take only "fubar", "bar" and "foobar" without the term "foo". This means, that a request for the top three terms on this shard structure yields a total term frequency of 10 for the term "foo", which is obviously wrong. The term "foo" on Shard C will not get aggregated. However, this cannot be the source of the error because our shard structure contains only of one shard. We have set the number of shards to 1[2].

There is an apparent difference in the term frequencies of stop-words (see Table 11.3 for the

[1] see `https://www.elastic.co/guide/en/elasticsearch/reference/1.7/search-aggregations-bucket-terms-aggregation.html` in [44]

[2] see `https://github.com/perdacherMartin/LeMATo/blob/master/grails-app/domain/lemato/es/ElasticObject.groovy#L30`

Shard A	Shard B	Shard C
foo 5	foo 5	fubar 10
fubar 4	bar 4	bar 8
fu 3	fooboo 4	foobar 6
foobar 2	fu 2	foo 5

Table 11.1: Querying the top three terms in Elasticsearch yields a wrong term frequency of 10 for the term "foo". This is due to top aggregation on each shard.

frequencies in AntConc and LeMATo and Table 11.2 for a list of stop-words. Relevant stop-words are marked in red in both tables). The reason for this is that, due to the stop-words filtering process in Elasticsearch most of the stop-words get removed. Unfortunately, not all stop-words, because there are some word forms which become potential stop-words after the stemming process. An example is the stemmed word "muss", which appears 1 482 times in Elasticsearch and 3 037 times in AntConc. AntConc also lists the frequency of all other word forms (e.g. "muesse" 53, "muessen 126", ...). The word form "muss" which appears 1 555 times in the text is the exact difference between both observed frequencies for the stemmed version of the word. We also provide an overview of the Elasticsearch German stop-words in Table 11.2.

denn, daß, muss, allem, allen, dem, den, aller, alles, der, des, über, ihnen, andere, meinem, durch, manchem, manchen, anderm, andern, meines, am, an, anderr, anders, doch, welches, jene, denselben, wollen, meinen, wirst, keine, dasselbe, ein, hatte, sollte, seine, unter, mancher, mir, mit, so, während, anderem, anderen, meiner, dieselben, anderes, einen, einer, dazu, musste, jenem, jenen, da, derer, manches, jener, jenes, weil, desselben, wird, die, bei, hab, ist, sind, dir, deinem, deinen, du, zum, deiner, deines, zur, um, viel, hat, könnte, welche, unse, derselbe, er, es, das, gewesen, aber, auf, ich, habe, damit, mein, eines, aus, einigen, wieder, ander, indem, zwar, einiges, einmal, sie, diese, dann, vom, von, wo, vor, soll, sehr, eure, alle, werde, weiter, keinem, keinen, was, und, keiner, keines, würde, jedem, jeden, oder, jeder, sein, uns, jede, demselben, mich, haben, manche, bin, dessen, bis, wenn, sondern, solche, nicht, euch, jedes, ihrem, ihren, ihrer, ihres, unsem, unsen, im, in, unser, unses, ihm, ihn, wollte, ihr, wir, anderer, sich, dort, würden, derselben, welcher, meine, warst, ohne, für, nach, weg, man, eine, euer, solchen, machen, solches, dein, hier, wie, sonst, hinter, zwischen, nun, nur, hin, einem, einigem, waren, ihre, jetzt, einiger, kann, auch, war, dieselbe, werden, einig, hatten, dieser, als, selbst, dich, einige, können, gegen, seinem, deine, solchem, also, solcher, zu, will, ob, welchem, welchen, etwas, diesem, diesen, nichts, seinen, dieses, seiner, seines, noch, eurem, euren, ins, eurer, eures, dies, bist, kein

Table 11.2: Stop-words in Lucene 4.10.4. These stop-words are obtained through the following code snippet: `GermanAnalyzer.getDefaultStopSet()`. The stop-words marked in red have a apparent difference in their frequency (see Table 11.3).

Numbers are considered as terms in Elasticsearch, but not in AntConc. This is why we always observe a frequency count of zero for any number in Table 11.3.

For completeness, we also provide a table of the top 500 terms in the appendix section (see Table C.1). In this table, we have marked each difference in the frequency count. If the

Keyword	LeMATo	AntConc
stadt	6843	6849
tageszeitung	2174	2174
taz	2967	2969
koln	8007	8028
dass	6179	6179
jahr	4889	4890
mehr	3535	3535
wurd	3295	3295
neu	3301	3301
erst	2779	2779
schon	3144	3144
sei	2709	2713
seit	2432	2435
sagt	2848	2848
gross	2135	2135
gibt	1886	1925
deutsch	2781	2781
imm	2007	2007
ganz	1803	1803
lang	1772	1772
zwei	1511	1513
viel	1724	2897
wenig	1465	1465
muss	1482	3037
gut	1599	1600
berlin	2362	2366
weit	1134	1134
heut	1421	1421
soll	1187	2895
beim	1162	1162
euro	2017	2019
rund	1226	1226
frankfurt	1946	1948
bereit	1088	1088
ab	1130	1130
mal	1493	1495
konnt	1154	1154
zeit	1210	1210
geht	1170	1195
dabei	1084	1084

Table 11.3: Top fourty term frequencies of our reference corpus with the frequencies observed in LeMATo and AntConc. Stop-words are marked in red.

frequencies in LeMATo are higher than the frequencies in AntConc, we have marked them blue. We marked them red, if the frequencies of Antconc are higher than the frequencies of LeMATo. Beyond that, there are a few terms (e.g. “ford”, “buch”, “sech”, “aug”) which occur more often in LeMATo than in AntConc. The reason for this is, that some of the word forms got lost through the stemming to lemmatization transformation.

11.2 Significance analysis

Here in this section, we want to verify the significance results in LeMATo. Therefore, we compare the results achieved from LeMATo with the results obtained from the corpus comparison in AntConc. For the calculation of the significance, we are using the same corpus in both programs, which is also used in [78]. We compare each sub-corpus to the rest of the corpus. Each of our three sub-corpora is labelled with a tag, which refers to a city name. The names of the corpora and their sizes in term frequencies are depicted in a pie chart in Figure 11.2.

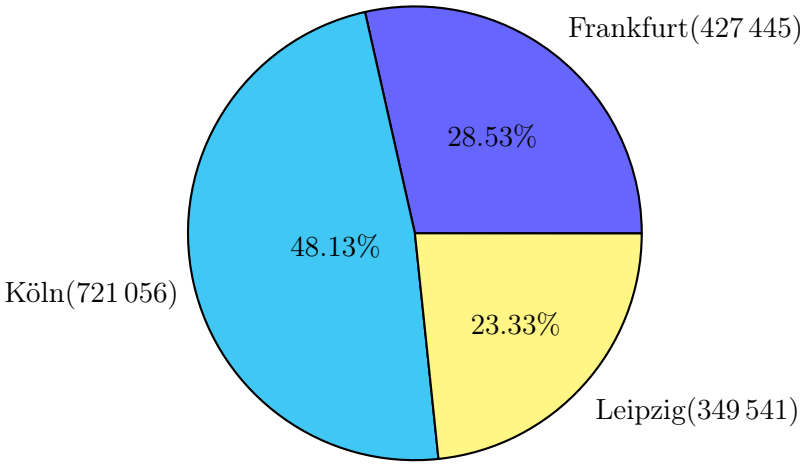


Figure 11.2: Term frequencies of the sub-corpora.

Each program has a different preprocessing on terms, which gives us different term frequencies. LeMATo uses stemming, and AntConc counts every distinct word form or summarises terms with lemmatization. To make the terms and their frequencies comparable we apply the same transformation here as previously outlined in Section 11.1 and Figure 11.1.

The significance calculations in both programs use the chi-square metric (see 7.1.1 for details and references), but they differ in their counting approach of terms. AntConc uses term frequencies, and LeMATo counts distinct text unit elements (documents, paragraphs or sentences) where a term occurs. This results in different significance even for the same measure. Observing significant terms by text unit counts is not very common in traditional lexicometrics. We decided to compare our results against raw term frequencies to get an idea how different our results are, but before we go into details of the results, we explain how text unit frequencies (e.g. document frequencies) are commonly used in Information Retrieval (IR).

Document frequencies are used to capture the generality and scope of the problem space to which a query belongs to and could be summarized in the general notion of a classification,

problem in IR. In classification we have a set of classes, and we seek to determine which objects belongs to which class.

LeMATo uses the "significant terms aggregation"[3] of Elasticsearch to extract the significant terms. The Elasticsearch site lists the following example use cases. These use cases are also based on document counts.

- Suggesting "H5N1" when users search for "bird flu" in text
- Identifying the merchant that is the "common point of compromise" from the transaction history of credit card owners reporting loss
- Suggesting keywords relating to stock symbol $ATI for an automated news classifier
- Spotting the fraudulent doctor who is diagnosing more than his fair share of whiplash injuries
- Spotting the tire manufacturer who has a disproportionate number of blow-outs

In LeMATo we are trying to find those terms which form the overall topic of the corpus we are analysing. Therefore, we used Elasticsearch significant terms aggregation, instead of the comparison of raw frequency counts. Nevertheless, we are still interested in how different our computation is in contrast to the significant terms on raw frequencies.

LeMATo calculates significant terms on three different kinds of text units: documents, paragraphs and sentences. The best approximation to the term frequencies in AntConc are the sentence counts. We take the 500 top significant terms from LeMATo and filter AntConc's results for these. We end up with approximately 495 data points, where about two-thirds of the sentence frequencies are equal to the raw frequencies and only some differ greatly in absolute counts (see top left plot in Figure 11.3, 11.4 and 11.5).

The results of these two programs are not comparable, because we compare on different counts. Even if some sentence counts are quite similar to the raw frequency counts, what really matters is the total count for the calculation of the chi-square measure. The term frequency count is for a factor of 14.85 larger than the sentence count. The difference in both counting approaches reflects also in the comparison of the ranks, which reveal a weak correlation result (see the top right plot in Figure 11.3, 11.5 and 11.4).

Finally, we have to emphasize here, that reliable results with meaningful statistics can only acchieved with at least a five time larger reference corpus [11]. Our smallest sub-corpus labelled with the tag "Leipzig" uses a 3.29 larger reference corpus.

11.3 Themes

Here in this section, we are taking a look at the themes which we have found in our corpus and our sub-corpora. We give a quick recap on the theme finding process. In LeMATo we define a query on our corpus, which is the same corpus as in our reference study [78]. The

[3] see `https://www.elastic.co/guide/en/elasticsearch/reference/1.7/search-aggregations-bucket-significantterms-aggregation.html`

query is used to retrieve the most significant terms on the matching documents. On the top N terms we build a co-occurrence matrix to build our vector space. We provide a dendrogram as a final visualisation and as an aid to find groups within the top significant terms. For more details on the implementation, we refer to Section 10.5.2.

In a first attempt we are looking at the dendrogram for the significant terms in our sub-corpus labelled with the tag "frankfurt" and we measure the co-occurrence on document level. The interesting thing is, that some local celebrities share a common sub-branch within our dendrogram. The names "Habermas", "Ignatz", "Bubis", "Suhrkamp" and "Forsyth" are together with a very frequent term "Strich". It turns out that there is a newspaper column in the newspaper "TAZ" which is called "Unterm Strich".

In an other branch we observe city names like "Stuttgart", "Bremen", "Potsdam", "Brandenburg", "Hannover". Within this branch there are also terms like "Gemeinde", "Präsident", "Institut" and "Bank", which seem to be closer related to "Stuttgart", because they share a common extra branch.

Another branch includes terms like "Bewegung", "ausländisch", "global", "globalisierung", "Veränderung" and "Debatte". There are several examples for branches which show the terms of a common topic. The most exciting example is that our dendrogram also reflects the discourse of the expansion of the airport in Frankfurt (see Figure 11.6).

At the bottom left of Figure 11.6 there are the terms "petra", "roth" and "oberburgermeisterin". Petra Roth was the first female mayor (i.e. "Oberbürgermeisterin") of Frankfurt am Main from 1995 to 2012.

The dendrogram offers not always the best legibility. Sometimes there are also stemmed forms of terms, where the reader does not immediatly knows what is meant. The best practice in such cases is to query (for example in a separate tab in the browser) those terms on LeMATo to get to know the terms context. The dendrogram is not a strict out-of-the-box grouping of terms. The dendrogram is a grouping aid, which gives a lot of freedom to the user, who has to decide on his own, whether to put this term into the group or not.

