UNIVERSITÄT PASSAU
Fakultät für Informatik und Mathematik

dragn: A Distant Reading System using Distributional Semantics

Thomas Huber

Matriculation number: 67287

BACHELOR THESIS

Chair of Digital Libraries and Web Information Systems

University of Passau

In partial fulfillment of the requirements for the degree of

Bachelor of Science (B.Sc.) in Computer Science

Supervisor: Professor Dr. Siegfried Handschuh

Abstract

Close Reading is the process of analysing a text in-depth. The goal is to get as much information as possible in general or to focus on a specific aspect or question. Franco Moretti, an Italian literary scholar, coined the term "Distant Reading". The goal of Distant Reading is to gain understanding of texts by skimming the contents to gain a more general overview of the content. Due to the nature of Distant Reading, it is possible to process more content, faster, than would be possible with Close Reading. To do so by hand without the assistance of a tool is obviously very laborious and difficult. Machine assisted work in this area can be much more efficient.
This thesis is built and improved upon an existing system, dubbed Skimmr, developed by Vit Novacek. The goal of the tool itself is to allow humans to effectively and efficiently perform Distant Reading on collections of texts or even just single texts. To accomplish this task, state-of-the-art technologies and frameworks are being used to improve the quality of the original system.
The new system, named "dragn", utilises a pipeline of four processing steps to create an information structure that users can query. An arbitrary number of texts can be used and processed by the system. First the texts are parsed into paragraphs, Noun Phrases (NP) extracted and an inverse index of sentences is built. Using this index, a modified pointwise mutual information (PMI) value is calculated over the corpus. The modified PMI used in this system takes into account the frequency of the two tokens co-occurring and a calculated weighted distance between the two. This causes the score to be increased the more often two tokens appear close to each other in a text. Performing the score calculation this way improves the quality of results for the Distant Reading. After building a vector space over those PMI-like values, relevant tokens are computed for each of the non-stopword tokens of the texts using Cosine Similarity. Using the computed data, files are written to the disk to make it easier to perform queries on the data and to allow the data being used in different ways, such as a different front end.
Users can query the texts to find text passages calculated to be relevant based on the modified PMI value and Cosine Similarity relations contained therein and see how different words in the texts are connected through the same metrics. Distant Reading becomes possible by having access to those select paragraphs and being able to read previous and following passages from the text.

Acknowledgment

I would like to thank Prof. Dr. Siegfried Handschuh for suggesting this topic to me and supervising my thesis. There have been obstacles on the way but his support helped me continue working on this project and ultimately finish it.
Further I want to thank Adamantios Koumpis for getting me involved with the Chair and suggesting to me to work there. Without him I would not be where I am today.
I want to think all team members, past and present of the Chair of Digital Libraries and Web Information Systems who have been hugely inspirational to me either directly or indirectly through their work.
Finally I want to thank Karin Nowotny, Felix Bernasch and Michael Maier for proofreading the thesis and providing feedback.

Thomas Huber

Table of Contents

List of Figures

List of Tables

Table of Code Snippets

Chapter 1

Introduction

Digital texts have and are still becoming increasingly more common. Whereas in the past the majority of texts were physical, with the ongoing digitisation of literature but also the inclusion of a digital version for new works becoming standard there now exists a great wealth of corpora usable for machine processing, and the number will increase only further. This of course opens up new possibilities for the Computer Science and (Digital) Humanities research. However, dealing with a large number of texts by hand can be difficult bordering on the impossible. Even analysing a single book by hand takes up a significant amount of time. So one has to ask: *How can one process the information contained in text collections large and small in a reasonable way?* Italian literary scholar Franco Moretti suggests an approach named "Distant Reading" in [**Mor16**]. As opposed to Close Reading, processing and extracting information from single texts in detail, the idea behind Distant Reading is to not handle each text individually, but a collection of texts as a whole. The content is aggregated and analysed not in detail, but rather more generally. The insight gained from Distant Reading is less detailed as opposed to Close Reading, however this allows the processing of multiple texts in the time it would take to read a single text closely. Distant Reading cannot replace the need to read texts to understand them fully. Instead, it lets the reader gain a superficial overview of a collection of texts or a subset thereof and use that understanding to preemptively filter out certain works that are not relevant to answering a certain (research) question.

1.1 Motivation

Distant Reading makes it possible to process and analyse large amounts of text or rather data. The technique allows one to gain insight into texts that would otherwise be very laborious to obtain. To further support the user, a visualisation of the results is desirable. In [**Mor13**] (p. 214-218) the author describes his procedure to plot a network of interactions between the characters in *Hamlet*. By having access to a network of connected characters or words in general in a text one can quickly gain understanding of the therein contained information and preemptively increase the understanding of the context by being able to see relations between words in the text. Finding relations between words or phrases in text however is complicated. One has to define how such a relation is built or formed and then extract them from, generally a large, corpus. The thesis builds upon the work of Vit Novacek [**NB14**], who finds relations by the Pointwise Mutual Information (PMI) and the Cosine Similarity between words. His system can be used to find links (from here on called *relations*) between words in a text.

My system builds upon his existing work to create a new system named **dragn** that allows users to more efficiently find relations in texts and to understand the nature of the relation better. The original system shows the relations but not the weight (how closely related the words are as a numeric value). Further it does not allow easy access to text passages relevant to the words that were queried for. The greatest limitation is the user interface, which does not let the user understand clearly and easily how or why a link between two words exists. To improve the quality of the output a new front end has been developed and the code of the original system been updated to be up to the state-of-the-art. My system keeps the information about corpora it has processed and lets the user select the corpus they want to work on.

Distributional semantics techniques are used to process the text into formats that can be employed by the front end and the results displayed to the user. As this thesis builds upon an existing work, it is not a new approach. The main contribution of this thesis to the usage of distributional semantics in a Distant Reading context is the modernisation of the existing system and improving the usability of the tool, especially for users from a non-technical background, to support and improve research in the field of literary sciences.

The basis of the system is the implementation of a pipeline of four steps, each with readable and

usable output. Each step produces independent files that allow users from a technical background to work with the result of the individual steps.

A modified PMI score sets the basis for the system. The PMI is a measure to describe how strongly correlated two events are, that is, to say whether they co-occur more often or less often than they would by chance. The PMI is calculated for each pair of tokens in all sentences of the corpus. High PMI scores demonstrate that the tokens appear more frequently together and thus a relation between them can be assumed. Generally speaking it measures the statistical (in)dependence of two events. For this thesis the PMI is used to measure the statistical independence of two text fragments (tokens): The higher the score, the higher the dependence and thus the more related they are. The lower the score, the lower the relation between the two; a score of zero indicates independence while a negative score indicates that two tokens co-occur less frequently than would be expected.

The PMI is further used to calculate tokens that are related to another one without having to co-occur directly.

As a simple example: *apple* appears often near *tree*, *leaf* appears often near *tree*. *Apple* and *tree* would have a high PMI and so would *leaf* and *tree*, therefore a relation between *apple* and *leaf* also exists, even though their PMI might be low.

Finding such relations or calculating the PMI can be a monumental task for even just a single, albeit medium- to large-sized, text. Using **dragn**, the system developed for this thesis, makes that easy. Additionally the system allows the user to find links between tokens that would be unexpected thanks to the graphical interface.

dragn supports tasks in the literary sciences and Humanities by helping find connections between words of not just single texts, but arbitrarily sized collections.

1.2 Objectives

The main objective of this Bachelor's project is to show how Distant Reading can be performed using a machine context and how this task can accomplished more effectively and efficiently when using a tool as opposed to doing so by hand. To achieve the objective, the existing back end of the *skimmr* system developed by [**NB14**] was used as a foundation and improved upon.

Skimmr is a tool that accomplishes the aforementioned tasks. My contributions are a rebuilt system using modern technologies, an analysis and explanation of how the system works and the development of a new interface to increase usability of the system for the intended task and expand the suitability of the original system in regards to the context of Distant Reading.
Ultimately, the system will provide the following functionality:

- Calculating the PMI of tokens in arbitrarily sized collections of texts.
- Calculating tokens related to other tokens, using Cosine Similarity.
- Easy querying of processed texts.
- Exploration of links in the graphical output of user queries.
- Context reading by having easy access to following and previous text passages found as a result of a query.

The system developed for this thesis is not intended to be a finalised, perfected version, but rather a solid foundation for further, more advanced research using **dragn**.
An example use case scenario will be outlined as part of this thesis. The bible will be processed by the system, intuitive queries performed on the result and interesting links found by the system will be explored.

1.3 Approach

To accomplish the main objectives of this thesis, the original system has been rebuilt and the structure of the system will be outlined before providing proof-of-concept by examining a use case.

1.4 Structure of the Report

The rest of the report is structured as follows. Chapter 2 gives a generalised overview of the system's steps followed by detailed explanation in the sub chapters. The third chapter contains an example use case to demonstrate the capabilities of the system and how it can be used as a

foundation for research in the Humanities and more specifically literary sciences. In the final chapter a summary of contributions and possible further works are highlighted.

Chapter 2

Design and implementation of dragn

The architecture of the **dragn** (**D**istant **R**eading **A**nd **G**raph **N**odes) system and its design will be outlined in this chapter. Furthermore, each of the pipeline steps of the system will be detailed and the algorithms and calculations performed therein explained. Each of the steps produces output that is written to the disk and persists even after that part of the system finishes for further analysis and use in other research.

The emphasis of this chapter is not only on the internal part of the system, the back end, but the interface or front end as well.