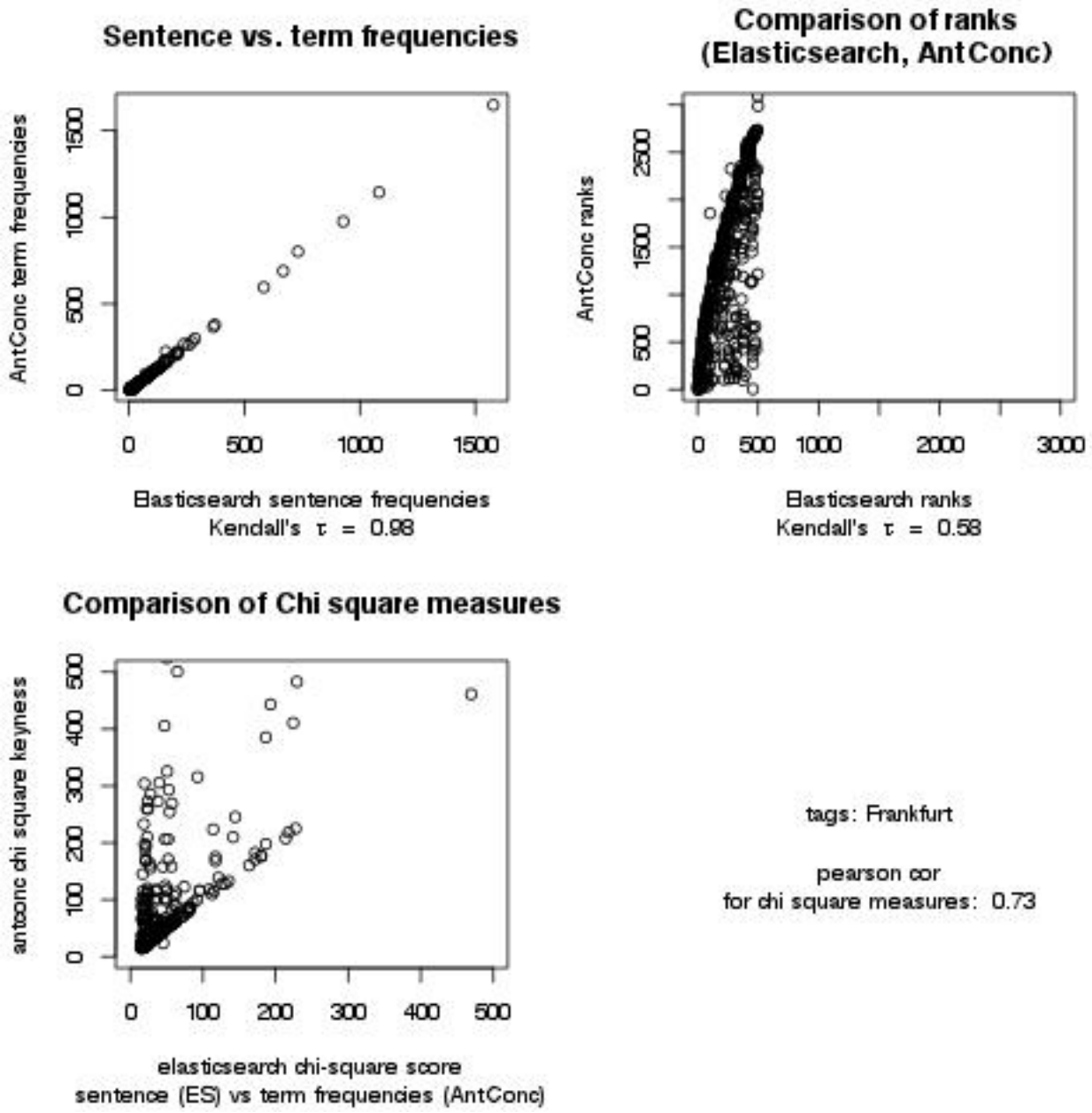


Figure 11.3: Comparison of the significance calculation in AntConc and in LeMATo for the sub-corpus labelled with the tag "Frankfurt".

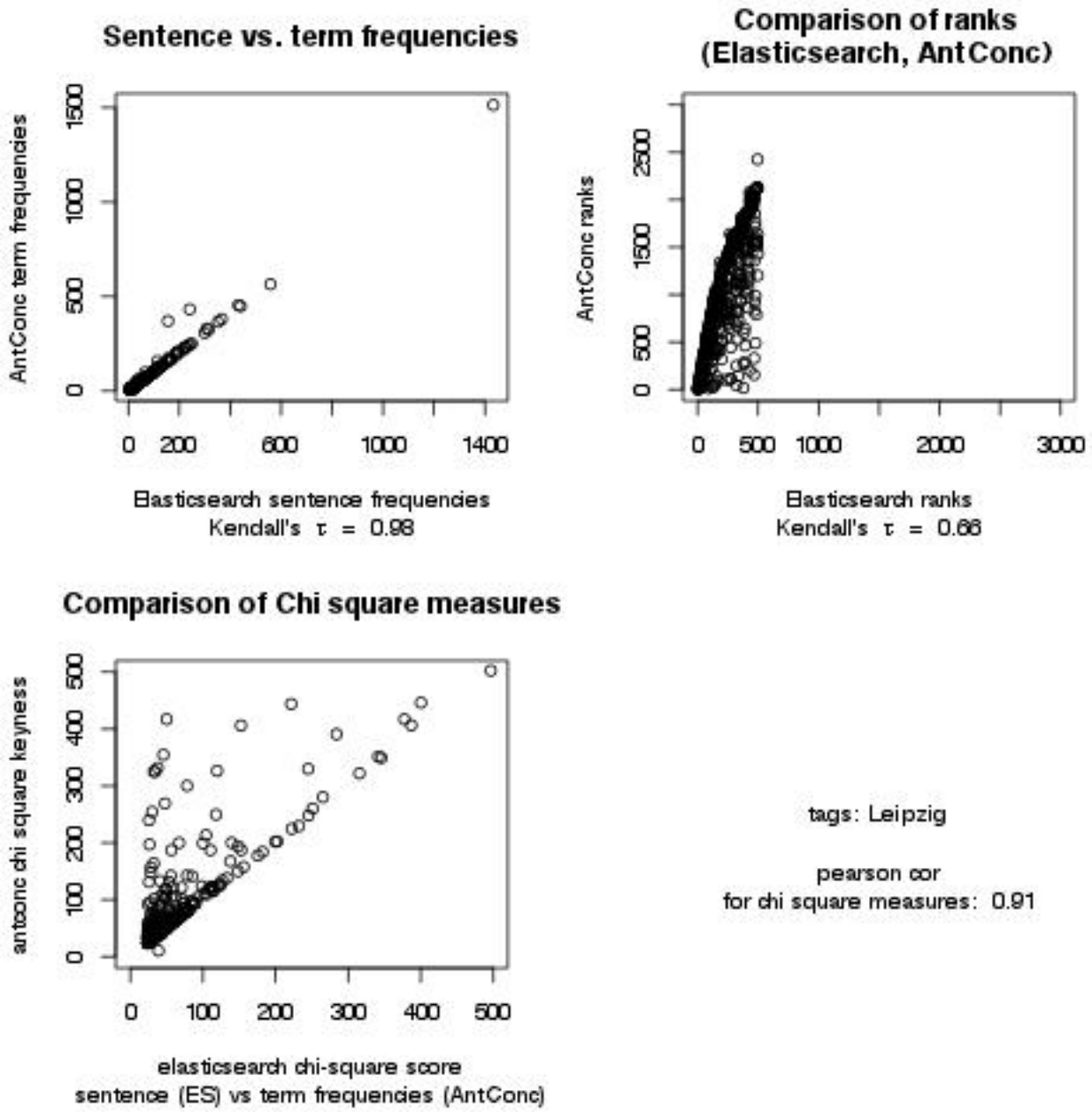


Figure 11.4: Comparison of the significance calculation in AntConc and in LeMATo for the sub-corpus labelled with the tag "Leipzig".

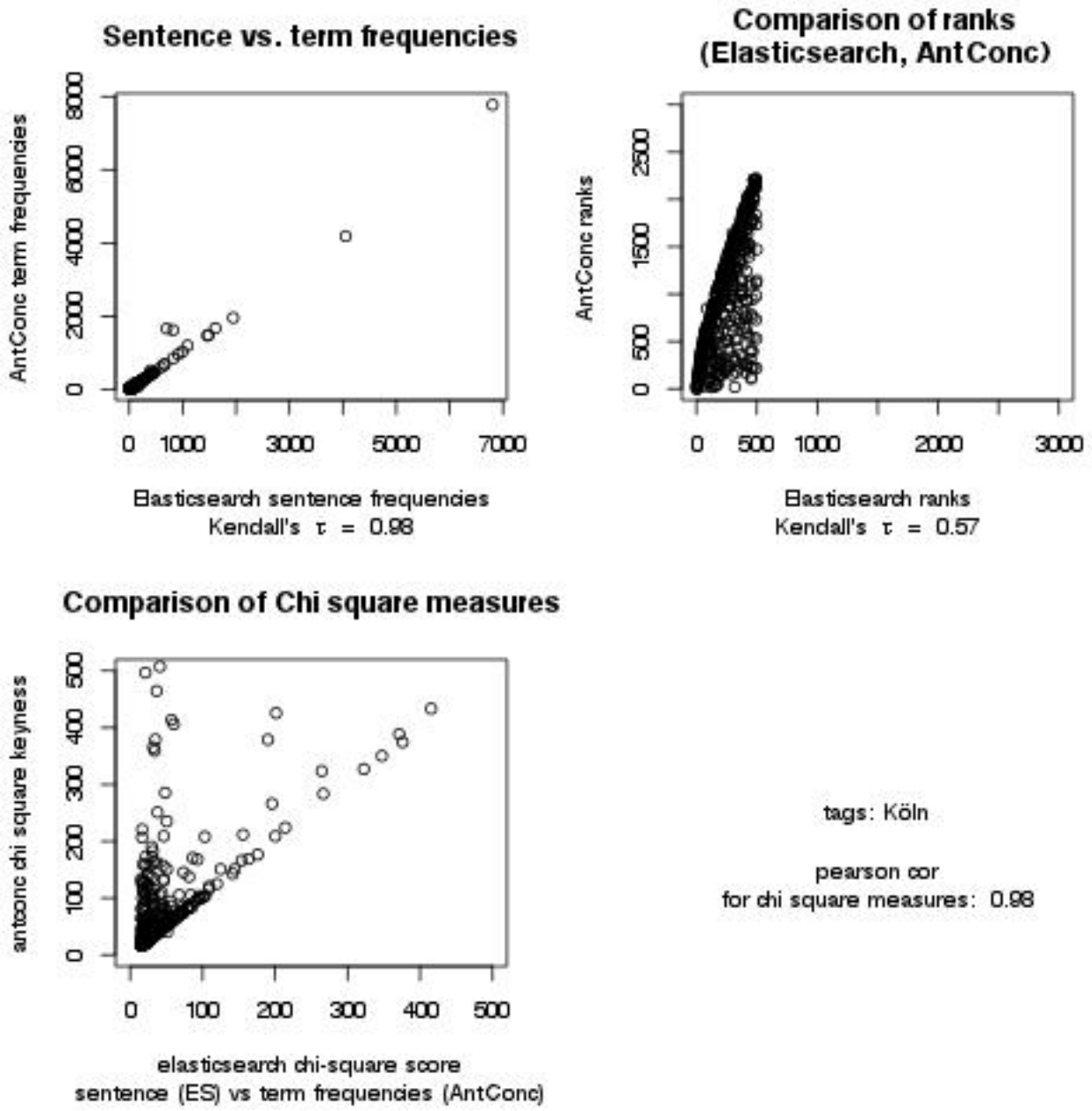


Figure 11.5: Comparison of the significance calculation in AntConc and in LeMATo for the sub-corpus labelled with the tag "Köln".

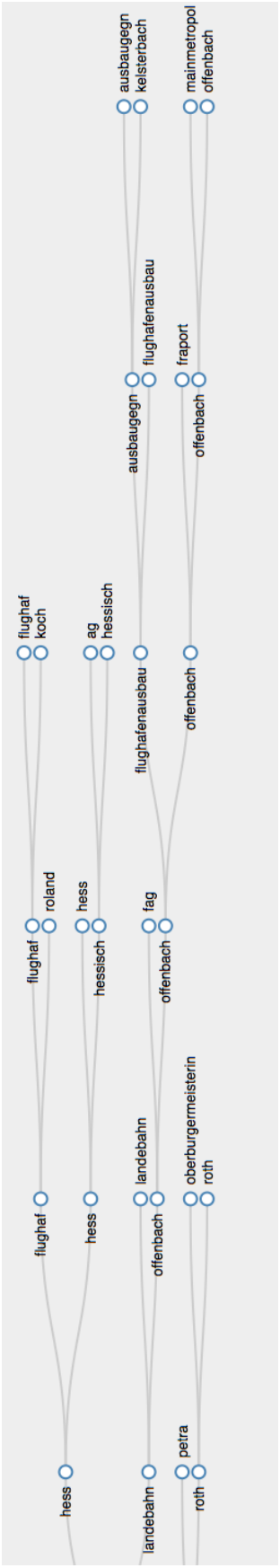


Figure 11.6: Dendrogram on the discourse of the expansion of the airport in Frankfurt.

Chapter 12

Concluding remarks

In this thesis, we have created LexicoMetric Analysis TOol (LeMATo), a tool with four lexicometric analysis methods to explore a tag annotated corpus. LeMATo is built with Elasticsearch, a powerful text search engine, which enables powerful preprocessing and querying of text. The linguistic preprocessing in LeMATo uses tokenization, filtering by stop words and stemming. Unfortunately, stemming does not support all kinds of theoretical assumptions (see Section 3.1.3), but it serves as a good basis for most of the cases. Currently, LeMATo offers only the processing of German text, but it is extensible to add other languages.

After the formulation of the hypothesis in discourse analysis, the corpus needs to be compiled regarding a particular research question. In this thesis, we have introduced our reference study [78] (see Section 4.2), which acquired newspaper articles from the LexisNexis interface. LeMATo parses the text files obtained from LexisNexis [13], creates a corpus and annotates the corpus with a publish date of each newspaper article and user defined tags. The human scientist is able to filter the corpus on articles within a period, with individual tags or with specific keywords. In the user guide (see Section B.2) we describe querying parts of the corpus with the powerful Lucene Query Domain Specific Language (DSL).

The *frequency analysis* (see 10.3) offers a good basis to explore the whole or even parts of the corpus. In a first overview page, LeMATo shows the top 500 terms with the highest document frequencies. It also provides additional information with term frequencies and Kendall's τ, which gives an idea whether a term increases or decreases in document frequencies over time. The user can select terms and see their term frequency distribution in a diachronic manner in a detail view. Furthermore, the user could even track the use or context of the terms of the newspaper articles, by clicking on the bar-chart in the detail view.

The *concordance analysis* (see 10.4) in LeMATo offers a robust context query system, which reveals the context of a queried term or word. We provide an additional view of the context with a double word tree.

The *analysis of characteristics in sub-corpora* and the *co-occurrence analysis* complete our lexicometric querying tool. In both analyses, we give an overview of the top hundred significant terms and an additional plot of text unit frequencies versus chi-square scores. On top of that, we provide a visualisation aid (i.e. dendrogram) for the user to put the terms into groups. Currently, we compare parts of the corpus to the rest of the corpus. There is a certain advantage in comparing the corpus to a language corpus. A reference corpus is larger and

gives, therefore, more accuracy in calculating significant terms. Unfortunately, we do not provide a comparison to reference corpora.

12.1 Lessons learned

It takes a lot of time to find out which set of frameworks is the right one depending on the needs. In the beginning, we looked at different search engines and how they could be applied in a web framework. For Grails [96] there are some plug-ins which work quite well for specific tasks, but if some customization is required, then the plug-in and the interaction with other frameworks could get very complex. One of the most important lessons, we have learned is, that the promise of saving a lot of time and effort through a plug-in or an additional abstraction is not always the truth.

As an example we have looked at the exciting "searchable plugin" of Grails. First of all it is not compatible with the current Grails 3.0.x version. Using the searchable plug-in comes with the cost of downgrading to Grails 2.4. There is some customisation necessary to work with Grails 2.4 and the Hibernate 4.x [94] library. Unfortunately, the searchable plug-in is not compatible to Hibernate 4. This means that it requires an update to work with current Hibernate implementations. This particular issue on the searchable plug-in requires a bug-fix on Compass. Compass is the interface to Lucene for the searchable plug-in. Unfortunately, Compass is not maintained any more, because the last release is more than five years ago.

There are also other options. For example, Hibernate and its own product Hibernate Search, but this also has not been updated since 2012, or the Solr Grails plug-in, which has not been updated since 2010.

Finally, in our opinion we took the best solution with Elasticsearch, which is not that handy like the "searchable plugin", but it offers a lot of different ways to query the text. Beyond all it is scalable, but within the development of LeMATo we did not focus on scalability.

After all, it is also not so easy to work with Elasticsearch and its Groovy or Java API, because it does not provide a Javadoc. Sometimes it is necessary to query particular fields or methods, which are not documented in the online documentation on the Elasticsearch Java API. For our purpose, we used an outdated version of an unofficial Javadoc.

We have learned a lot in this thesis about frameworks, their interactions and the lack of support for outdated programs. With this experience we have gained during the work on this thesis, we are able to familiarize ourselves with frameworks and concepts in a much better and faster way than before.

12.2 Outlook

There are some missing features which are currently not implemented in LeMATo. A *job processing system* would enable the processing of much bigger requests on the corpus. A job processing would also save results. This is currently not the case. Intermediate results are not saved in LeMATo.

The *preprocessing* in LeMATo needs customisation on different languages, and it is necessary to perform lemmatization instead of stemming.

There is also a *REST interface to LexisNexis*, but unfortunately, the University of Vienna does not have access to that. With the use of the REST interface of LexisNexis within LeMATo, it would be possible to add additional metadata to the corpus, and we would enable a much cleaner parsing of the text files. Currently, if the format of the text files changes, it will not be possible to import any LexisNexis text files into LeMATo.

Additional work has to be done for obtaining and calculating co-occurrence matrices. At the moment this step is much too slow. It might be possible to write an *Elasticsearch plug-in* to obtain the co-occurrence matrix directly from Elasticsearch and reduce the latency for the significance analysis.

At the time of writing it is not planned to continue the work on LeMATo. We provide an online repository[1] where it is possible to download the code. We also give instructions to run LeMATo on different operating systems.

With LeMATo we provide a good basis for a human scientists to perform a lexicometric analysis on text, but currently LeMATo is only a prototype. This means there is still room for improvements left.

[1] see `https://github.com/perdacherMartin/LeMATo`

Appendices

Appendix A

Abstract

A.1 English abstract

Discourse analysis aims to reveal connections between text or speech to institutional structures, furthermore, discourse analysis connects text to historical or social context. Several disciplines analyse a text in different ways. In social sciences and humanities, content analysis has become the standard approach. Content analysis starts with a research question and the formulation of appropriate hypotheses. These presumptions before analysing text are incompatible with discourse analysis. In lexicometrics, the process of interpretation is shifted to the end of the analysis and is, therefore, suitable for discourse analysis. Here we present LexicoMetric Analysis TOol (LeMATo), a web application which analyses the text in a lexicometric way. LeMATo is built upon the ideas published in [33] and contains their four approaches to analyse text. LeMATo enables the *analysis of frequencies* and a *concordance analysis* on a corpus or particular parts of it. In the *analysis of sub-corpora* and *the co-occurrence analysis* LeMATo examines significant terms of text units in contrast to individual parts of the corpus. We provide a download of LeMATo from `https://github.com/perdacherMartin/LeMATo`.

A.2 Deutsche Zusammenfassung

Diskursanalyse versucht Verbindungen zwischen Text oder Sprache und institutionellen Strukturen herzustellen. Darüber hinaus verbindet die Diskursanalyse Text mit historischem und sozialem Kontext. Text wird in verschiedenen Disziplinen auf verschiedene Weise analysiert. In den Sozial- und Geisteswissenschaften ist die Inhaltsanalyse die Standardmethode, um Text zu analysieren. Die Inhaltsanalyse beginnt mit der Formulierung einer Forschungsfrage und den dazugehörigen Hypothesen. Die Annahmen die man vor der Textanalyse trifft, sind jedoch unvereinbar mit der Diskursanalyse. In der Lexikometrie wird die Interpretation an das Ende des Forschungsprozesses verlagert und ist daher für die Diskursanalyse geeignet. Hier präsentieren wir das LexikoMetrische Analyse TOol (LeMATo), eine Webanwendung, welche den Text in lexiko- metrischer Weise analysiert. LeMATo basiert auf den in [33] veröffentlichten Vorstellungen und enthält vier Ansätze, um Text zu analysieren. LeMATo ermöglicht die *Analyse von Häufigkeiten* und die *Konkordanzanalyse* für einen Textkorpus oder bestimmten Teilen davon. Darüber hinaus untersucht LeMATo in der *Analyse von Subkorpora* und in der *Kookkurrenzanalyse* signifikante Terme in Texteinheiten im Vergleich zu bestimmten Bereichen des Korpus. Wir bieten einen Download von LeMATo unter: `https://github.com/perdacherMartin/LeMATo` an.

Appendix B

User Guide

B.1 Corpus definition

The first step is to assemble the corpus regarding a particular research question. LeMATo in its current version supports text files in the LexisNexis text format. LexisNexis [3] pioneered in 1970 the electronic accessibility of journalistic documents. It is the largest electronic database for legal and public-records related information. For students, LexisNexis is only accessible on the university network (`http://e-solution.lexisnexis.de/KSH/de/index.html`). In the following we are trying to reconstruct the corpus of our reference study [78]. In a first step, we query the LexisNexis database with the queries provided in Listing B.1 in the German-speaking press between the years 1999 and 2005.

Our reference study limits its corpus to four distinct newspapers ("Süddeutsche Zeitung", "die taz", "Der Spiegel" and "Stern"). Unfortunately, the newspaper articles from "Süddeutsche Zeitung" are not available anymore in the LexisNexis environment, so instead of having 2.3 million words we have 1.5 million words.

Each file, obtained from LexisNexis, is now ready for import into LeMATo, but, first of all, we need to add a new corpus to LeMATo. Therefore, we select "Manage corpora" and "New Corpus" from the side menu. Enter a "name" and a "description" of the corpus you want to add. In our case, we call it "Mattisek 2008" and we enter "Die neoliberale Stadt von Annika Mattisek (2008)" as a description of our corpus (see Figure B.1).

To import the file into LeMATo, we select "Extend Corpora" from the side menu and "New File" from the navigation bar. In the current panel (see Figure B.2), we choose our corpus from the drop-down menu, select the file from the dialogue and specify some tags. Tags are used to separate between the different sub-corpora. In our case, we use the tags "Frankfurt",

Listing B.1: LexisNexis queries

```
Frankfurt W/10 Stadt
Leipzig W/10 Stadt
Koeln W/10 Stadt
```

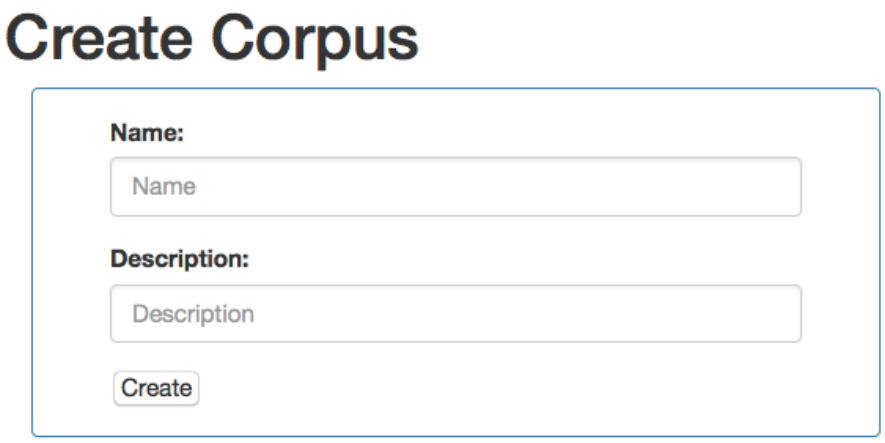


Figure B.1: Creating a new corpus with name and description.

because the file originates from the "Frankfurt" query in Listing B.1 and "DerSpiegel", because the articles in this file are from the newspaper "DerSpiegel".