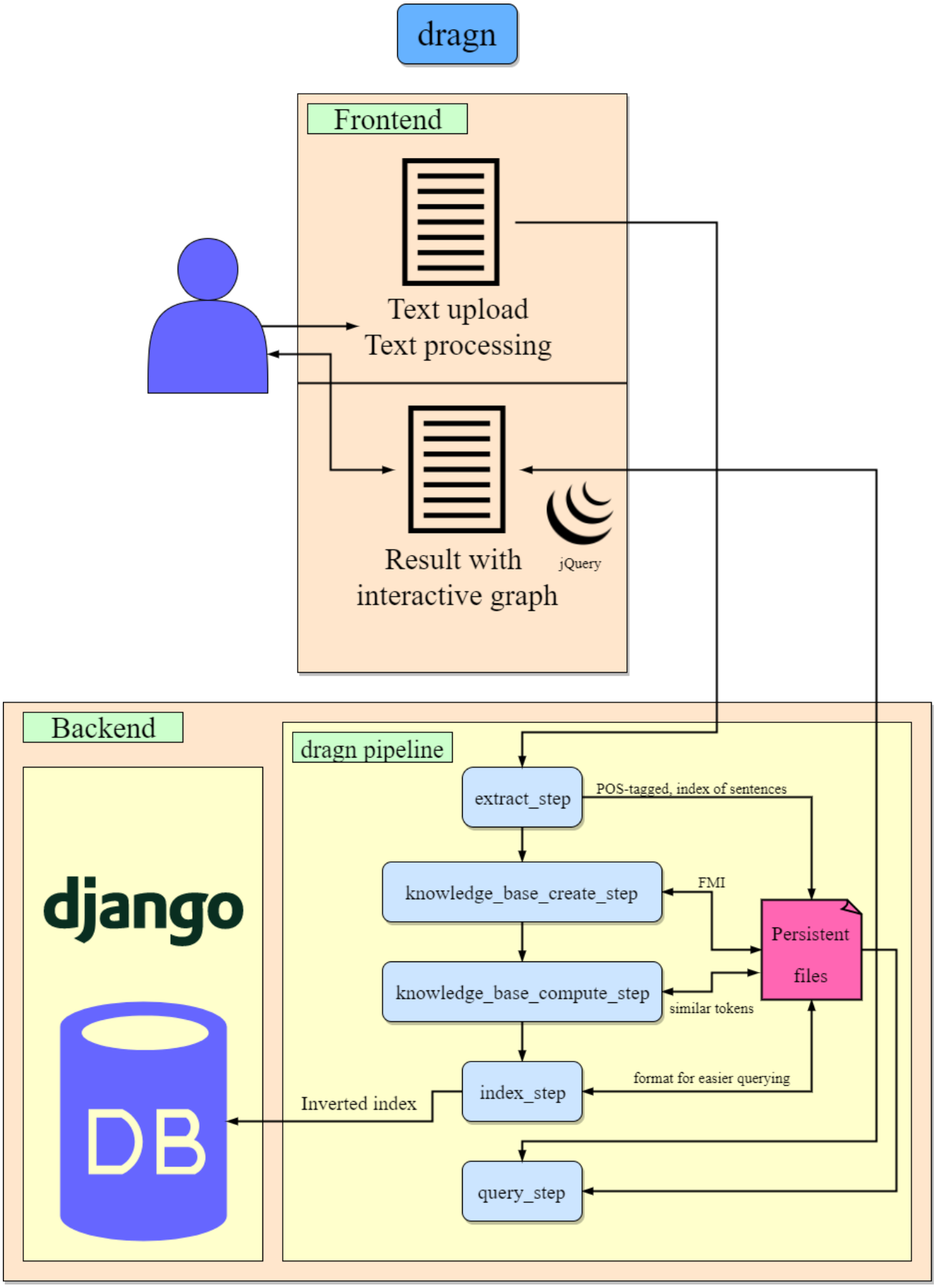

Figure 2.1: Overview of "dragn" system architecture

2.1 Architecture Overview

The original system, Skimmr [**NB14**] is written in Python 2.7, whereas **dragn** is written in Python 3.5. The main "support" framework of the system is Django 1.10.6, which is mostly used to lessen the burden on the user when setting up the system on a network and to easily provide an interactive interface to the user. The system was built with the deployment on a (research) network in mind to allow multiple people to use it for work at once.
For the interaction part specifically, another framework is used: *Cytoscape.js* [**FLH+16**]. It allows the display of graphs from JSON (Javascript Object Notation) data and supports exporting the result graphs as JSON as well, allowing users to use the results more easily in complex ways.
The core of the system is built upon four steps (five if including the queries):

- extract_step - Extracts paragraphs and tokens from texts, applies POS-tags and builds an inverted index of sentences.
- knowledge_base_create_step - Calculates a modified PMI, the FMI, (for more details, see chapter 2.3.1) over all tokens in the texts.
- knowledge_base_compute_step - Calculates Cosine Similarity over tokens from the texts using the FMI values.
- index_step - Brings the results of the previous two steps in a more easily accessible format.
- query_step - Performs the user's queries on the data and produces a graph in JSON format for use with *Cytoscape.js*.

Texts are uploaded into the system through the web interface, afterwards the texts to process are selected. For this, the *Alias System* is used: A combination of texts is processed together and this combination of texts can be selected from a dropdown when performing a query. The name in the dropdown is an alias for the bundled texts, hence the name, and this bundling persists. An arbitrary number of aliases can exist at once, allowing the user to select which combination of texts to query.
While the texts themself are stored directly on the disk, the aliases and their corresponding texts are stored in the Django database.

Overall, the system follows the MVC (Model-View-Controller) pattern, with the HTML front end and Django embodying the View and Controller respectively, while the system's back end with the _step pipeline is the Model part of the pattern.
MVC is an important pattern used commonly in Software Engineering and using it gives the system greater flexibility regarding the display of the results and data, as an integral part of the MVC architecture is to have the user interface be completely replaceable without having to modify the core of the system.

2.1.1 Use of Django

Django is used because of its widespread use and ease of setup on different systems. More specifically, the HTML pages are served by Django and its internal templating engine. The database integration of Django is used for the Alias System and for the inverted index.

2.1.2 Inverted Index

An inverted index is the mapping of text fragments, generally processed words in a text, to a list of texts they appear in. As an example: For this inverted index, *apple* appears in documents

Table 2.1: Example inverted index

Token	Documents
apple	story.txt, story2.txt, story2.txt
tree	story2.txt, story3.txt, story49.txt

story.txt, *story2.txt* and *story2.txt7*.
Data structured in this way allows easy and quick access to texts containing certain words.
In **dragn**, such an index is used to bolden all tokens appearing in paragraphs relevant to the query that also appear in the result graph. For more details, see chapter 2.7.3.

2.1.3 Alias System

dragn works by passing it collections of texts which the system then processes into files that are used to satisfy the user's queries. Due to the nature of the system, it is very likely that the user might want to analyse permutations of their corpus. To support that need, the *Alias System* was

implemented.

Before the user can work with the system, the texts that are going to be the base for the queries have to be uploaded to the system and selected. The selected combination of texts is assigned an *Alias* in the system and an entry in the database is added to map the Alias to the names of the texts. This is then used by the system to identify which files belong to which combination of texts by having the files written to folders named after the Alias.

When performing a query the user can then select the Alias from a list and query only the texts belonging to that Alias. The Aliases and the files created during the processing of the texts persist between queries and thus the user can at any time decide to work with a specific subset of their texts by selecting the corresponding Alias and performing a query.

By default, the name of the Alias is a combination of the file names to lessen the burden of knowledge on the user, but any other presentation is possible.

An example:

Table 2.2: Example of the Alias System

Alias	Documents
story.txt,story2.txt	story.txt, story2.txt
story.txt	story.txt

In this example based on table 2.2, the user can select "story.txt" as the Alias and perform queries on the processed files of the document "story.txt" without having to make the system process the text beforehand, because that has already been done.

The Alias System is especially advantageous when multiple users are working with the system. Previously processed permutations of texts created by other users are selectable as an Alias and can immediately be queried.

Due the separation of processed files into different sub-directories as per their Alias the individual permutations queries on a specific Alias experience no slowdown from the other Aliases, as would be the case without a separation into multiple files. In that case, either a single massive file would have to be loaded and the relevant upshot of the previous processing of files be filtered out to get only the content relevant to the current texts or complex database queries doing the same would need to be performed.

The core of the system is to find information like "apple is connected to tree", but to also allow

the user to understand why that connection has been found and where this connection becomes apparent in texts. However, the user might be interested to see the connections of a certain word to other words in a single or multiple specific texts and compare them. "Apple" might appear often near "tree" in a botanical text and have a high PMI, but in a different text their PMI might be low.

In **dragn** this problem is accounted for by producing multiple files containing such information and having those files be mapped to an Alias of either a single text or multiple texts together.

Without the Alias System the files of a previous processing would either have to be deleted and thus the texts would have to be processed again later, or a very complex system would have to be built to contain and extract the information for multiple texts from the files.

The Alias System circumvents this problem and is an integral part of the system's usability.

2.2 Extraction of Noun Phrases and calculation of distance

The following chapters are intended to give a detailed insight into the different parts of the system, how they work individually and how their outputs are used in the consecutive steps.

This chapter aims to provide the reader with a comprehensive understanding of the first step in the **dragn** pipeline, *extract_step*.

As briefly touched upon in the previous chapter 2.1.3, each of the four main steps of the **dragn** pipeline produces persistent files that are contained in folders named after the Alias of the texts. The folders for those files are created as part of this first step and the uploaded texts are moved into the corresponding directory.

The texts are then processed one by one. Initially, the paragraphs are extracted from the texts. They are part of the result of the queries, paragraphs containing words relevant to the query (see 2.6). The system operates under the assumption that paragraphs in a text are logical, self-contained units and as such the system works on a per-paragraph basis. A clarification with an example will follow shortly.

Next is the POS-tagging of the sentences in the current paragraph. POS stands for "Part Of Speech" and tagging words in such a way means to assign labels such as "Noun", "Proper Noun", "Adjective". The tagset used is the Penn Treebank [**MMS93**]. Grammatical inconsistencies may

exist in text, especially if it is not a properly published work where editors are proofreading, but generally it can be assumed that the grammar is correct enough to properly label the parts of the sentences.

The POS-tags are useful because they allow for the extraction of Noun Phrases from the sentences. For this extraction, grammar from the original system developed by [**NB14**] is used. This grammar, combined with parsers from NLTK (Natural Language Toolkit) is used by the parser to find Noun Phrases. Noun Phrases are concatenated words in a text that describe a single entity, such as *yellow dog*, as described in [**Abn87**].

Two grammars for the English language are currently in the system, developed by [**NB14**].

```
NP_GRAMMAR_COMPOUND = """
NP: {
        <JJ.*>*(<N.*>|<JJ.*>)+
        (
            (<IN>|<TO>)?<JJ.*>*(<N.*>|<JJ.*>)+
        )*
        (
            (<CC>|,)<JJ.*>*(<N.*>|<JJ.*>)+
            (
                (<IN>|<TO>)?<JJ.*>*(<N.*>|<JJ.*>)+
            )*
        )*
    }
"""
NP_GRAMMAR_SIMPLE = """
NP: {<JJ.*>*(<N.*>|<JJ.*>)+}
"""
```

Listing 2.1: Noun Phrase grammars

Note that *NP_GRAMMAR_COMPOUND* has been split into multiple lines for easier readability in this thesis at the cost of not being valid for use in Python anymore. The correct grammar

is a single line which makes it harder on the reader to understand what is happening and is of course still included as such in the source code of the system.
The purpose of the grammars is to extract Noun Phrases from the texts.
First the POS-tagged sentences for each paragraph of a text are chunked using the *NP_GRAMMAR_COMPOUND*. An explanation with an example of NP-chunking and the role of grammars will now be discussed to help the reader understand the following parts.

NP-chunking and grammars

Generally speaking the goal of NP-chunking is to find Noun Phrases in texts. Consider the following sentence:

The/DT little/JJ yellow/JJ fruit/NN outside/IN Passau/NNP.