We import each file obtained from LexisNexis into LexicoMetric Analysis TOol (LeMATo). In our case we have nineteen files and 12.5 MB and the import took nine minutes. The files have an average length of 660 kB, and for that it took 24 seconds per file on average.

B.2 Querying in LeMATo

Elasticsearch is a powerful tool for querying text. To deliver the full power of Elasticsearch to the end user, we enhanced LeMATo with an excellent query parser syntax: "Apache Lucene Query Parser Syntax"[1]. We have designed LeMATo to have a tool, which is easy to use. For an advanced use of LeMATo, there is the possibility to query a particular part of the corpus in more detail. For example, to perform the four fundamental analyses introduced in [33], there is no need to know the Lucene querying syntax. The "Lucene Query Parser Syntax" enables filtering on particular parts of the corpus (e.g. time periods, tags) for a better understanding of particular parts of the corpus.

Each text file from LexisNexis contains a lot of articles. These articles are equal to documents in LeMATo. Furthermore it is also possible to query paragraphs or sentences. This is important for querying the close proximity of terms. Sometimes the outcome of significant terms for the sentences is poor, because there are not enough sentences for the word of interest. Therefore LeMATo provides the possibility to expand the queries on paragraphs.

Each one of the three text units, sentence, paragraph or document is organized the same way. It can be queried on three different fields:

- **tags** (tags entered in corpus definition)
- **publishDate** (extracted from the text files)
- **textBody** (the text itself, default field)

[1] `https://lucene.apache.org/core/2_9_4/queryparsersyntax.html`

Figure B.2: Importing a LexisNexis file and adding it to the corpus.

Listing B.2: LexisNexis queries

```
tags: "Frankfurt"
```

Filtering the corpus on tags: Lucene supports fielded data. When performing a search, there is a need to specify a field. Searching on fields is done by typing the field name followed by a colon ":" and the term. As an example to query the corpus on the tag "Frankfurt", the query is formulated as shown in Listing B.2.

There are some cases where it is necessary to filter on two tags. The best way to perform a query on two tags (e.g. "Frankfurt" and "DerSpiegel") is to use the AND operator, which is depicted in Listing B.3. For filtering on every article containing either "Frankfurt" or "DerSpiegel", change the "AND" in Listing B.3 to an "OR".

Filtering on the dates or periods: The date fields, like "publishDate" in LeMATo is used slightly different to the "tags" field. Here we have the possibility to filter the corpus on periods. The date format in LeMATo is organized with a format string of "yyyy-mm-dd". The first four digits represent the year, followed by the month in a two-digit format and a two-digit day. There is a "-" separator between the year and month as well as between the month and the day specifier. The most intuitive use case for dates is query-ranges.

A range query allows one to match documents between a lower and an upper bound. As an example, see Listing B.4 to query all documents for the year 2002. This query will match all doc-

Listing B.3: Filter on sub-corpora

```
tags: "Frankfurt" AND tags: "DerSpiegel"
```

Listing B.4: Filter on periods

```
publishDate: [2002-01-01 TO 2002-12-31]
```

Listing B.5: Filter on text

```
Steuer
```

uments, whose "publishDate" is between the values `2002-01-01` (inclusive) and `2002-12-31` (inclusive).

Filtering on the text: A query is broken up into terms and operators. In the previous paragraphs operators like "AND" and "OR" were introduced. Terms could be single terms like "Fußball" or phrases like "günstige Steuern", two or more words within double quotes. The text field "textBody" is the default field in LeMATo. The query in Listing B.5 is equal to the query: `textBody: "Steuer"`.

B.3 Frequency analysis

The first analysis in LeMATo which reveals the insides of a corpus, is the frequency analysis. Choosing "Frequency analysis" from the side menu, will start this analysis. The frequency analysis requires some parameters to get started (see Figure B.3):

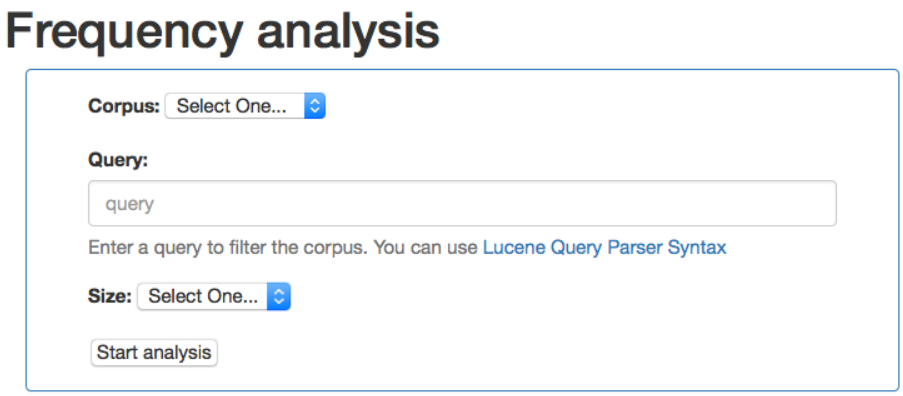


Figure B.3: Parameter selection for frequency analysis.

- The corpus itself.
- A Lucene query (optional) to filter the corpus in more detail. Leave this field empty if you want to analyse the corpus as a whole.
- the size stands for how many words you want to look at. Unfortunately, LeMATo does not offer to query all terms.

The higher the size here, the longer it will take to query the corpus (approximately 4 seconds for a size of 100). For more details on latency see Section 10.6.

There are two results of this analysis. The first result contains four bar charts, which give information of the overall frequency distributions of the corpus (see Figure B.4).

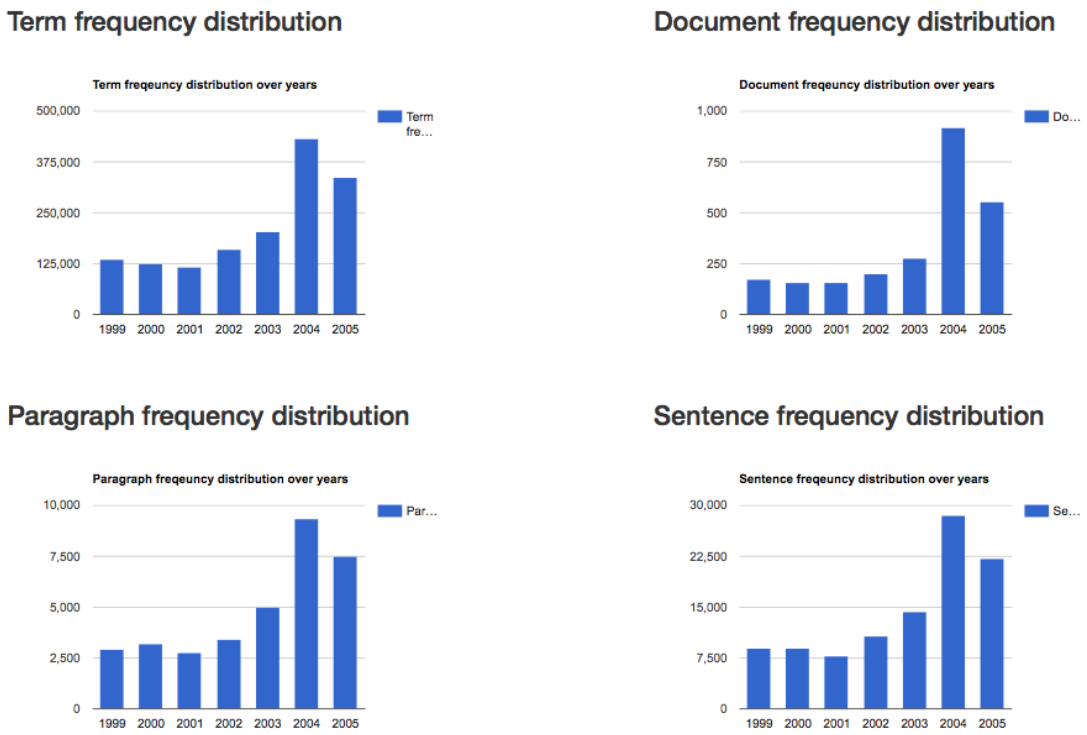


Figure B.4: Frequency distributions for the overall corpus. From left to right: term frequencies, document frequencies, paragraph frequencies and sentence frequencies.

Calculating term frequencies is not for free in Elasticsearch and often requires an additional query. At a first glance, we are approximating term frequencies with sentence or paragraph frequencies and give additional information with document frequencies, after clicking on a term, we also provide term frequencies.

The second result is a table with six columns, which displays the top-size-frequent words in the corpus (see Figure B.5). The left most column enables the researcher to select terms from the table, which are displayed in a diachronic view by simply clicking on the "Show detail for selected" button.

The diachronic view gives insides to all frequencies of the terms and their diachronic distribution (see Figure B.6).

Sometimes it is important to view the occurrences of a word in more detail. Therefore, it is possible in the diachronic view to click on a bar of the charts to get a list of documents, paragraphs or sentences with the occurrence of the term, filtered for the corresponding year (see Figure B.7).

Furthermore, it is possible to view the document itself. To see the term of interest at the first glance, we provide a highlighting of the term.

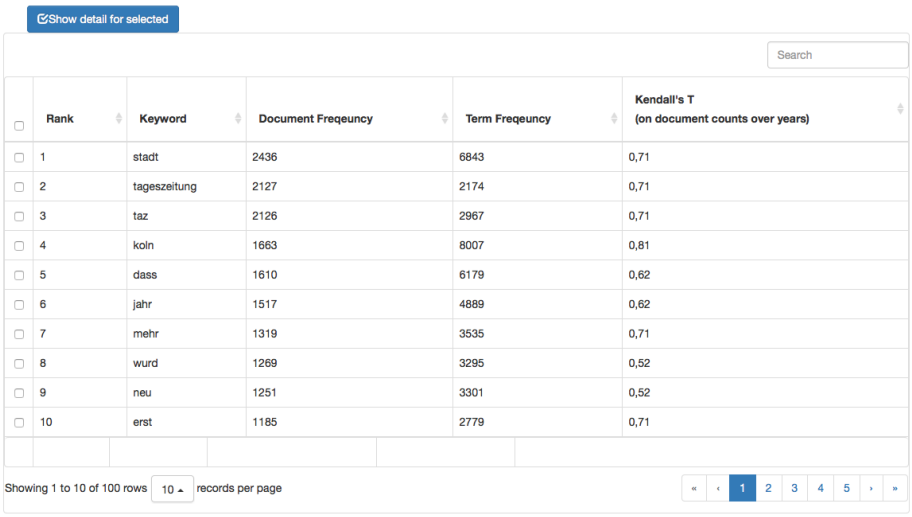


Figure B.5: Table of words with top frequencies within the corpus. The columns from left to right: rank (order of highest document frequency, keyword (term), document frequency, term frequency and kendall's τ).

B.4 Concordance analysis

The concordance analysis reveals every use of a keyword in its specific context. The parameters for this analysis are the corpus itself, the keyword of interest (centered word) and an optional Lucene query to filter on particular parts of the corpus (see Section B.2).

In our case, we are using the word "Berlin" and filter for the tag "Köln" (see Figure B.8). The result of the concordance analysis is presented with two different features. First, we are showing a "double word tree"[2] (see Figure B.9) and, second, we are listing all occurrences of "Köln" in a table (see Figure B.10).

Sometimes the meaning of a stemmed term is not clear. In this case the concordance analysis can help to look up a stemmed term and how it is used in the documents.

B.5 Characteristics analysis of sub-corpora

For the characteristics analysis of sub-corpora, choose "Sub-corpora analysis" from the sidebar menu. The parameters for this analysis are the corpus itself, the tags of the sub-corpora of interest, the size (count of words for the result) and the unit of text, where you want to measure the significance (document, paragraph or sentence).

We are querying the corpus for the tags "Frankfurt", "Köln" and "Leipzig" to see their differences. For each tag we provide a table with the calculated score and the counts for each word in the subset and in the superset (see Figure B.11).

Furthermore we provide a zoom able scatter chart, which shows the score and the document frequency relation (see Figure B.12).

[2] `https://developers.google.com/chart/interactive/docs/gallery/wordtree`

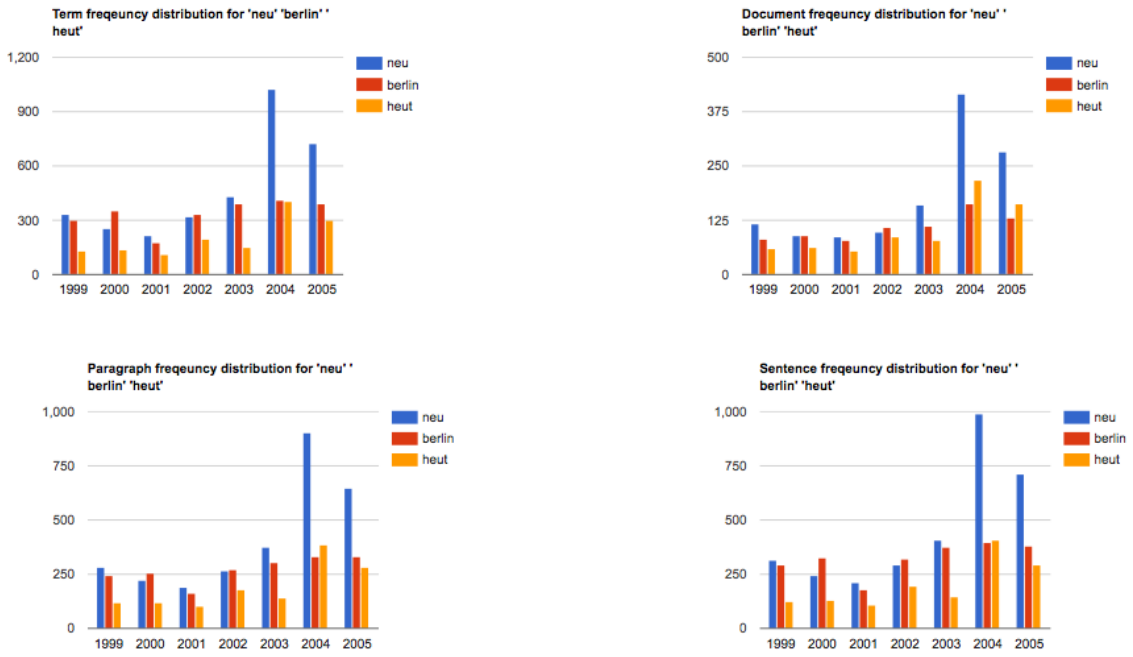


Figure B.6: Diachronic view for selected terms. From left to right: term frequencies, document frequencies, paragraph frequencies and sentence frequencies.

On top of that we are trying to visualize the relation of the words in a dendrogram. At the moment, the dendrogram is organized as a binary tree. The dissimilarity of the groups is not included in the tree until now. The leaves in the dendrogram tree are labelled with the significant words. The nodes in the tree are labelled with the most frequent word in the sub-branch. The composition of the dendrogram is completely calculated by the s-space package.

A part of the dendrogram is depicted in Figure B.13. The rule of thumb to read a dendrogram of this format is: If there are a lot of deep hierarchies, this means the dendrogram grows in width, then the group tends to be homogeneous. If the dendrogram grows in height, then there are more heterogeneous groups. The elements of each branch can be visualized by right-clicking on a node, then every significant term in the sub-branch is displayed in a new browser window (see Figure B.14).

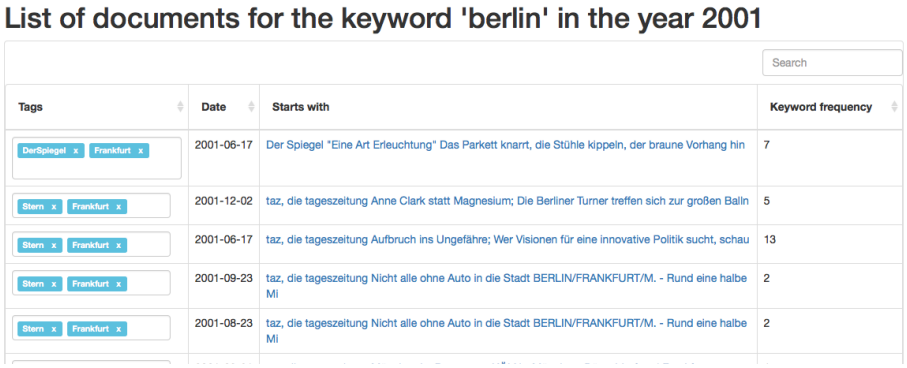


Figure B.7: List of documents with the occurrence of the term “berlin” in the year 2001. The columns of the table from left to right: tags, date, starts with (the first 100 characters of the document), keyword frequency (how often does the term occur in the document)

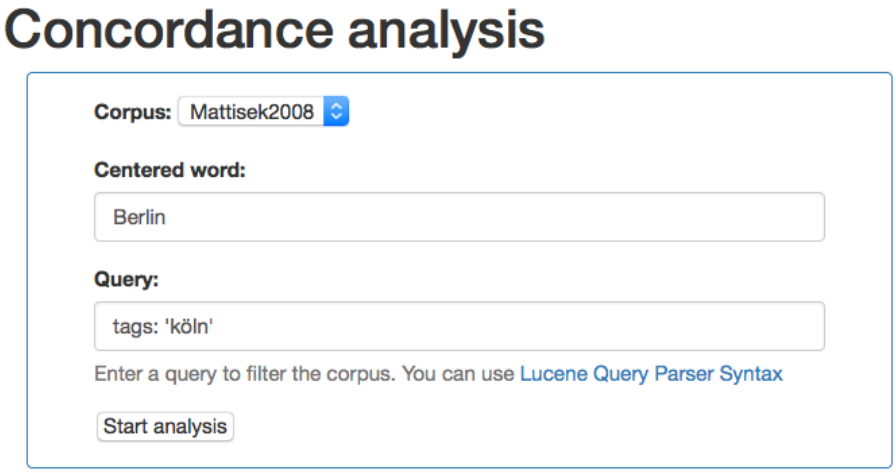
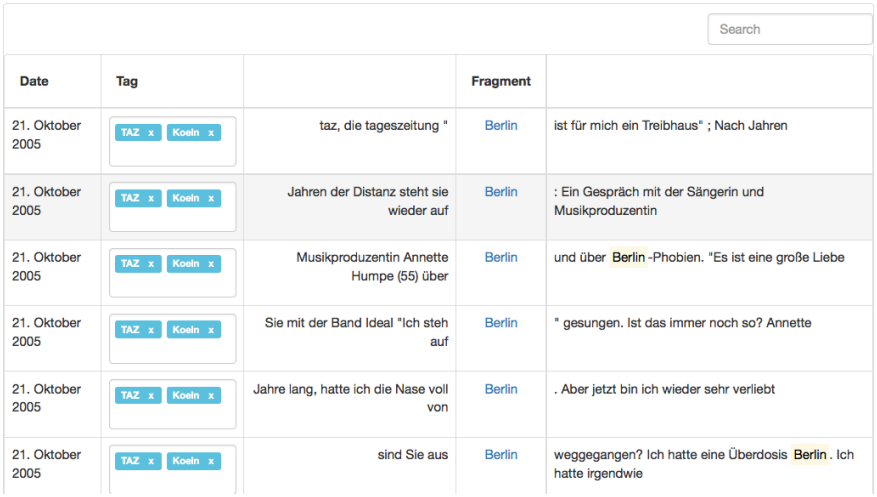


Figure B.8: Parameter selection for the concordance analysis.

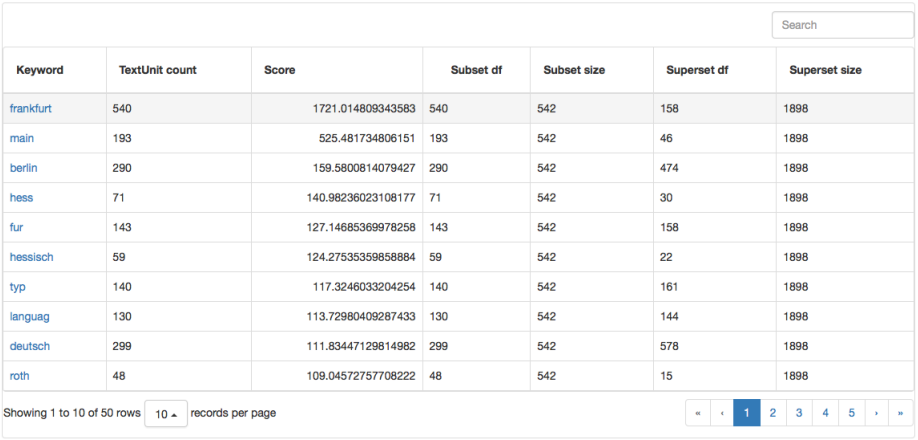
Figure B.9: Double word tree for all occurrences of the term “Berlin”.



Search

Date	Tag		Fragment	
21. Oktober 2005	TAZ x Koeln x	taz, die tageszeitung "	Berlin	ist für mich ein Treibhaus" ; Nach Jahren
21. Oktober 2005	TAZ x Koeln x	Jahren der Distanz steht sie wieder auf	Berlin	: Ein Gespräch mit der Sängerin und Musikproduzentin
21. Oktober 2005	TAZ x Koeln x	Musikproduzentin Annette Humpe (55) über	Berlin	und über Berlin -Phobien. "Es ist eine große Liebe
21. Oktober 2005	TAZ x Koeln x	Sie mit der Band Ideal "Ich steh auf	Berlin	" gesungen. Ist das immer noch so? Annette
21. Oktober 2005	TAZ x Koeln x	Jahre lang, hatte ich die Nase voll von	Berlin	. Aber jetzt bin ich wieder sehr verliebt
21. Oktober 2005	TAZ x Koeln x	sind Sie aus	Berlin	weggegangen? Ich hatte eine Überdosis Berlin . Ich hatte irgendwie

Figure B.10: Concordance analysis table for all occurrences of the term “Berlin”. The columns from left to right: date, tags and the text fragment, where the word occurs.



Search

Keyword	TextUnit count	Score	Subset df	Subset size	Superset df	Superset size
frankfurt	540	1721.014809343583	540	542	158	1898
main	193	525.481734806151	193	542	46	1898
berlin	290	159.5800814079427	290	542	474	1898
hess	71	140.98236023108177	71	542	30	1898
fur	143	127.14685369978258	143	542	158	1898
hessisch	59	124.27535359858884	59	542	22	1898
typ	140	117.3246033204254	140	542	161	1898
languag	130	113.72980409287433	130	542	144	1898
deutsch	299	111.83447129814982	299	542	578	1898
roth	48	109.04572757708222	48	542	15	1898

Showing 1 to 10 of 50 rows 10 records per page « ‹ 1 2 3 4 5 › »

Figure B.11: Table for the characteristics analysis of subcorpora for the tag: “Frankfurt”

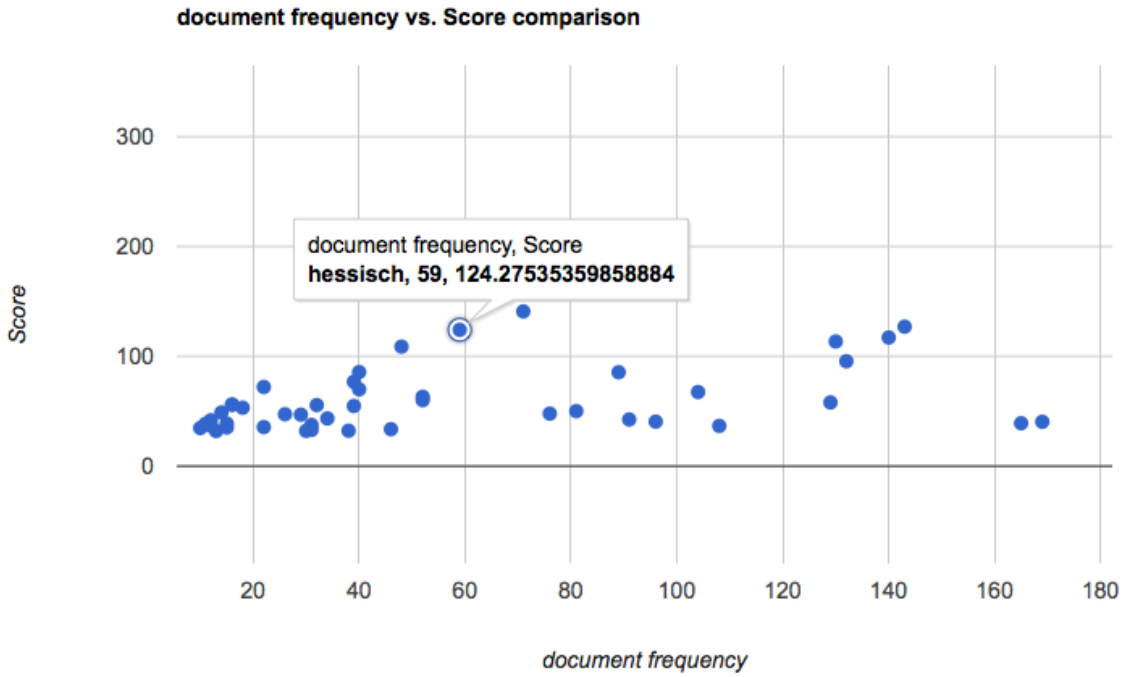


Figure B.12: Scatterchart of document frequencies against the score (Chi-square significance).