The letters after each word are the word's corresponding POS-tag.
For this sentence it makes sense to consider *little yellow fruit* as a Noun Phrase. *fruit* is clearly a noun and *yellow* and *little* refer to the fruit. To combine those words into a single Noun Phrase we use the grammar from above.
In the first line we have

<JJ.*>*(<N.*>|<JJ.*>)+

which can be read as:

- zero or more adjectives (the Penn Treebank POS-tags for adjectives all begin with JJ, hence the JJ.*, see [**Tre**])
- ... followed by either one or more (the + means one or more, it applies to whatever is inside the brackets):
  - a noun (the POS-tags for nouns all start with N, hence the N.*)
  - another adjective

Therefore, by reading the tags in the example sentence given, we see that *little yellow fruit* follows the rules and thus would be a Noun Phrase according to the grammar.
As we read the next line, we have the continuation of the grammar:

- zero or more of (the asterisk at the end of round bracket) the following construct:
  - exactly one of either (the question mark means zero or one times)
    - preposition or subordinating conjunction
    - the word "to"
  - followed by any number of adjectives
  - followed by one or more of either:
    - a noun
    - an adjective

We continue to read the sentence and see that *outside Passau* satisfies this next line of the grammar. Therefore *little yellow fruit outside Passau* is a Noun Phrase we extracted from the sentence using the compound grammar.
The two lines that were omitted for this example work in exactly the same way, the Noun Phrases we can extract would simply be longer and more complex.

Usage of the extracted Noun Phrases in dragn and distance calculation

In **dragn** the grammars are used to extract the Noun Phrases from each sentence. As the system processes each paragraph for each text individually, we have access to the sentences and can use them to map the extracted Noun Phrases to the sentences they appear in to create what is essentially an inverted positional index. As Noun Phrases can be made up of multiple words, they allow the user to more finegrainedly perform queries, instead of simply searching for "fruit" it becomes possible to search for "red fruit" as an example. Extracting Noun Phrases makes it possible to search for relations only between Noun Phrases and find more specific relations, such as "apple IS A red fruit". For more details, refer to the suggestions for further work in chapter 4.2.
As the overall goal is to find tokens in text that are related to each other in some way, it is a good idea to filter out combinations that are very unlikely to be linked. Consider the simple Noun

Table 2.3: Inverted positional index

Lexical unit (for example Noun Phrases or just single verbs)	Sentence position in paragraph
little yellow fruit	0, 2, 3
apple	0
rocket	20

Phrases "apple", which appears in the first sentence, and "rocket", which occurs twenty sentences later (see table 2.3). A semantic link between those two words is very unlikely due to their distance in sentences. For that reason, a weighted distance score is calculated.

The general idea behind this is that words that are very far apart are unlikely to be related. Therefore, we use the positional index that was created while extracting the Noun Phrases to filter out pairs that cannot be related.

Calculating the score is simple. For each combination of lexical units from the positional index, we look at all the positions of the two where the distance is below five sentences. If the distance is too big, for example ten sentences, they are very unlikely to be related. We continue with the calculation of their weighted distance, which is $w+ = 1/(1 + distance$. The further apart they are, the lower the update to their weighted distance. If their weighted distance is above the threshold of 1/3 the pair is added to a list for further processing in the next step, where the PMI between the two is being calculated.

For the data in table 2.3 we would thus get the following result for *apple* and *little yellow fruit*:

$$\begin{aligned} w &= \frac{1}{1+0} = 1 \text{ (positions 0 and 0)} \\ w+ &= \frac{1}{1+2} = 1 + 0.333 = 1.333 \text{ (positions 2 and 0)} \\ w+ &= \frac{1}{1+3} = 1.333 + 0.25 = 1.583 \text{ (positions 3 and 0)} \end{aligned} \tag{2.1}$$

It is easy to see that the higher the distance, the lower the change to the weight becomes. Important to note here is that tokens which appear consistently multiple sentences apart would have a score higher than the threshold and be considered. In that case, a relation between the two might be possible and the system keeps them to have their PMI calculated.

2.3 Calculation of modified PMI between words - FMI

In this step the modified PMI is calculated over the units from the texts. The basis for that are the extracted <*token*> *close to* <*token2*> triples calculated as per chapter 2.2, from here on referred to as SPO- or *Subject-Predicate-Object* triples as per the naming conventions in the original system developed by [**NB14**]. If their textual origin, the *provenance* is included, they will be referred to as *SPOP-tuple*.

2.3.1 FMI - PMI with emphasis on the frequency and weighted distance

As mentioned before, the system does not use the classic Pointwise Mutual Information or PMI but a modified one as developed by Vit Novacek. In this chapter, first that modified PMI is explained and then the differences to the classic PMI highlighted.
An approach similar to the one in this thesis is used in [**RJ09**] to compare PMI and LSA [**Lan06**]. The approach discussed therein uses a model based on word counts whereas in this thesis the distance between two words is used. In [**CH90**], the PMI is defined as:

$$\mathrm{PMI}(x;y) \equiv \log_2\left(\frac{p(x,y)}{p(x)\,p(y)}\right) \tag{2.2}$$

It is used to measure statistical independence between two outcomes x and y. A positive PMI means the two outcomes co-occur more often than would be expected by chance, so therefore a connection between the two can be assumed, and a negative PMI means they co-occur less often than would be expected. A score of zero means they are independent [**Bou09**].
We take the probability of the two outcomes co-occurring and divide it by the probability of them simply occurring by chance (refer to [**CH90**], p. 2, chapter 4).
As a simple example, *apple* and *tree* would have a high PMI in a text about an orchard. For the sake of this example, assume they are dependant on each other. The occurrence of the fruit is closely tied to the word *tree*.
The PMI is a useful measure because we can calculate how dependant the appearance of a

phrase or word in a text is on the appearance of another phrase or word. A modified PMI that takes into account the average weighted distance calculated as in 2.1 and emphasises the frequency more strongly is used in this system. The original concept behind this was developed and defined by Vit Novacek [**NB14**].

$$\text{FMI}(p1;p2) \equiv joint(p1,p2) * \log_2\left(\frac{\#relations * (joint(p1,p2)+joint(p2,p1)}{\#p1 * \#p2}\right) * \frac{\sum distances}{\#distances} \tag{2.3}$$

$$\text{With joint}(p1,p2) \equiv \text{Number of relations with "p1 close to p2"} \tag{2.4}$$

First the equivalence between the middle part (the log) and the PMI formula will be shown. In the context of this system we look at the relations that were calculated in the first step 2.2 and check in how many of them the phrases or words we are calculating the PMI for appear.

$$\text{PMI}(p1;p2) = \log_2\left(\frac{\frac{y}{x}}{\frac{z}{x} * \frac{w}{x}}\right)$$

Where

y = number of relations with both p1 and p2

x = number of relations

z = number of relations with p1

w = number of relations with p2

$p1/p2$ = words or phrases from a text

By inserting those into the PMI formula we get:

$$\begin{aligned}
\mathrm{PMI}(p1;p2) &= \log_2\left(\frac{\frac{y}{x}}{\frac{z}{x}*\frac{w}{x}}\right) = \\
&= \log_2\left(\frac{\frac{y}{x}}{\frac{zw}{x^2}}\right) = \\
&= \log_2\left(\frac{yx^2}{xzw}\right) = \\
&= \log_2\left(\frac{xy}{zw}\right) \\
\mathrm{FMI}(p1;p2) &= \log_2\left(\frac{\#relations*(joint(p1,p2)+joint(p2,p1)}{\#p1*\#p2}\right) = \\
&= \log_2\left(\frac{x*y}{z*w}\right) = \\
&= \mathrm{PMI}(p1;p2)
\end{aligned}$$

Therefore

$$\mathrm{FMI}(p1;p2) = joint(p1,p2)*\mathrm{PMI}(p1;p2)*\frac{\sum distances}{\#distances}$$

We thus see the FMI formula is just the PMI formula but with the average weighted distance and the frequency added.

An example will now be calculated to show how the formula works: Consider the following data:

Table 2.4: Example result of extract_step

Extracted phrase / token	Phrase / Token that will be checked	Weighted distance	Provenance
apple	roof	1.2	story.txt_1
tree	apple	0.8	story.txt_1
apple	tree	0.6	story.txt_2
tree	leaf	1.3	story.txt_2
tree	roof	0.7	story.txt_1

The first line in table 2.4 means "apple" has a weighted distance (2.1) of 1.2 to "roof" in *story.txt_1*, which itself means "the first paragraph of the file story.txt".

We will use "tree" and "leaf" as the example for the calculation.

$$\begin{aligned} \text{PMI}(tree;leaf) &= \log_2\left(\frac{\frac{1}{5}}{\frac{4}{5}*\frac{1}{5}}\right) = 0.3219 \\ \text{FMI}(tree;leaf) &= 1*0.3219*\frac{1.3}{1} = 0.70 \end{aligned} \tag{2.5}$$

joint(p1;p2) is always positive, if there is no co-occurrence then calculating the PMI or FMI makes no sense. The same applies for a negative joint value, in the context of co-occurrences in a text this can never happen.

Now consider the average weighted distance. This value too can never be negative or zero. It can be lower than 1, in which case the FMI would be lower than the PMI value. This would be the case if the weighted distance between "tree" and "leaf" was not 1.3 but 0.8 for example. A low weighted distance means the distance between the two words or phrases is high.

We can therefore see that the modified PMI value, the FMI, rewards words co-occurring in different paragraphs, but consistently. The more paragraphs they co-occur in, the higher the joint frequency multiplier. On the other hand, it punishes high distance in the co-occurrences, the higher the distance the lower the weighted distance and the lower the average weighted distance multiplier.

This modified FMI is useful precisely because of those properties. The PMI does not consider the distance at all, it uses a binary measure to count how often two events co-occur (the events here being words or phrases co-occurring in a text). This is not good enough. Words that appear very close together, very often, are likely more related to each than words that appear together, but far apart. In the worst case two words co-occur in many different paragraphs consistently, but far apart. With the normal PMI measure they would have the same score as a pair of words that appears directly after each other in the same paragraphs. However one would expect that words that appear consistently close to each other are more related to each other than those consistently far apart from each other. The FMI formula discussed in this chapter scores the closely co-occurring words higher while still keeping the expressiveness, whether two events occur more frequently together than apart, in this context appearing together in a text, of the PMI formula and is thus a good measure to use in the context of Distant Reading. We can query for words that we know occur in a text from the context or title and find other words or phrases

that appear near it and thus gain additional information about the text with minimal effort.

2.3.2 Normalisation of the FMI values

The extracted weighted distances as described in chapter 2.2 are used to calculate the FMI values as described in the previous chapter 2.3.1. The result is a dictionary mapping the SPO-triples to their FMI score.
The calculated FMI values are then normalised. For that, the SPO-triples are sorted by their FMI value. The lowest value of the 5% of top FMI scores is then used as the normalisation value. The values are all normalised to a value between a minimum value, default being 0.1, and 1 as the upper limit. The normalisation performed is a simple division of the FMI value by the normalisation value.
This procedure results in lower normalised FMI values for the bottom 95% of values and a higher normalised value for the top 5%. This is intentional to have the top values be weighted even higher. Doing this emphasises those top results more when performing the query.
The normalised values are then written to the disk with the help of the Alias System as shown in chapter 2.1.3 and used to calculate the words or phrases that are related to another word or phrase not directly through their FMI value, but through co-occurrence of of a word or phrase that has a high FMI value with the original.
The calculated triples are written to the disk and then used to build a vector space and calculate the Cosine Similarity in the next step.

2.4 Calculation of Cosine Similarity based on FMI values

The goal of this step is to compute tokens or phrases that are *related to* another token or phrase. Note that *related to* refers specifically to the relation calculated over the vector space of the SPO-triples, their similarity to be exact. The exact process will be explained in this chapter, but first that vector space must be built.
The *close to* SPO-triples calculated with their FMI in the previous chapter 2.3 are loaded in using the Alias System 2.1.3. Then a sparse matrix representation of those triples is calculated. A sparse matrix is a matrix in which most elements are zero.

Table 2.5: Example sparse matrix

	apple	tree	leaf	fruit
apple	0	0.9	0.5	0
tree	0.9	0	0.6	0
leaf	0.5	0.6	0	0.4
fruit	0	0	0.4	0

It is constructed by iterating over the SPO-triples we calculated in the previous step. The way the matrix as represented in table 2.5 is read is as follows:

tree is close to leaf with an FMI value of 0.6 (row 3)

2.4.1 Cosine Similarity and its role in dragn

The Cosine Similarity for two vectors x and y is given by:

$$cos(x, y) = \frac{x \cdot y}{|x| \cdot |y|}$$

Where

$$x = (x_1, x_2, ...x_n)^\intercal$$

$$y = (y_1, y_2, ...y_n)^\intercal$$

are vectors in an n-dimensional vector space and

$$|x| = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$$

is the length of the vector, as discussed in [**Sin01**] and [**RKA12**].
Simply put, the cosine of the angle between two vectors is used to compare the similarity between the vectors. The lower the angle the more similar the vectors are.

Continuing the example from above, the rows in table 2.5 are vectors in the vector space over the processed texts. We can compute the similarity for "apple" and "tree" based on their respective FMI values with other words in the table.

$$x_y = \text{FMI value for x with y}$$

Then we calculate the lengths of the "tree" and "apple" vectors:

$$|tree| = \sqrt{tree_{apple}^2 + tree_{leaf}^2} = \sqrt{0.9^2 + 0.6^2} = 1.08$$
$$|apple| = \sqrt{apple_{tree}^2 + apple_{leaf}^2} = 1.030$$

Next we calculate the product of the two rows in table 2.5:

$$\begin{aligned}\overrightarrow{tree} \cdot \overrightarrow{apple} &= tree_{apple} * apple_{apple} + tree_{tree} * apple_{tree} + tree_{leaf} * apple_{leaf} + tree_{fruit} * apple_{fruit} \\ &= 0.9 * 0 + 0 * 0.9 + 0.6 * 0.5 + 0 * 0 = 0.6 * 0.5 = \\ &= 0.3\end{aligned}$$

We insert those values into the Cosine Similarity formula and get:

$$cos(tree, apple) = \frac{0.3}{1.08 * 1.03} = 0.270$$

Therefore the Cosine Similarity based on the FMI values is 0.270, which is on the lower side. The Cosine Similarity values range from 0 to 1, with 1 indicating there is no angle between the two vectors, thus showing they are pointing in the same direction in their vector space.
Let

$$x = (x_1, x_2, ..., x_n)^\mathsf{T}$$
$$y = (y_1, y_2, ..., y_n)^\mathsf{T}$$
$$x = y$$

then

$$|x| = \sqrt{x_1^2 + x_2^2 + ... + x_n^2} = \sqrt{y_1^2 + y_2^2 + ... + y_n^2} = |y|$$

$$|x| * |y| = |x|^2 = x_1^2 + x_2^2 + ... + x_n^2$$

$$cos(x, y) = \frac{x_1 * y_1 + x_2 * y_2 + ...x_n * y_n}{|x| * |y|} = \\ = \frac{x_1^2 + x_2^2 + ... + x_n^2}{x_1^2 + x_2^2 + ... + x_n^2} = 1$$

Therefore for two tokens that have the exact same FMI values and thus co-occurrences with the same distances their similarity value will be one, indicating they are functionally identical in this context.

The Cosine Similarity can be used to calculate similarity between documents as described in [**Sin01**], however as shown in this chapter one can calculate the similarity between vectors of FMI values. The result we get here is a similarity score calculated between two phrases or words in a text, based on their FMI values, which as explained in chapter 2.3 is based on the co-occurrence of phrases in a text. The Cosine Similarity in this context is therefore grounded in the co-occurrence of two phrases not with each other, but with tokens in texts they both co-occur with. Their similarity increases

A similarity scoring function is useful in the context of Distant Reading, for given words appearing in texts we want to find words or phrases that are related in some way. This system makes the assumption that words with word vectors pointing in similar directions (thus having a high Cosine Similarity) are related to each other. For their similarity to be high, they need to have high FMI for the same words. In simpler terms, if two words co-occur often with the same words but not each other, the assumption is made that they are related.

The FMI value that is used as the score value for the similarity computation connects two phrases or words when they co-occur in a text. By using those values as the basis for the calculations we can find phrases that are similar by checking which words they share a high FMI value with, or

in other words, how big the angle between the two vectors of FMI values is.
Ultimately we find phrases that share common FMI values. If a word co-occurs with other words often, and a second word co-occurs with the same words and a comparable frequency, we establish a relation between the two with the method discussed in this chapter. Those two words appear near the same words often, so we can assume the two words themselves are related. This gives us a less-intuitive way of finding connections between words in texts, that would be hard to accomplish by hand.
We now have useful information that we could query and gain information about texts from. To do that, the following chapter shows how the produced files are formatted and written to new files that can be used as the basis for the queries or used in a different system.

2.5 Creation of formatted files / index

The ability to expand and build upon **dragn** is one of the core goals of the overall design of the system. In the previous chapters the algorithms and calculations performed in the system were outlined, all that is left is bringing the data in a form that is easier to query. In this chapter I will show how the different files for the queries are created and in which format they are. It is then possible to use the results produced in **dragn** for other systems but with a different scope or use. The subsection titles refer to function names in the system.

2.5.1 Creation of SPO-triples list

First the SPOP-tuples as created in chapter 2.3 are loaded. They are in the format:

<subject> <predicate> <object>, score

Where *subject* and *predicate* are words or phrases from the texts that the system found a relation between, and *predicate* is one of *close to* for the FMI relations or *related to* for the similarity relations.
All the relations that were found for the current text(s) are written to a file on the disk.
The format is, where

t is the tab character:

```
subject\tpredicate\tobject\tscore
```

By loading this file and parsing the data one can have access to all the relations that were calculated over a set of texts. The Alias System as outlined in chapter 2.1.3 makes it possible to keep the files created for collections of texts and compare the results, as an example usage.

2.5.2 Creation of SPO shortlist

In this file, the subject, object and score of the SPOP-tuples are saved in the format:

```
subject\tobject\tscore
```

The contents of this file can be used to quickly find which words have a relation between them and with what value. Due to the reduced content of the file it is smaller and can be loaded faster, which becomes increasingly more relevant with the number and size of the processed texts.

2.5.3 Creation of SPOP-tuples list

The first part of this function is to create an inverted index over the processed texts. This index is used in the following chapter to improve the information value of the query results.
The second and main purpose of this function is to map SPO-triples to their score in provenances.
The format is:

```
(subject, predicate, object)\t[(provenance, score)]
```

The "related to" triples are added separately to get a text-specific set of relations. We do that by iterating over all the "related to" triples we have to get all the relations with them and to know the provenances they belong to. We iterate over all "related to" triples and for the subject and object of the relations we find the relations they have in common.
For those extracted relations two new triples are built:

subject predicate object-from-commons

object-from-commons predicate object

Where *object-from-commons* is the object of the (relation, object) tuple. The tuples are mapped

against the word they have a relation with, so one can access all the relations a word or phrase has and finds a list of the tuples described.

```
mapping = {
    "apple": [(close to, tree), (related to, leaf)]
}
mapping["apple"] = [(close to, tree), (related to, leaf)]
```

Listing 2.2: Example relation mapping for "apple"

For those new triples the mapping of triples to (provenance, score) tuples is checked and for each of the provenances the maximum score is extracted, then we add a line in the known format:
subject predicate object, provenance, score
Where score is updated to be:

```
score = original\_triple\_score * new\_triple\_score
```

and subject or object respectively replaced with the object from the (relation, object) tuple. This results in artificial "related to"-triples that exist in provenances. An example of the usage: *apple related to tree, 0.8* is the triple we look at. We find:

```
relations[apple] = (close to, fruit)
relations[tree]  = (close to, fruit)
```

close to fruit is found in the relations for both *tree* and *apple*. Therefore we check the two triples:

```
apple close to fruit (we keep the subject from the original triple)
tree close to fruit (we keep the object from the original triple)
```

We now look up the triple-provenance mapping and find the following:

```
provenance_mapping[(apple close to fruit)] = (story1_1.txt, 0.6)
provenance_mapping[(tree close to fruit)]  = (story1_1.txt, 0.7)
```

The maximum score for story1_1.txt is 0.7 for the triple "tree close to fruit", which is the triple where we kept the object. That means we keep the original object and replace the subject with *fruit* and get:

```
fruit related to tree, story1_1.txt, 0.8*0.7
```

We thus can say that *fruit is related to tree in the text story1_1.txt with a score of 0.56.* This was created because we have the original relation *apple related to tree* and *fruit* appears near both *apple* and *tree* in the same paragraph, so we augment the data by adding this new relation *fruit related to tree.*

2.5.4 Summary of the indexing

In this chapter I have shown how the calculated data is augmented to improve results and how it is brought into formats that are easier to use with my own system but also to use with other systems. For that reason the structure of the data was explained, thus enabling the reader to use the data in their own work easily.
This step is rather complex from a programming and understanding standpoint.The reader is encouraged to use this chapter and the files created in this step for their own project or research. Now that the files are all processed and in place, all that is left is to perform queries. The usage of the files created in this chapter and their role in the queries will be shown in the following chapter and the last step.

2.6 Calculation of query results

This chapter will explain how queries are answered in **dragn**. The implementation shown here is just an example, the data output outlined in chapter 2.5 can be used in different ways and for different projects. In this chapter however the output of the previous chapter will be used as is.

2.6.1 Calculation of relevance to the query

First a *Fuzzy Set* over all relations that contain a query term is calculated using the mapping of tokens to their contained relations, see chapter 2.5.1. The original implementation of the fuzzy sets and the idea to use them in this way come from Vit Novacek [**NB14**].
Whereas a "normal" set has a binary relation to its contained elements (an element is either part

of the set or not) a fuzzy set has a degree of membership, as introduced in [**Zad65**].
Calculating a fuzzy set in the context of the system and specifically the query results in a fuzzy set describing how high the membership of a token to the query is.
If a query contains more than one query term and terms share a relation but with different scores, the higher of the scores is used as the membership.
All relations (= SPO-triples) containing members of this fuzzy set and a query term are then checked one by one. The score of the SPO-triples is then multiplied by the membership from the fuzzy set.
This increases the score of low-scoring relations if the complementary relation has a higher value and decreases the scores overall. A high degree of membership increases the score in comparison to low membership.
A dictionary then holds the score we just calculated for subject and object of the SPO-triples, the same steps are performed for all relations containing fuzzy set member and a query term. The result is a dictionary mapping tokens to a score, the tokens sorted by their score will become the result of the query later.
Additionally the provenances for the SPO-triples are loaded from the mapping (see 2.5.3) and the multiplication of membership and score is performed again as above. This score and the triple are then likewise mapped to each provenance.
With this relatively simple calculation out of the way it is now possible to create a graph showing relations between tokens from the texts and the query terms and show relevant text samples. This will be discussed in the following chapter.