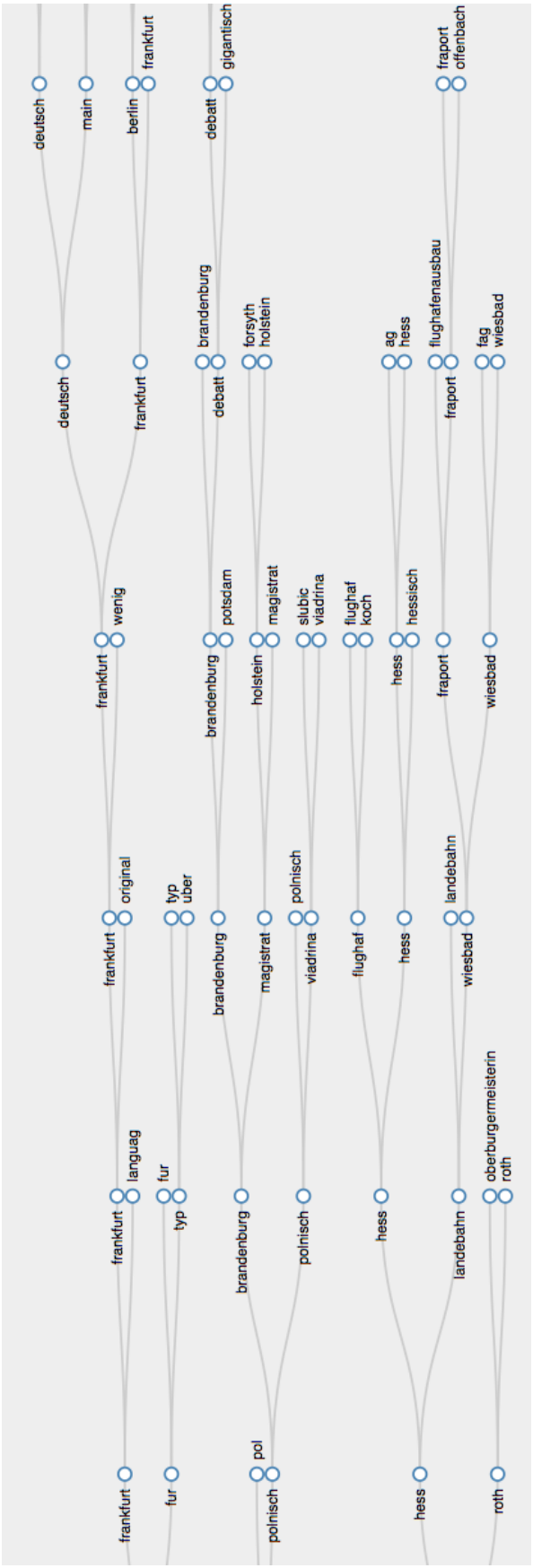


Figure B.13: Part of dendrogram for the tag "Frankfurt" (size: 50 terms).

term	textunit frequency	score	remove
landebahn	22	72.47352392549412	✖
flughafenausbau	10	35.162559032997734	✖
fraport	14	49.308749311730544	✖
offenbach	11	38.6947386970782	✖
fag	12	37.163482654618576	✖
wiesbad	15	39.205062278834674	✖

Figure B.14: All significant terms of a subbranch. Obtained through a right-click on a node in the dendrogram.

Appendix C

Supplementary material

Keyword	LeMATo	AntConc

Table C.1: Top 500 frequencies of our reference corpus with the term frequencies observed in LexicoMetric Analysis TOol (LeMATo) and AntConc (ordered by Elasticsearch document frequencies). Higher term frequency scores are marked for LeMATo in blue and AntConc in red.

stadt	6843	6849
tageszeitung	2174	2174
taz	2967	2969
koln	8007	8028
dass	6179	6179
jahr	4889	4890
mehr	3535	3535
wurd	3295	3295
neu	3301	3301
erst	2779	2779
schon	3144	3144
sei	2709	2713
seit	2432	2435
sagt	2848	2848
gross	2135	2135
gibt	1886	1925
deutsch	2781	2781
imm	2007	2007
ganz	1803	1803
lang	1772	1772
zwei	1511	1513
viel	1724	2897
wenig	1465	1465
muss	1482	3037
gut	1599	1600
berlin	2362	2366
weit	1134	1134

Table C.1 – continued from previous page

heut	1421	1421
soll	1187	2895
beim	1162	1162
euro	2017	2019
rund	1226	1226
frankfurt	1946	1948
bereit	1088	1088
ab	1130	1130
mal	1493	1495
konnt	1154	1154
zeit	1210	1210
geht	1170	1195
dabei	1084	1084
etwa	1112	1112
deutschland	1745	1745
mensch	1513	1514
drei	1084	1084
macht	1088	1091
hatt	1049	1049
land	1336	1336
million	1563	1563
alt	1297	1297
letzt	938	938
arbeit	1139	1139
end	993	993
klein	1103	1103
allerding	794	794
dafur	853	853
grun	1376	1376
komm	869	869
tag	1184	1184
geld	1020	1020
geb	749	749
eig	946	946
war	880	4493
weg	833	1411
prozent	1399	1399
fall	874	874
steht	816	816
kurz	757	757
halt	774	774
word	822	822
lass	786	786
cdu	1270	1270
jung	1086	1086
kommt	778	779
leb	1097	1097

Table C.1 – continued from previous page

woch	835	835
spd	1140	1140
gerad	756	757
frag	837	837
fast	755	755
ja	1138	1138
geh	716	717
heisst	715	715
hoh	771	771
wer	839	842
steh	655	655
haus	1009	1009
gar	675	675
teil	697	702
beid	777	777
kam	775	775
erklart	603	603
recht	780	780
deshalb	684	684
sogar	640	640
kaum	673	673
den	664	17856
moglich	596	596
politik	820	821
beispiel	635	635
leipzig	1705	1708
wichtig	632	632
grund	577	577
offentlich	689	689
seh	624	625
spat	795	795
uns	746	1764
zahl	733	733
stell	600	600
musst	648	648
gab	678	684
sieht	555	555
meist	644	644
nach	583	6010
gleich	621	621
konn	604	604
weiss	718	718
hamburg	1052	1056
bess	558	558
vier	640	640
schliesslich	551	551
vergang	618	618

Table C.1 – continued from previous page

davon	554	554
allein	593	593
statt	518	518
zweit	584	584
einfach	555	555
20	666	0
gest	508	508
neb	510	510
wohl	514	514
liegt	544	544
mann	928	928
anfang	493	493
strass	787	788
fruh	626	626
frau	1004	1004
welt	777	777
politisch	673	673
funf	602	602
projekt	690	692
monat	546	546
eigentlich	522	522
zeigt	527	527
gemacht	483	483
platz	638	638
bekomm	505	506
kind	1114	1114
zehn	536	536
zusamm	457	457
dageg	451	451
frei	533	533
genau	501	501
jedoch	475	475
zuruck	507	507
naturlich	499	503
sprech	454	454
stark	485	485
spiel	902	904
lasst	546	546
laut	437	437
vielleicht	544	544
inzwisch	523	523
leut	610	610
gehort	460	460
bleibt	420	420
lag	490	490
ausserd	404	404
darauf	446	446

Table C.1 – continued from previous page

schnell	514	514
erhalt	475	475
klar	463	463
der	485	50269
fordert	414	414
schw	510	510
stellt	427	427
gemeinsam	455	455
bleib	445	445
ging	477	481
richtig	459	459
bild	655	656
hoch	466	466
find	438	438
ehemalig	494	494
genannt	415	415
bislang	436	436
eben	448	448
burg	596	596
eher	427	427
warum	506	507
oft	461	462
grosst	510	510
bericht	390	390
bekannt	440	440
geplant	418	418
nie	576	576
tun	427	427
1	616	0
30	470	0
stadtisch	492	492
sowi	400	400
besonders	416	416
einzig	400	400
sag	536	536
sich	410	11839
stand	462	462
fest	415	415
ort	483	483
lieb	550	551
nah	429	429
damal	501	501
bau	577	577
bish	403	403
international	484	484
gilt	411	411
munch	502	504

Table C.1 – continued from previous page

tatsachlich	375	375
thema	396	396
darf	397	397
fur	1622	1623
off	431	431
schlecht	394	394
typ	308	308
wiss	429	429
darub	374	374
entscheidung	397	397
zukunft	436	436
kost	468	468
original	312	312
rat	431	431
trotz	373	373
knapp	369	369
15	454	0
jahrig	444	477
uhr	893	894
chef	502	502
gern	414	414
raum	414	414
schwarz	520	520
geschicht	671	671
stund	490	490
mark	814	815
derzeit	335	335
nehm	359	359
gegenub	360	360
mitglied	427	427
uberhaupt	382	382
pro	483	485
probl	369	369
rot	451	451
dusseldorf	558	562
durf	364	364
findet	364	364
pet	390	392
geword	369	369
mitt	362	362
liess	394	394
paar	457	457
50	374	0
folg	348	348
hand	391	391
languag	274	274
sech	390	35

Table C.1 – continued from previous page

unternehm	546	546
meint	325	325
wirklich	360	360
sollt	355	355
nrw	469	476
ost	752	753
schul	723	723
spiegel	689	689
sozial	502	502
plan	380	381
best	374	374
wahl	528	528
deutlich	326	326
bring	325	325
chanc	357	357
oberburgermeist	359	359
mocht	338	338
mitarbeit	438	438
fuhr	326	326
ide	366	366
insgesamt	324	324
daran	314	314
kunftig	346	346
partei	595	595
ziel	338	338
erfolg	319	319
red	385	385
lieg	313	313
versuch	348	348
obwohl	318	318
grupp	384	384
sach	313	313
setzt	331	331
wort	379	379
zud	306	306
besuch	392	392
bald	333	333
nam	385	385
versucht	302	302
fuhrt	302	302
18	402	0
40	360	0
acht	298	298
durft	307	307
zunach	289	289
art	364	366
niemand	341	341

Table C.1 – continued from previous page

main	315	315
problem	306	306
kritik	299	299
100	305	0
angebot	344	344
europaisch	422	422
leid	299	299
kommun	419	419
mai	409	409
privat	370	370
blick	321	322
ein	327	11309
famili	441	441
vorsitzend	284	284
sitz	292	292
trotzd	276	276
verwaltung	375	375
wolfgang	353	353
uber	657	657
genug	274	274
krieg	522	523
auss	279	279
kultur	375	378
michal	320	321
halb	267	267
scheint	288	288
voll	298	298
han	322	322
buch	502	495
septemb	329	329
zeig	265	265
25	345	0
preis	363	363
sprach	351	351
immerhin	258	258
abend	313	313
roll	314	314
ubrig	273	273
vollig	279	279
west	475	475
2003	288	0
europa	451	451
nachd	260	260
fritz	242	242
glaub	317	317
kampf	350	350
je	312	312

Table C.1 – continued from previous page

verlor	276	276
2004	418	0
berichtet	243	243
direkt	267	267
setz	247	247
ford	277	135
gegeb	252	252
offenbar	242	242
gesprach	246	246
braucht	272	272
ebenfall	247	247
morg	300	300
trag	242	242
bund	322	322
ebenso	278	278
dennoch	250	250
herr	328	328
reich	350	350
zieh	249	249
14	329	0
2005	544	0
ahnlich	267	267
jugendlich	404	404
jurg	315	315
16	310	0
bonn	408	408
fehl	231	231
finanziell	243	243
stimm	301	301
vertret	245	245
zumind	241	241
fand	252	252
zahlt	252	252
gesellschaft	281	281
hilf	283	283
nordrhein	285	285
zusatzlich	243	243
beginn	246	246
entsprechend	223	223
konzept	284	284
wollt	261	261
60	249	0
zuvor	258	258
10	376	0
erfolgreich	268	268
freund	355	355
nacht	352	353