2.6.2 Graph generation and the dragn graph system

dragn uses its own, simple graph system with three parts:

- **Nodes**

  They have a name and width and colour. A container for Edge objects.

- **Edges**

  They have a start and an end, which can be anything. Intended for them to be Nodes.

- **Graph**

  A graph is a container for nodes.

Important here is that the graph system is very simply built and easily extendable. The main purpose of it is to generate RFC4627 compliant JSON, as described in [**Cro06**]. The simplistic nature allows for the creation of any kind of graph that can be exported to JSON very easily.
In **dragn** the graphs are created from the data in the previous chapter, 2.6.1. A maximum number of nodes is created based on the sorted scores. Each node represents a token or phrase from the processed texts. If the node label appears in the query, the node is coloured differently to make it easier for the user to find what they are interested in.
After the nodes are generated, all the relations are checked. If both subject and object appear as a node, an edge is created between them, with the colour showing their relation, up to a maximum number of edges as specified by the user.

2.7 dragn interface

This section contains a short explanation and summary of the **dragn** user interface.

2.7.1 Navigation bar

The navigation bar is where the user enters the information for their query. It contains buttons to go to the pages for text upload and text processing.

Figure 2.2: Overview of "dragn" interface navigation bar

- 1: The buttons to go the upload and process pages, as well as back to the main page.
- 2: The maximum number of nodes to show in the result graph.
- 3: The maximum number of edges to have in the result graph. As edges can overlap and generally go both ways, this is not an exact value.

- 4: The number of paragraphs to return.
- 5: Whether to consider relations between words that are not in the query or not.
- 6: The dropdown from which the Alias (see chapter 2.1.3) for the query is selected.
- 7: The button to initiate the query.
- 8: The context reader. Used to quickly move between texts that contain a certain word.
- 9: The button to scroll back to the top of the page where the result graph is shown.
- 10: The field for the query.

2.7.2 Result graph options

The result graphs can be re-ordered or be used to add certain tokens to the query. The context menu shown here can be brought up by right clicking a node in the graph.

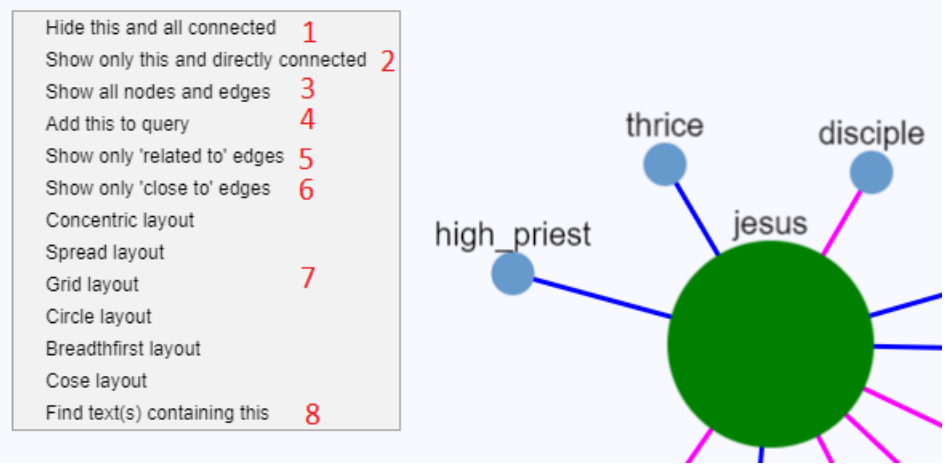

Figure 2.3: Overview of "dragn" result graph and options

- 1: Hides the selected node and all connected nodes.
- 2: Hides all but the selected node and all connected nodes.
- 3: Shows all previously hidden nodes.
- 4: Adds the token to the query.

- 5: Shows only the "related to" edges, the ones described in chapter 2.4.
- 6: Shows only the "close to" edges, the ones described in chapter 2.3.
- 7: Layout options for the nodes.
- 8: Adds the token to the context reader.

The context reader has been briefly mentioned in 2.7.1. It allows the user to scroll to paragraphs containing a certain word to get better understanding of either the relations it has with other words or to gain more understanding of the word itself by reading the paragraphs it appears in.

2.7.3 Result graph

A result graph can be seen in figure 2.3.
The result graph is interactive and created with Cytoscape.js [**FLH+16**], [**Cyt**]. By default the ordering is random, but different automatic node alignments are available.
The following relations can be found in the result graph:

colour	meaning
blue	close to, FMI-based
red	related to, Cosine Similarity based on FMI scores
magenta/pink	both close to and related to

Table 2.6: Explanation of edge colours

2.7.4 Result paragraphs and relations

Each query produces not only the result graph, but a list of paragraphs relevant to the query with relations relevant to the query that appear in those paragraphs.
The paragraphs are scored based on their relevancy to the query as described in chapter 2.6, with the highest scoring paragraph having a score of 1. For each paragraph the contained relations that are relevant to the query are shown in a sortable table, containing both the relations based on the FMI (see chapter 2.3) as well as those over the cosine similarity (see 2.4).
Additionally each paragraph in the result has buttons for "Previous" and "Next" paragraphs, respectively showing the paragraph preceding or following the paragraph for which the button

was pressed in an overlay, which contains the same buttons. With this it is possible to get additional understanding of a certain paragraph by reading the paragraphs surrounding it.
Words in the result paragraphs can be added to the query directly by simply clicking them, thus allowing the user to explore possible links found in the paragraphs directly and easily.

2.8 Summary of the architecture overview

dragn consists of a pipeline of steps, where the output of the previous step forms the basis for the next step. They are:

- Splitting the texts into paragraphs and extracting Noun Phrases as described in 2.2
- Calculating the modified PMI, the FMI, to establish connections between tokens in the texts, 2.3
- Calculating the Cosine Similarity based on the FMI of the tokens, as shown in 2.4
- Formatting the results into files to use for the queries, 2.5
- Performing the queries based on the available data of the processed texts, as produced in the previous steps, was described in chapter 2.6

The system was built upon the foundation of "Skimmr", the predecessor tool developed by Vit Novacek, who has written about it in [**NB14**].
The code of the new system is up to current technical standards and a more intuitive user interface lets the user perform their tasks better and helps them understand the connections between tokens in their texts better. Next follows an example use case utilising the new system's capabilities.

Chapter 3

Use case

3.1 Setup for the use case

One example use case of the system with multiple queries and the results that can be derived from them will be discussed in this chapter. The text that will be used is *The King James Version of the Bible* as found on Project Gutenberg [**PGB**].
It was chosen for two reasons:

- well-known, the reader can be expected to have sufficient knowledge to follow the use case even if they have not read it in its entirety
- big size, analysing by hand would take considerable effort and thus using a tool such as **dragn** is preferable

I want to emphasise that while only a single text is used for this use case, achieving the result demonstrated in this chapter would be analogous for multiple texts. The full text has been sent through the **dragn** pipeline and processed beforehand.
Unless otherwise specified only the 10 highest scoring relations for each relation type were included in the tables.

3.2 Query: jesus

The parameters for the first query are as shown here:

- Query: jesus
- Max nodes: 11
- Max edges: 300
- Top text samples: 100

Setting the maximum number of nodes to 11 results in a graph containing 'jesus' and ten words or phrases related in some way to 'jesus'. Setting the number of edges (also known as vertices) to a high value means all the connections will be shown. It is possible to set it to a lower value to see only the relations with the highest values if so desired.

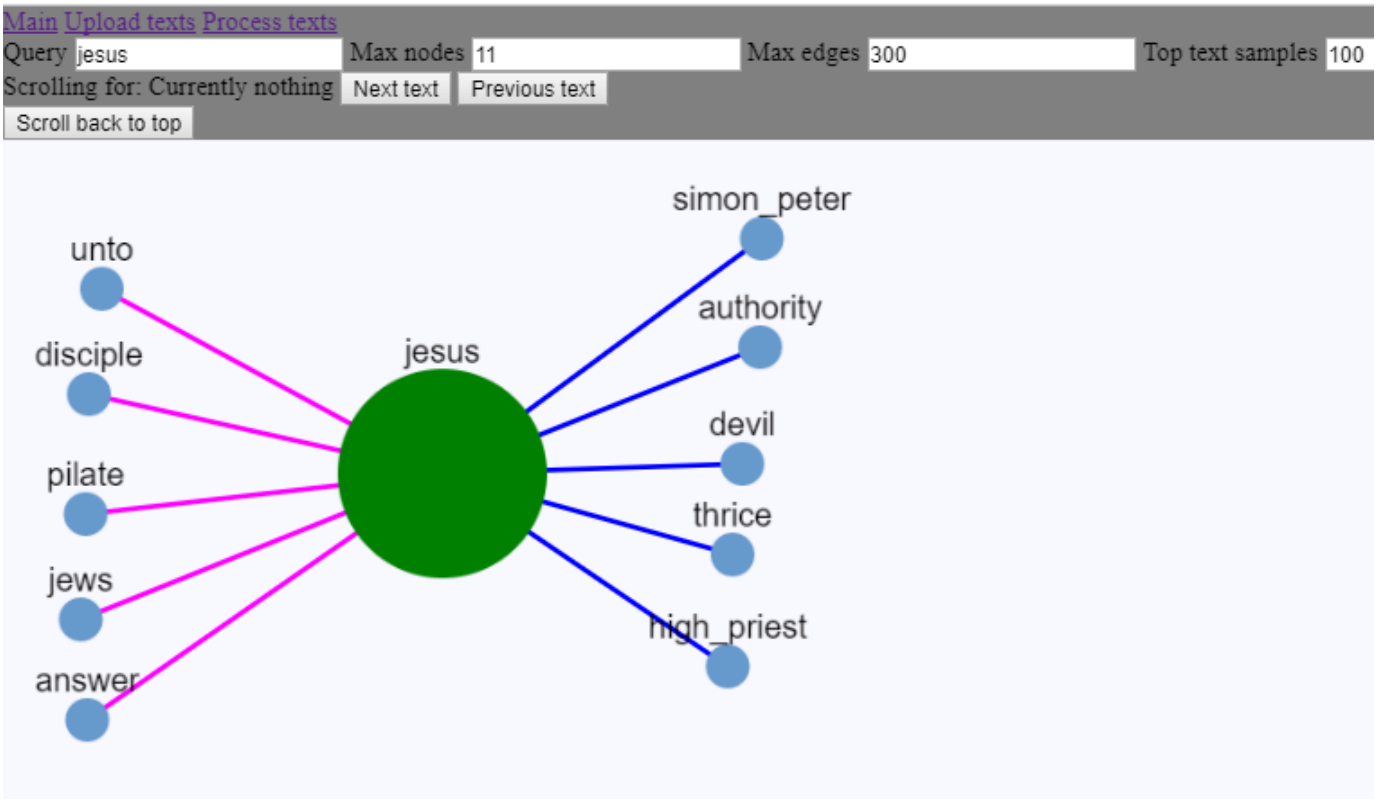

Figure 3.1: Graph for query 'jesus'

3.2.1 Relations in the graph for 'jesus,'

The following relations can be found in the result graph:

subject	predicate	object
jesus	close to	simon_peter
jesus	close to	authority
jesus	close to	devil
jesus	close to	thrice
jesus	close to	high_priest
jesus	close to	unto
jesus	close to	disciple
jesus	close to	pilate
jesus	close to	jews
jesus	close to	answer
jesus	related to	unto
jesus	related to	disciple
jesus	related to	pilate
jesus	related to	jews
jesus	related to	answer

Table 3.1: Relations in the result graph for the query 'jesus'

3.2.2 Paragraphs relevant to the query 'jesus'

The three most relevant paragraphs with their score in regards to the query are:

bible_full.txt_21514, 1 (Paragraph 21514 in the file)

18:33 Then Pilate entered into the judgment hall again, and called Jesus, and said unto him, Art thou the King of the Jews? 18:34 Jesus answered him, Sayest thou this thing of thyself, or did others tell it thee of me? 18:35 Pilate answered, Am I a Jew? Thine own nation and the chief priests have delivered thee unto me: what hast thou done? 18:36 Jesus answered, My kingdom is not of this world: if my kingdom were of this world, then would my servants fight, that I should not be delivered to the Jews: but now is my kingdom not from hence.

subject	predicate	object	score
jesus	close to	pilate	1.0
jesus	close to	jews	1.0
jesus	close to	unto	1.0
jesus	close to	enter	1.0
jesus	close to	answer	1.0
jesus	close to	say	1.0
jesus	close to	world	0.8775063207175372
jesus	close to	king_of_jews	0.676004216459174
jesus	close to	judgement_hall	0.6739292059866504
jesus	close to	my_kingdom	0.5174179934374722
jesus	related to	jews	0.1706899529268509
jesus	related to	say	0.09814261250477314
jesus	related to	answer	0.034030420168504064
jesus	related to	call	1.8997625097621448e-07

Table 3.2: Relations found for bible_full.txt_21514

bible_full.txt_18763, 0.976216153617806313358088895

4:12 Now when Jesus had heard that John was cast into prison, he departed into Galilee; 4:13 And leaving Nazareth, he came and dwelt in Capernaum, which is upon the sea coast, in the borders of Zabulon and Nephthalim: 4:14 That it might be fulfilled which was spoken by Esaias the prophet, saying, 4:15 The land of Zabulon, and the land of Nephthalim, by the way of the sea, beyond Jordan, Galilee of the Gentiles; 4:16 The people which sat in darkness saw great light; and to them which sat in the region and shadow of death light is sprung up.

subject	predicate	object	score
jesus	close to	galilee	1.0
jesus	close to	nazareth	1.0
jesus	close to	john	1.0
jesus	close to	saw	1.0
jesus	close to	sit	1.0
jesus	close to	say	1.0
jesus	close to	depart	1.0
jesus	close to	capernaum	0.8716620885947697
jesus	close to	fulfill	0.8598812801210459
jesus	close to	dwelt_in_capernaum	0.2627662846808881
jesus	related to	john	0.19577093636686257
jesus	related to	galilee	0.1565230454832303
jesus	related to	nazareth	0.11995120425385629
jesus	related to	depart	0.11585843258452326
jesus	related to	say	0.11561349559243331
jesus	related to	saw	0.11249972841114071

Table 3.3: Relations found for bible_full.txt_18763

bible_full.txt_18947, 0.93378342385614294973457592 80

11:2 Now when John had heard in the prison the works of Christ, he sent two of his disciples, 11:3 And said unto him, Art thou he that should come, or do we look for another? 11:4 Jesus answered and said unto them, Go and shew John again those things which ye do hear and see: 11:5 The blind receive their sight, and the lame walk, the lepers are cleansed, and the deaf hear, the dead are raised up, and the poor have the gospel preached to them.

subject	predicate	object	score
jesus	close to	raise	1.0
jesus	close to	disciple	1.0
jesus	close to	preach	1.0
jesus	close to	say	1.0
jesus	close to	answer	1.0
jesus	close to	unto	1.0
jesus	close to	gospel	0.7220483212774963
jesus	close to	john	0.6948157183491428
jesus	close to	receive	0.31074260643649365
jesus	close to	deaf_dead	0.2668806801251524
jesus	related to	answer	0.1307478787056254
jesus	related to	say	0.12355058556155199
jesus	related to	kingdom_of_god	0.09684031278231553
jesus	related to	suppose	0.0777098842002729
jesus	related to	disciple	0.06644093989493592
jesus	related to	receive	0.027963657641532764
jesus	related to	come	1.9729556723760588e-07

Table 3.4: Relations found for bible_full.txt_18947

3.2.3 Analysis of the query 'jesus'

A simple one-word query will produce a graph with words or phrases that co-occur either directly with the word from the query or that co-occur with the same words as discussed in chapters 2.4 and 2.6.1.

As such, it is not unexpected that the result graph shows that *Jesus* often co-occurs with a disciple of his, *Simon Peter* or the man responsible for making him a martyr, *Pilate* as both seen in 3.1 and another of his disciples, John, as we can see in table 3.3. In the same table we can see that the highest scoring of the Cosine Similarity relations (related to) is the one with John; indicating that 'John' and 'Jesus' have a high similarity. Further the expected connections between his preachings and the audience thereof can be found in the highest scoring paragraph, as seen in table 3.2: *jesus close to jews, jesus close to king_of_jews, jesus close to preach.* Jesus had a strong connection to Jews and was known for his preachings. The result graph emphasises this connection and lets the user gain information about Jesus and what he did without reading a single page of the bible. Of course such an analysis is very superficial, but that is a given considering the nature of Distant Reading.

Especially interesting is paragraph 21514 (see table 3.2). In it, Jesus has a conversation with Pilate, the man who would later sentence him to be crucified. In the conversation, Jesus indicates that he is the servant of heaven: "*My kingdom is not of this world*". This short paragraph captures the spirit of Jesus relatively well, essential information about him is contained therein. His celestial nature is further extrapolated by the second-highest scoring paragraph: "*The people which sat in darkness saw great light; and to them which sat in the region and shadow of death light is sprung up.*" and the third-highest scoring paragraph, where Jesus talks about the miracles he performed: "*Go and shew John again those things which ye do hear and see: [...] The blind receive their sight, and the lame walk, the lepers are cleansed, and the deaf hear, the dead are raised up*".

The appearance of 'devil' in the result graph may be unexpected at first and indicate an unexpected connection between Jesus and the Devil. Most likely the connection stems from the passages of the bible where the Devil tempts Jesus. 'Devil' is absent from the highest scoring paragraphs because the relation between 'Jesus' and 'Devil' is not strong enough to score the paragraphs containing it high enough. To further explore the aforementioned link of the rela-

tion 'jesus close to devil' we can see in the graph 3.1 we can perform a second query: 'jesus,devil'.

3.3 Query: jesus, devil

The parameters for the first query are as shown here:

- Query: jesus,devil
- Max nodes: 11
- Max edges: 300
- Top text samples: 100

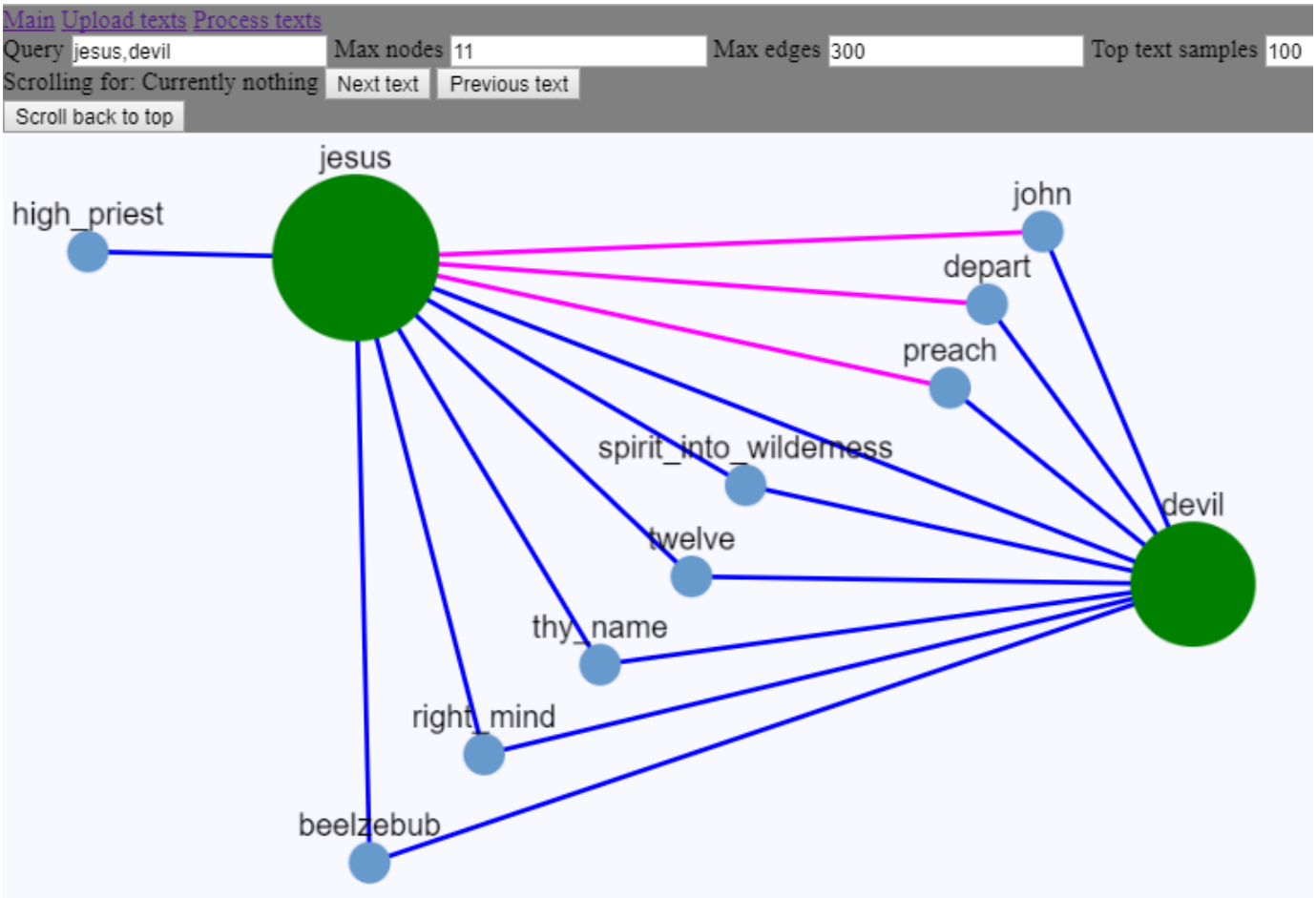

Figure 3.2: Graph for query 'jesus,devil'