Table C.1 – continued from previous page

rhein	338	338
geschaft	313	313
leer	250	251
2	344	0
2000	264	0
amt	324	324
endlich	254	254
spielt	237	237
kopf	282	282
staat	368	368
wohnung	375	375
beteiligt	236	236
brauch	236	236
gekomm	233	233
such	245	245
danach	233	233
erzahlt	316	317
interess	232	232
jahrlich	250	250
gewinn	260	260
lebt	274	275
2002	272	0
gestellt	214	214
schaff	229	229
lauf	220	220
verschied	240	240
kun	425	427
arbeitet	255	255
bereich	226	226
fehlt	221	221
leicht	220	220
treff	252	252
blieb	231	231
leit	220	220
aufgab	224	224
sieb	241	241
bestimmt	239	239
erstmal	211	211
gehor	214	214
januar	264	264
programm	259	259
veranstaltung	228	228
12	315	0
einzeln	236	236
forderung	258	258
sinn	219	219
ang	275	275

Table C.1 – continued from previous page

denk	258	258
milliard	455	455
ess	319	320
hart	237	237
rathaus	230	230
reicht	206	206
sorg	219	219
spricht	221	221
wert	231	231
entwicklung	219	219
falsch	240	240
geschaftsfuhr	239	239
moglichkeit	213	213
beschloss	205	205
erreicht	218	218
historisch	260	260
kenn	229	229
aug	217	62
dah	205	205
dritt	212	212
juli	252	252
met	291	291
wirtschaft	239	239
betroff	224	224
musik	358	358
nutz	201	201
personlich	209	209
plotzlich	258	258
schramma	329	329
besond	208	208
ern	259	276
mindest	206	206
verfugung	197	197
aktion	266	266
aktull	213	213
darin	190	190
zentral	214	214
betrieb	233	233
gefordert	210	210
jedenfall	203	203
kraft	240	240
verkauf	262	262
entschied	203	203
hall	333	335
hauptstadt	245	245
job	323	323
17	281	0

Table C.1 – continued from previous page

kommend	191	191
marz	265	265
polizei	401	402

Bibliography

[1] *British National Corpus, version 3 (BNC XML Edition) Distributed by Oxford University Computing Services on behalf of the BNC Consortium. URL: http://www.natcorp.ox.ac.uk/*. University of Oxford, Research Technologies Service, 2007.

[2] Angermuller, J. *Poststructuralist Discourse Analysis: Subjectivity in Enunciative Pragmatics*. Postdisciplinary Studies in Discourse. Palgrave Macmillan, 2014.

[3] Anthony, L. AntConc (Version 3.4.3) [Computer Software]. Tokyo, Japan: Waseda University. Available from: `http://www.laurenceanthony.net/`, 2014.

[4] Apache Software Foundation. Apache Lucene (Version 4.10.4) [information retrieval software library]. License: Apache License 2.0. Available from: `http://lucene.apache.org`, 2015.

[5] Apache Software Foundation. Tomcat (Version 8.0.23) [servlet container HTTP web server]. License: Apache License 2.0. Available from: `http://tomcat.apache.org/`, 2015.

[6] Atlas.ti. ATLAS.ti - version 7.0 [Computer Software] ATLAS.ti Scientific Software Development GmbH. http://atlasti.com/, 2012.

[7] Baker, P. *Using Corpora in Discourse Analysis*. Bloomsbury Discourse. Bloomsbury Academic, 2006.

[8] Banerjee, S., and Pedersen, T. The design, implementation, and use of the ngram statistics package. In *Proceedings of the 4th International Conference on Computational Linguistics and Intelligent Text Processing* (Berlin, Heidelberg, 2003), CICLing'03, Springer-Verlag, pp. 370–381.

[9] Baroni, M., Lenci, A., and Onnis, L. Isa meets lara: An incremental word space model for cognitively plausible simulations of semantic learning. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition* (Stroudsburg, PA, USA, 2007), CACLA '07, Association for Computational Linguistics, pp. 49–56.

[10] Belica, C., and Steyer, K. Korpusanalytische Zugänge zu sprachlichem Usus. *AUC (Acta Universitatis Carolinae), GERMANISTICA PRAGENSIA XX. Praha* (2006).

[11] Berber-Sardinha, T. Comparing corpora with wordsmith tools: How large must the reference corpus be? In *Proceedings of the Workshop on Comparing Corpora - Volume 9* (Stroudsburg, PA, USA, 2000), WCC '00, Association for Computational Linguistics, pp. 7–13.

[12] Berkovitz, J. An architectural blueprint for Flex applications. http://joeberkovitz.com/blog/reviewtube/, Dec. 2006. The original article (`http://www.adobe.com/devnet/flex/articles/blueprint.html`) got deleted by Adobe. Presentation at Max 2006 Conference: `http://joeberkovitz.com/max2006/RI304W_FlexBestPractices_JoeBerkovitz.ppt`.

[13] Biemann, C. TinyCC 2.0 User's Manual (Version 2.1.1.) [Computer Software]. University of Leipzig. Available from: http://wortschatz.uni-leipzig.de/cbiemann/software/TinyCC2.html, 2007.

[14] Bonnafous, S. Analyse de contenu. In *Dictionnaire d'analyse du discours*, P. Charaudeu and D. Maingueneau, Eds. Paris: E'd. du Seuil, 2002, pp. 39–41.

[15] Bordag, S. A comparison of co-occurrence and similarity measures as simulations of context. In *Proceedings of the 9th International Conference on Computational Linguistics and Intelligent Text Processing* (Berlin, Heidelberg, 2008), CICLing'08, Springer-Verlag, pp. 52–63.

[16] Brailich, A., Germes, M., Glasze, G., Pütz, R., and Schirmel, H. Die diskursive Konstitution von Großwohnsiedlungen in Frankreich, Deutschland und Polen. *Europa Regional 17* (2009).

[17] Brunet, E., and Luong, X. Computer processing and quantitative text analysis: Hyperbase, an interactive software for large corpora. In *Proceedings of the Conference on Data Analysis, Learning Symbolic and Numeric Knowledge* (Commack, NY, USA, 1989), Nova Science Publishers, Inc., pp. 207–214.

[18] Bullinaria, J. A., and Levy, J. P. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods* (2007), 510–526.

[19] Church, K. W., and Hanks, P. Word association norms, mutual information, and lexicography. *Comput. Linguist. 16*, 1 (Mar. 1990), 22–29.

[20] Cohen, T., Schvaneveldt, R., and Widdows, D. Reflective random indexing and indirect inference: A scalable method for discovery of implicit connections. *J. of Biomedical Informatics 43*, 2 (Apr. 2010), 240–256.

[21] Culy, C., and Lyding, V. Double tree: An advanced kwic visualization for expert users. In *Information Visualisation (IV), 2010 14th International Conference* (July 2010), pp. 98–103.

[22] Cyril, B. Kookkurrenzdatenbank CCDB. Eine korpuslinguistische Denk- und Experimentierplattform für die Erforschung und theoretische Begründung von systemisch-strukturellen Eigenschaften von Kohäsionsrelationen zwischen den Konstituenten des Sprachgebrauchs [Online web application] Institut für Deutsche Sprache, Mannheim ©2001 ff. Available from: `http://corpora.ids-mannheim.de/ccdb/`, 2001.

[23] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*, 6 (1990), 391–407.

[24] Defays, D. An efficient algorithm for a complete link method. *Comput. J. 20*, 4 (1977), 364–366.

[25] Diaz-Bone, R., Bührmann, A. D., Rodriguez, E. G., Schneider, W., Kendall, G., and Tirado, F. The field of foucaultian discourse analysis : structures, developments and perspectives. *Historical Social Research 33*, 1 (2008), 7–28.

[26] Dice, L. R. Measures of the Amount of Ecologic Association Between Species. *Ecology 26*, 3 (July 1945), 297–302.

[27] Dijkstra, E. W. On the role of scientific thought. In *Selected Writings on Computing: A Personal Perspective*. Springer-Verlag, 1982, pp. 60–66.

[28] Dockter, H., Murdoch, A., Faber, S., Niederwieser, P., Daley, L., Gröschke, R., DeBoer, D., and Appling, S. Gradle (Version 2.3) [build tool and command line environment]. License: Apache License 2.0. Available from: `http://gradle.org/`, 2015.

[29] Ducrot, O. *Dire et ne pas dire: principes de sémantique linguistique*. Collection Savoir. Hermann, 1972.

[30] Dumais, S. T., and Nielsen, J. Automating the assignment of submitted manuscripts to reviewers. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 1992), SIGIR '92, ACM, pp. 233–244.

[31] Dunning, T. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist. 19*, 1 (Mar. 1993), 61–74.

[32] Dzudzek, I. *Hegemonie kultureller Vielfalt - Eine Genealogie kultur-räumlicher Repräsentationen der UNESCO*. Forum Politische Geographie. Lit Verlag, 2013.

[33] Dzudzek, I., Glasze, G., Mattissek, A., and Schirmel, H. Verfahren der lexikometrischen Analyse von Textkorpora. In *Handbuch Diskurs und Raum. Theorien und Methoden für die Humangeographie sowie die sozial- und kulturwissenschaftliche Raumforschung.*, vol. 1. Bielefeld: Transcript-Verlag., 2009, pp. 233–260.

[34] Elastic. Elasticsearch groovy api (version 1.7.3) [elasticsearch api]. license: Apache license 2.0. Available from: `https://github.com/elastic/elasticsearch-groovy`, 2015.

[35] Elastic. Elasticsearch Java API (Version 1.7.x) [Elasticsearch API]. License: Apache License 2.0. Available from: `https://www.elastic.co/guide/en/elasticsearch/client/java-api/1.7/java-api.html`, 2015.

[36] Elastic. Elasticsearch (version 1.7.3) [text search engine]. license: Apache license 2.0. Available from: `https://www.elastic.co/products/elasticsearch`, 2015.

[37] Everitt, B., Landau, S., and Leese, M. *Cluster Analysis*. A Hodder Arnold Publication. Wiley, 2001.

[38] Evert, S. *The Statistics of Word Co-Occurrences: Word Pairs and Collocations*. PhD thesis, Stuttgart, Germany, 2005.

[39] Foucault, M. *Ordem do discurso (A)*. Edições Loyola, 1970.

[40] Foucault, M. *The archaeology of knowledge / Michel Foucault ;translated from the French by A.M. Sheridan Smith*. Tavistock Publications London, 1972.

[41] Francis, W. N., and Kucera, H. Brown corpus manual. Tech. rep., Department of Linguistics, Brown University, Providence, Rhode Island, US, 1979.

[42] Gabrilovich, E., and Markovitch, S. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence* (San Francisco, CA, USA, 2007), IJCAI'07, Morgan Kaufmann Publishers Inc., pp. 1606–1611.

[43] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software (Adobe Reader)*. Pearson Education, 1994.

[44] Gormley, C., and Tong, Z. *Elasticsearch: The Definitive Guide*. O'Reilly Media, Incorporated, 2015.

[45] Gritman, Jason. Httpbuilder (version 0.7.1) [easy http client for groovy]. license: Apache license 2.0. Available from: `https://github.com/jgritman/httpbuilder`, 2015.

[46] Hall, S. *Representation: Cultural Representations and Signifying Practices*. COMM1107:. SAGE Publications, 1997.

[47] Harris, Z. Distributional structure. *Word 10*, 23 (1954), 146–162.

[48] Helsloot, N., and Hak, T. Pecheux's contribution to discourse analysis. *Historical Social Research 33*, 1 (2008), 162–184.

[49] Hilpert, M., and Gries, S. T. Assessing frequency changes in multistage diachronic corpora: Applications for historical corpus linguistics and the study of language acquisition. *Lit Linguist Computing 24*, 4 (Dec. 2009), 385–401.

[50] Holtsberg, A., and Willners, C. Statistics for sentential co-occurrence, 2001.

[51] IDS. COSMAS I/II (Corpus Search, Management and Analysis System) [Online web application] Institut für Deutsche Sprache, Mannheim ©1991-2010. Available from: `https://cosmas2.ids-mannheim.de/cosmas2-web/`, 2010.