3.3.1 Relations in the graph for 'jesus,devil'

The following relations can be found in the result graph:

subject	predicate	object
jesus	close to	high_priest
jesus	close to	beelzebub
jesus	close to	right_mind
jesus	close to	thy_name
jesus	close to	twelve
jesus	close to	spirit_into_wilderness
jesus	close to	devil
jesus	close to	preach
jesus	close to	depart
jesus	close to	john
jesus	related to	preach
jesus	related to	depart
jesus	related to	john
jesus	related to	jews
jesus	related to	answer
devil	close to	beelzebub
devil	close to	right_mind
devil	close to	thy_name
devil	close to	twelve
devil	close to	spirit_into_wilderness
devil	close to	jesus
devil	close to	preach
devil	close to	depart
devil	close to	john
devil	related to	preach
devil	related to	depart
devil	related to	john

Table 3.5: Relations in the result graph for the query 'jesus,devil'

3.3.2 Paragraphs relevant to the query 'jesus,devil'

bible_full.txt_18763, 1

4:12 Now when Jesus had heard that John was cast into prison, he departed into Galilee; 4:13 And leaving Nazareth, he came and dwelt in Capernaum, which is upon the sea coast, in the borders of Zabulon and Nephthalim: 4:14 That it might be fulfilled which was spoken by Esaias the prophet, saying, 4:15 The land of Zabulon, and the land of Nephthalim, by the way of the sea, beyond Jordan, Galilee of the Gentiles; 4:16 The people which sat in darkness saw great light; and to them which sat in the region and shadow of death light is sprung up.

subject	predicate	object	score
jesus	close to	depart	1.0
jesus	close to	say	1.0
jesus	close to	sit	1.0
jesus	close to	nazareth	1.0
jesus	close to	john	1.0
jesus	close to	galilee	1.0
jesus	close to	saw	0.8775063207175372
jesus	close to	capernaum	0.8716620885947697
jesus	close to	fulfill	0.8598812801210459
jesus	close to	zabulon_nephthalim	0.2627662846808881
jesus	related to	john	0.19577093636686257
jesus	related to	galilee	0.1565230454832303
jesus	related to	nazareth	0.11995120425385629
jesus	related to	depart	0.11585843258452326
jesus	related to	say	0.11561349559243331
jesus	related to	saw	0.11249972841114071

Table 3.6: Relations found for bible_full.txt_18763

bible_full.txt_19624, 0.998009490199328966661295721 01

“ 5:15 And they come to Jesus, and see him that was possessed with the devil, and had the legion, sitting, and clothed, and in his right mind: and they were afraid.”

subject	predicate	object	score
jesus	close to	devil	1.0
jesus	close to	sit	1.0
jesus	close to	right_mind	0.7637623651652926
jesus	close to	legion	0.581402615101797
jesus	close to	afraid	0.08126797684445049
devil	close to	possess	1.0
devil	close to	right_mind	1.0
devil	close to	clothe	0.522321459997405
devil	close to	legion	0.513647727750805
devil	close to	afraid	0.14408345959927174
jesus	related to	unto	0.09325454471923895
jesus	related to	come	1.9729556723760588e-07

Table 3.7: Relations found for bible_full.txt_19624

bible_full.txt_18947, 0.9908170021801287219957252699

11:2 Now when John had heard in the prison the works of Christ, he sent two of his disciples, 11:3 And said unto him, Art thou he that should come, or do we look for another? 11:4 Jesus answered and said unto them, Go and shew John again those things which ye do hear and see: 11:5 The blind receive their sight, and the lame walk, the lepers are cleansed, and the deaf hear, the dead are raised up, and the poor have the gospel preached to them.

subject	predicate	object	score
jesus	close to	preach	1.0
jesus	close to	disciple	1.0
jesus	close to	say	0.7637623651652926
jesus	close to	unto	0.581402615101797
jesus	close to	answer	0.08126797684445049
jesus	close to	gospel	0.7220483212774963
jesus	close to	john	0.6948157183491428
jesus	close to	receive	0.31074260643649365
jesus	close to	deaf_dead	0.2668806801251524
jesus	close to	lame_walk_leper	0.2668806801251524
jesus	related to	answer	0.1307478787056254
jesus	related to	say	0.12355058556155199
jesus	related to	unto	0.10397979732017294
jesus	related to	kingdom_of_god	0.09684031278231553
jesus	related to	suppose	0.0777098842002729
jesus	related to	disciple	0.06644093989493592
jesus	related to	receive	0.027963657641532764
jesus	related to	come	1.9729556723760588e-07

Table 3.8: Relations found for bible_full.txt_18947

3.3.3 Analysis of the query 'jesus,devil'

As we can see, 'bible_full.txt_18763', section 3.3.2, has a higher score for this query than for the previous one. This simply means that for the second query 'devil' has a relation with at least one of the phrases occurring in this paragraph. The relation does not have to occur in that particular paragraph for it to play a role in the score, 'devil close to / related to some-phrase-in-the-paragraph' increases the score for that particular paragraph. In the case of the two queries explored thus far, it is enough to boost the score over the query for just 'jesus'.
Further of note is the size of the nodes for 'jesus' and 'devil': A bigger size means that the node's relations have overall higher values, thus the relations containing 'jesus' have, on average, a higher score than the ones containing 'devil'.
To showcase the actual distant reading capabilities of **dragn**, all texts containing 'devil' that were relevant to the query will now be examined. Doing so is simple, simply select the option from the context menu for 'devil'. The first three found paragraphs are as follows:

> **bible_full.txt_19138, 0.75045290322628340519037392 87**
> 17:18 And Jesus rebuked the devil; and he departed out of him: and the child was cured from that very hour.

> **bible_full.txt_20138, 0.74306027286191294030158131 2**
> 4:1 And Jesus being full of the Holy Ghost returned from Jordan, and was led by the Spirit into the wilderness, 4:2 Being forty days tempted of the devil. And in those days he did eat nothing: and when they were ended, he afterward hungered.

> **bible_full.txt_18754, 0.62344194712910246329642967 87**
> 4:1 Then was Jesus led up of the spirit into the wilderness to be tempted of the devil.

The paragraphs shown here are the first three containing the word 'devil' in the result for the query 'jesus,devil'.
Using the context reader it is to possible to gain further context for the passages. It allows reading the text passages before and after a given paragraph, thus allowing the reader to gain additional understanding by reading only a few paragraphs as opposed to reading the entire book.
Reading the paragraphs before 19138 lets the reader find mention of a character named 'Elias':

bible_full.txt_19133

17:12 But I say unto you, That **Elias** is come already, and they knew him not, but have done unto him whatsoever they listed. Likewise shall also the Son of man suffer of them.

Now the analysis in section 3.2.3 can be continued. As expected, the connection between Jesus and the Devil is not an unexpected one, Jesus exorcises a demon, referred to as 'devil' in paragraph 19138, comes face to face with a demon (referred to as 'devil') in table 3.7, and is tempted while meditating or praying in the wilderness. The found paragraphs contain interactions between Jesus and the Devil; paragraphs that would be hard to find by hand without the assistance of a tool. **dragn** gives the reader additional understanding of the paragraphs by allowing the user to read previous and following text passages. Thus additional context and understanding of the bible passages can be gained without reading them in their entirety.

Finding the passages in the bible containing both characters was made very easy as well, a task which would took significant effort otherwise.

Unclear phrases or words can be added to the query to get information about them the same way was done with 'Jesus'.

One such case will be examined using the name 'Elias': Who is Elias and what is his relation to Jesus? Is there a connection between Elias, Jesus and the Devil? Instead of reading further paragraphs, the user can utilise **dragn** to get an overview of that character and his relation to Jesus.

3.4 Query: jesus,elias

- Query: jesus,elias
- Max nodes: 11
- Max edges: 300
- Top text samples: 100

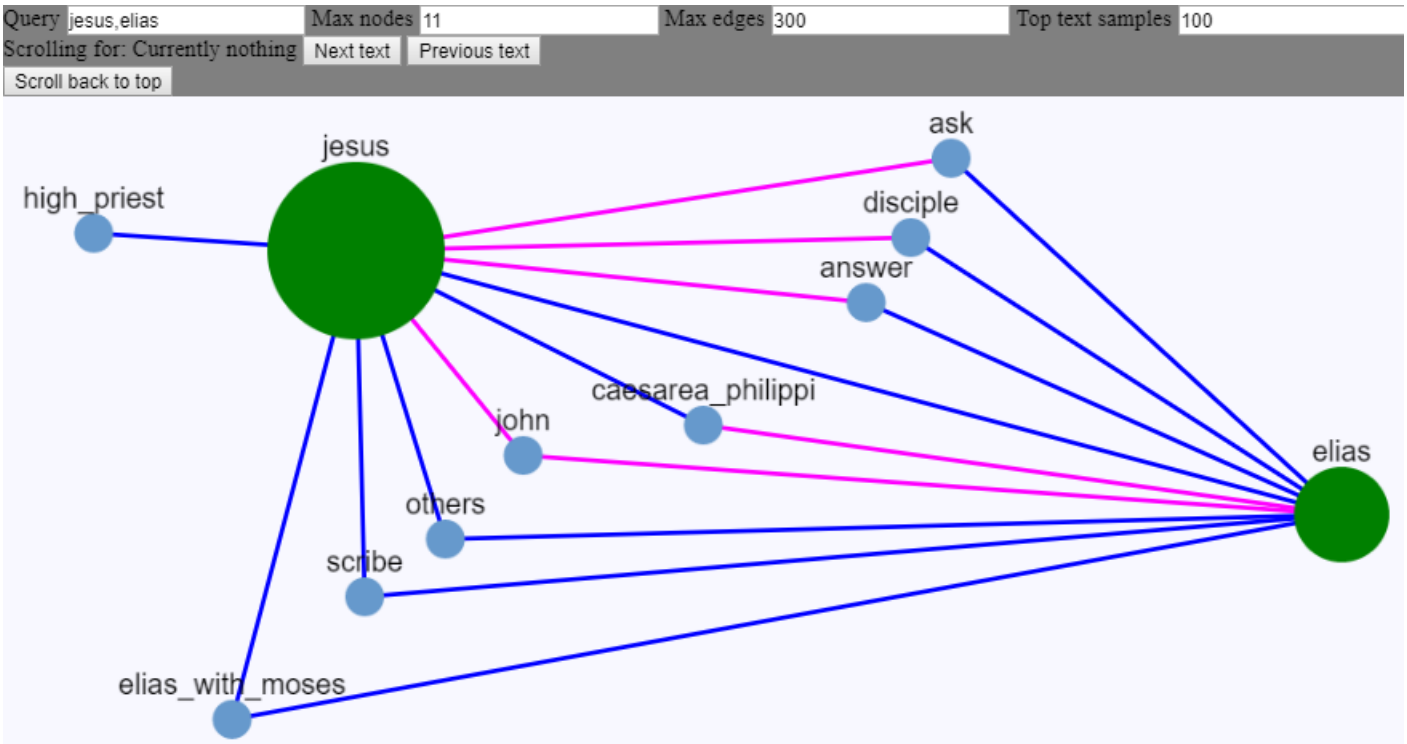

Figure 3.3: Graph for query 'jesus,elias'

3.4.1 Paragraphs relevant to the query 'jesus,elias'

As the goal of this query is to specifically find the relation between Jesus and Elias, and to understand who Elias is and what he does, the relations in the paragraphs are omitted. The three highest scoring paragraphs are:

> **bible_full.txt_19741, 1**
> 8:27 And Jesus went out, and his disciples, into the towns of Caesarea Philippi: and by the way he asked his disciples, saying unto them, Whom do men say that I am? 8:28 And they answered, John the Baptist; but some say, Elias; and others, One of the prophets.