[52] Jones, K. S. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation 28* (1972), 11–21.

[53] Jones, M. N., Kintsch, W., and Mewhort, D. J. High-dimensional semantic space accounts of priming. *Journal of Memory and Language 55*, 4 (2006), 534 – 552. Special Issue on Memory Models.

[54] Jurgens, D., and Stevens, K. Event detection in blogs using temporal random indexing. In *Proceedings of the Workshop on Events in Emerging Text Types* (Stroudsburg, PA, USA, 2009), eETTs '09, Association for Computational Linguistics, pp. 9–16.

[55] Jurgens, D., and Stevens, K. Hermit: Flexible clustering for the semeval-2 wsi task. In *Proceedings of the 5th International Workshop on Semantic Evaluation* (Stroudsburg, PA, USA, 2010), SemEval '10, Association for Computational Linguistics, pp. 359–362.

[56] Kendall, M. G. A new measure of rank correlation. *Biometrika 30*, 1/2 (1938), 81–93.

[57] Kilgarriff, A. Why chi-square does not work and an improved LOB-Brown comparison . In *ALLC-ACH Conference* (1996).

[58] Kilgarriff, A. Comparing corpora. *International journal of corpus linguistics 6*, 1 (2001), 97–133.

[59] Kilgarriff, A., Rychly, P., Smrz, P., and Tugwell, D. The sketch engine. In *Proceedings of EURALEX* (2004).

[60] Klein, W., and Geyken, A. Das Digitale Wörterbuch der Deutschen Sprache (DWDS). 79–96.

[61] Krasner, G. E., and Pope, S. T. A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program. 1*, 3 (Aug. 1988), 26–49.

[62] Krippendorff, K. *Content Analysis: An Introduction to Its Methodology.* SAGE Publications, 2012.

[63] Kullback, S., and Leibler, R. A. On information and sufficiency. *Ann. Math. Statist. 22*, 1 (1951), 79–86.

[64] Kupietz, M., Belica, C., Keibel, H., and Witt, A. The German Reference Corpus DeReKo: A Primordial Sample for Linguistic Research. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)* (Valletta, Malta, may 2010), N. C. C. Chair), K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, Eds., European Language Resources Association (ELRA).

[65] Kupietz, M., and Lüngen, H. Recent developments in dereko. In *Proceedings of the Ninth conference on International Language Resources and Evaluation (LREC 14)* (2014), p. 2385.

[66] Laforge, G., Theodorou, J., King, P., and Champeau, C. Groovy (Version 2.4.3) [object oriented, imperative programming language]. License: License: Apache License 2.0. Available from: `http://groovy-lang.org/`, 2015.

[67] Lamalle, C., Martinez, W., Fleury, S., and Salem, A. Lexico3 - outils de statique textuelle [computer software]. université de la sorbonne nouvelle paris 3. Available from: `http://www.tal.univ-paris3.fr/lexico/lexico3.htm`, 2001.

[68] Landauer, T. K., and Dutnais, S. T. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *PSYCHOLOGICAL REVIEW 104*, 2 (1997), 211–240.

[69] Lebart, L., Salem, A., and Berry, L. *Exploring Textual Data (Text, Speech and Language Technology).* Springer, Dec. 1997.

[70] Leff, A., and Rayfield, J. T. Web-Application Development Using the Model/View/Controller Design Pattern. In *Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing* (Washington, DC, USA, 2001), EDOC '01, IEEE Computer Society, pp. 118–127.

[71] Lin, D. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 2* (Stroudsburg, PA, USA, 1998), COLING '98, Association for Computational Linguistics, pp. 768–774.

[72] Lund, K., and Burgess, C. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers 28*, 2 (1996), 203–208.

[73] Mani, I., and Pustejovsky, J. Temporal discourse models for narrative structure. In *Proceedings of the 2004 ACL Workshop on Discourse Annotation* (Stroudsburg, PA, USA, 2004), DiscAnnotation '04, Association for Computational Linguistics, pp. 57–64.

[74] Mann, W. C., and Thompson, S. A. Rhetorical Structure Theory: a theory of text organization. Tech. Rep. RS-87-190, USC/Information Sciences Institute, 1987. Reprint series.

[75] Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[76] Martin, E. *FRANTEXT, autour d'une base de données textuelles: témoignages d'utilisateurs et voies nouvelles*. Dictionnairique et lexicographie. Didier érudition, 1992.

[77] Martin, J. *Programming real-time computer systems*. Prentice-Hall series in automatic computation. Prentice-Hall, 1965.

[78] Mattissek, A. *Die neoliberale Stadt - diskursive Repräsentationen im Stadtmarketing deutscher Großstädte*. Urban studies. Bielefeld : transcript-Verl., 2008.

[79] McEnery, T., and Hardie, A. *Corpus Linguistics: Method, Theory and Practice*. Cambridge Textbooks in Linguistics. Cambridge University Press, 2011.

[80] Miller, J. Patterns in practice - convention over configuration. *Microsoft MSDN magazine 24*, 02 (2009). `http://https://msdn.microsoft.com/en-us/magazine/dd419655.aspx`.

[81] Niwa, Y., and Nitta, Y. Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In *Proceedings of the 15th Conference on Computational Linguistics - Volume 1* (Stroudsburg, PA, USA, 1994), COLING '94, Association for Computational Linguistics, pp. 304–309.

[82] Padó, S., and Lapata, M. Dependency-based construction of semantic space models. *Comput. Linguist. 33*, 2 (June 2007), 161–199.

[83] Pantel, P., and Lin, D. Discovering word senses from text. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2002), KDD '02, ACM, pp. 613–619.

[84] Pantel, P., and Lin, D. Document clustering with committees. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2002), SIGIR '02, ACM, pp. 199–206.

[85] Pearson, K. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine 50* (1900), 157–175.

[86] Pedersen, T. Ngram Statistics Package version v1.27 [Computer Software]. University of Minnesota, Duluth. Available from: `http://sourceforge.net/projects/ngram/`, 2013.

[87] Pedersen, T. SenseClusters version v1.03 [Computer Software]. University of Minnesota, Duluth. Available from: http://senseclusters.sourceforge.net/, 2013.

[88] Pedersen, T., Kayaalp, M., and Bruce, R. Significant lexical relationships. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96* (1996), pp. 455–460.

[89] Pereira, F., Tishby, N., and Lee, L. Distributional clustering of english words. In *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics* (Stroudsburg, PA, USA, 1993), ACL '93, Association for Computational Linguistics, pp. 183–190.

[90] Pivotal Software. Spring (Version 4.1.7) [application framework]. License: Apache License 2.0. Available from: http://spring.io/, 2015.

[91] Purandare, A., and Pedersen, T. Word Sense Discrimination by Clustering Contexts in Vector and Similarity Spaces, 2004.

[92] Pêcheux, M. *Analyse automatique du discours.* Sciences du comportement. Dunod, Paris, 1969.

[93] Rayson, P., and Garside, R. Comparing corpora using frequency profiling. In *Proceedings of the Workshop on Comparing Corpora - Volume 9* (Stroudsburg, PA, USA, 2000), WCC '00, Association for Computational Linguistics, pp. 1–6.

[94] Red Hat. Hibernate (Version 4.3.10) [object relational mapping]. License: GNU Lesser General Public License. Available from: https://http://hibernate.org/, 2015.

[95] Rijsbergen, C. J. V. *Information Retrieval*, 2nd ed. Butterworth-Heinemann, Newton, MA, USA, 1979.

[96] Rocher, G. Grails (Version 3.0.3) [web application framework]. License: Apache License 2.0. Available from: https://grails.org/, 2015.

[97] Rocher, G., Ledbrook, P., Palmer, M., Brown, Jeff, D. L., Beckwith, B., and Hotari, L. The Grails Framework - Reference Documentation (Version 3.0.3) License: License: Apache License 2.0. Available from: http://grails.github.io/grails-doc/3.0.3/guide/, 2015.

[98] Rohde, D. L. T., Gonnerman, L. M., and Plaut, D. C. An improved model of semantic similarity based on lexical co-occurence. *Communications of the ACM 8* (2006), 627–633.

[99] Rorty, R. Wittgenstein, Heidegger, and the reification of language. In *Essays on Heidegger and others*, vol. 2. Cambridge University Press, 1991, pp. 50–65. Cambridge Books Online.

[100] Sahlgren, M. An introduction to random indexing. In *In Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005* (2005).

[101] Sahlgren, M. *The Word-Space Model: using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces.* PhD thesis, Stockholm University, 2006.

[102] Sahlgren, M., Holst, A., and Kanerva, P. Permutations as a means to encode order in word space. In *Proceedings of the 30th Annual Conference of the Cognitive Science Society*, V. Sloutsky, B. Love, and K. Mcrae, Eds. Cognitive Science Society, Austin, TX, 2008, pp. 1300–1305.

[103] Salton, G. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[104] Salton, G. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Computer Science Series. Addison-Wesley, 1989.

[105] Salton, G., and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing & Management 24*, 5 (1988), 513 – 523.

[106] Salton, G., Wong, A., and Yang, C. S. A vector space model for automatic indexing. *Commun. ACM 18*, 11 (Nov. 1975), 613–620.

[107] Sasirekha, K., and Baby, P. Agglomerative hierarchical clustering algorithm - a review. *International Journal of Scientific and Research Publications (IJSRP) 3*, 301 (2013), 1–3.

[108] Saussure, F. D., Bally, C., Sechehaye, A., and Riedlinger, A. *Cours de linguistique générale*. Lausanne ; Paris : Payot, 1916.

[109] Schirmel, H. *Sedimentierte Unsicherheitsdiskurse: die diskursive Konstitution von Berliner Großwohnsiedlungen als unsichere Orte und Ziel von Sicherheitspolitiken : mit 2 Tabellen und 23 Kontextkästen*. Erlanger geographische Arbeiten / Sonderband. Selbstverl. der Fränkischen Geograph. Ges., 2011.

[110] Schütze, H. Automatic word sense discrimination. *Comput. Linguist. 24*, 1 (Mar. 1998), 97–123.

[111] Schütze, H., and Pedersen, J. A vector model for syntagmatic and paradigmatic relatedness. In *Making sense of words*. Oxford, England: Ninth Annual Conference of the UW Centre for the New OED and Text Research, 1993, pp. 104–113.

[112] Scott, M. Wordsmith tools version 6 [computer software]. liverpool: Lexical analysis software. Available from: `http://www.lexically.net/wordsmith/`, 2012.

[113] Sibson, R. SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal 16*, 1 (Jan. 1973), 30–34.

[114] Singhal, A., Salton, G., Mitra, M., and Buckley, C. Document length normalization. *Inf. Process. Manage. 32*, 5 (Sept. 1996), 619–633.

[115] Surhone, L., Tennoe, M., and Henssonow, S. *Russian National Corpus*. Betascript Publishing, 2010.

[116] Tognini-Bonelli, E. *Corpus Linguistics at Work*. Studies in corpus linguistics. J. Benjamins, 2001.

[117] Turney, P. D., and Pantel, P. From frequency to meaning: Vector space models of semantics. *CoRR abs/1003.1141* (2010).

[118] VERBI. MAXQDA - The Art of Textanalysis version 11 [Computer Software] Marburg, Germany: VERBI Software. http://www.maxqda.com/, 2012.

[119] Wattenberg, M., and Viégas, A. B. The word tree, an interactive visual concordance. *IEEE Transactions on Visualization and Computer Graphics* (2008), 1221–1228.

[120] Webber, B. L. Discourse deixis: Reference to discourse segments. In *ACL* (1988), ACL, pp. 113–122.

[121] Wieser, C. Lexikometrische Diskursanalyse als Methode der Kritischen Geopolitik am Beispiel des Arabischen Frühlings. *Global Studies Working Papers, Institute of Geography, Universität Tübingen* (2012).

[122] Wright, J. Deconstructing development theory: Feminism, the public/private dichotomy and the mexican maquiladoras*. *Canadian Review of Sociology/Revue canadienne de sociologie 34*, 1 (1997), 71–91.

[123] Yates, F. Contingency tables involving small numbers and the χ^2 test. *Journal of the Royal Statistical Society B1* (1934), 217–235.

[124] Étienne Brunet. HyperBase version 10 [Computer Software]. l'Université Nice Sophia Antipolis. Available from: `http://ancilla.unice.fr/?redirected_from=ancilla.unice.fr/~brunet/pub/hyperbase.html`, 2015.