> **bible_full.txt_19111, 0.776064213669834324123738014 7**
> 16:13 When Jesus came into the coasts of Caesarea Philippi, he asked his disciples, saying, Whom do men say that I the Son of man am? 16:14 And they said, Some say that thou art John the Baptist: some, Elias; and others, Jeremias, or one of the prophets.

> **bible_full.txt_19132, 0.771735089750873980937950215 1**
> 17:10 And his disciples asked him, saying, Why then say the scribes that Elias must first come? 17:11 And Jesus answered and said unto them, Elias truly shall first come, and restore all things.

3.4.2 Analysis of the query 'jesus,elias'

It becomes obvious that Jesus holds Elias in high regard and that Elias is a person of importance, as indicated by him saying that 'Elias truly shall first come, and restore all things' (Paragraph 19132). The result however does not easily reveal the nature of Elias; his profession or importance in more detail. The query being 'jesus,elias' most likely causes Jesus to overshadow Elias and thus the paragraphs that focus solely on Elias are scored lower. To fix that, a simple query of 'elias' can be performed.

3.5 Query: 'elias'

- Query: 'elias'
- Max nodes: 21
- Max edges: 300
- Top text samples: 100

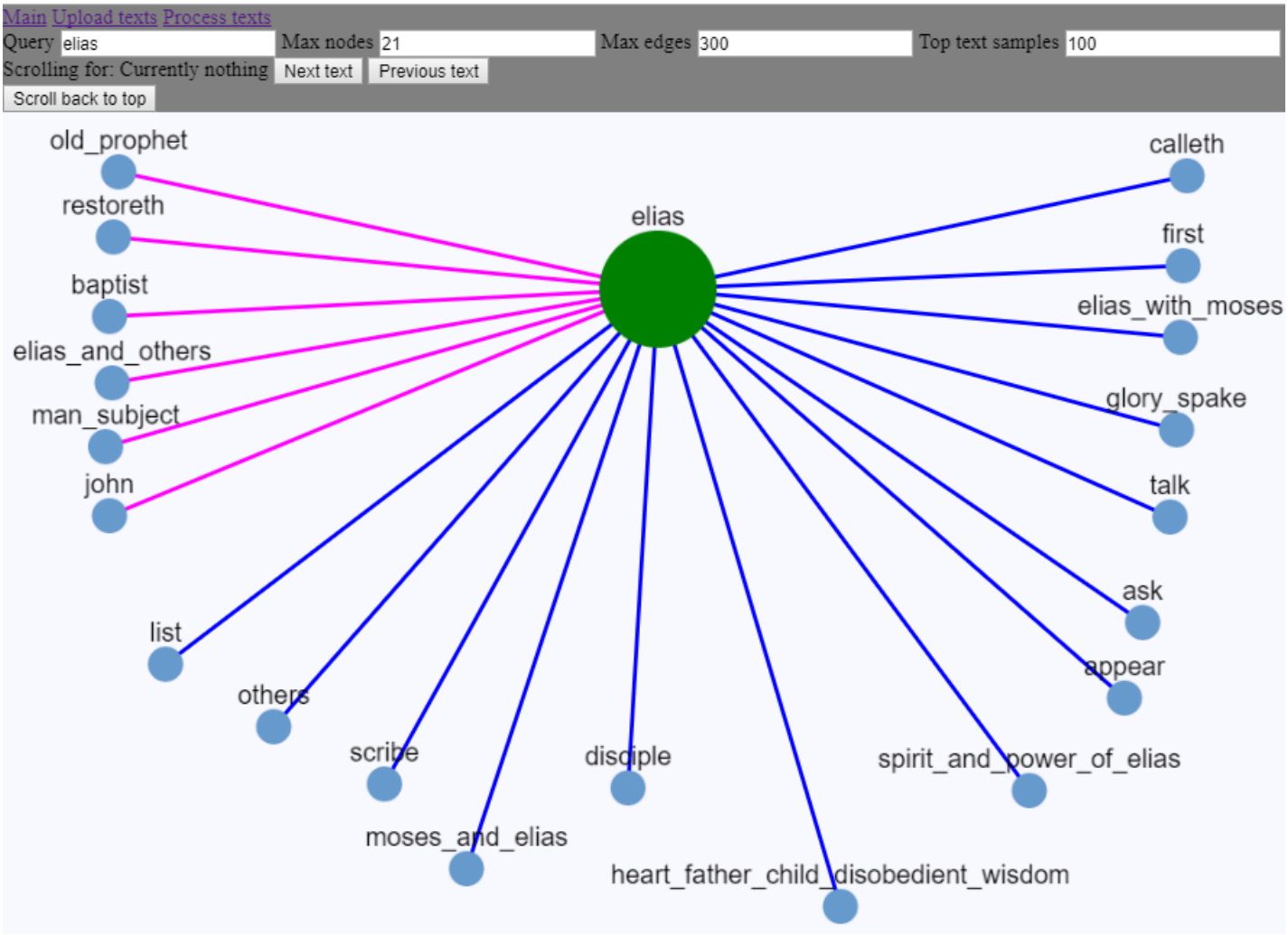

Figure 3.4: Graph for query 'elias'

3.5.1 Analysis of the result for query 'elias'

Again the paragraphs from chapter 3.4.1 containing 'Jesus' and 'Elias' are found. A quick glance over the found paragraphs shows that Elias is often mentioned together with Jesus and rarely alone. However, among the result one particular paragraph can be found:

> **bible_full.txt_24032, 0.17476822341006249859421759 79**
> 5:17 Elias was a man subject to like passions as we are, and he prayed earnestly that it might not rain: and it rained not on the earth by the space of three years and six months.

By reading the previous paragraph it becomes clear Elias is used solely as the example of a very faithful man:

> 5:16 Confess your faults one to another, and pray one for another, that ye may be healed. The effectual fervent prayer of a righteous man availeth much.
> 5:17 Elias was a man subject to like passions as we are, and he prayed earnestly that it might not rain: and it rained not on the earth by the space of three years and six months.
> 5:18 And he prayed again, and the heaven gave rain, and the earth brought forth her fruit.
> 5:19 Brethren, if any of you do err from the truth, and one convert him; 5:20 Let him know, that he which converteth the sinner from the error of his way shall save a soul from death, and shall hide a multitude of sins.

So faithful, that his prayers influence the weather on a global scale. Using that information, we can better understand the passage found for the query 'jesus,devil' (chapter 3.3.3) and 'jesus,elias' (chapter 3.4.1). The assumption can be made that Elias is mentioned as an example of a person with strong faith, a positive example that others shall follow. This additional information was not immediately clear when just reading the paragraphs on their own without the assistance of **dragn**.

3.6 Summary of the use case

Using **dragn** it was possible to gain a rudimentary understanding of 'Jesus', as seen in chapter 3.2. While a full analysis is neither possible nor intended, we were able to ascertain certain aspects of his character. Connections between Jesus and Jews were found, as were connections to his celestial nature in chapter 3.2.3. An unexpected connection between Jesus and the Devil could be checked as seen in chapter 3.3 and paragraphs describing their interactions were found. Both of those tasks would have taken a considerable amount of time when doing by hand or even using tools not designed for the task of Distant Reading, such as a basic search interface. Further, using the context reader it was possible to read passages before and after the paragraphs found as a result of the query in chapter 3.3.3 which lead us to find mention of a certain character named 'Elias'. Attempting to find a link between Jesus and Elias lead us to find paragraphs in which Jesus was either mistaken for Elias, or in which it seemed that Jesus held Elias in high regards and considered him a person of importance as described in chapter 3.4. It did not become apparent who Elias is or what he did to make him so noteworthy. Reading the result paragraphs for the query 'elias' however revealed that Elias became known for having faith strong enough to cause droughts or rain on a global scale as mentioned in chapter 3.5. Thus the mentions of Elias gained additional context: Elias was most likely used as a positive example, a person of strong faith, an example to follow. Using the **dragn** search interface it was possible to gain a basic understanding of Jesus, his connection and interactions to the Devil and who this 'Elias' person is and what he did, in a quick, direct and logical way, using Distant Reading and nodes in a graph.

Chapter 4

Conclusion

This project has showcased how the task of Distant Reading can be accomplished using a tool, by building upon an existing work in the area and improving the functionality to support domain experts working on tasks related to Distant Reading or text analysis in general. In this final chapter my contributions and findings will be summarised as well as possible future additions or improvements outlined.

4.1 Summary

The objective of this thesis was to highlight the usefulness of machine tools in a Distant Reading or text analysis context as well as to detail how such a task can be accomplished using modern technologies by building upon an existing system developed by Vit Novacek [**NB14**]. To do so, the system architecture as well as the required steps and the processes and algorithms performed in those steps were explained in detail. First, the texts are tokenised (see chapter 2.2), after which a modified PMI between the tokens is calculated as shown in chapter 2.3. After that, the calculated values are used to find the Cosine Similarity between the tokens to have two relations that can be shown to the user, as detailed in chapter 2.4. The data is then organised into files to use with the front end (chapter 2.5) and lastly the calculation of the user queries is outlined in chapter 2.6.

The usefulness of the system was emphasised by examining a use case, the King James version of the Bible, as seen in chapter 3. A simple query of 'jesus' was used as the initial query. The

result of the query was then used as the basis for further queries related to the respective previous one, to get understanding of the relation between characters and to find text passages containing those characters. Specifically, in many of the paragraphs found for the query 'jesus' a character named 'Elias' was mentioned. **dragn** could then be used to find out more about the relation between Jesus and Elias as well as help find passages explaining who Elias was and what made him noteworthy. The Distant Reader functionality played a part in this, it allowed us to read paragraphs before and after a given paragraph found as a result of the query to get additional information without having to read entire chapters or the entire book. Thus, Distant Reading was made possible by using a tool, **dragn**.
A short discussion and possible future work will be outlined in the final chapter.

4.2 Discussion and recommendations for further work

As shown as part of the thesis, **dragn** can be used to accomplish basic and more advanced Distant Reading tasks. Possible improvements and continuations of this work are as follows:

- extraction of all Named Entities and showing the relations between them in the result graph
- side-by-side comparison of the result of multiple queries over different Aliases (2.1.3)
- highlighting unusually highly or lowly correlated pairs in texts by comparing the same relations to values in previously processed texts
- different weighting of query terms
- generation of a summary of a text with the aids of the calculated FMI and Cosine Similarity values and Distributional Semantics techniques
- finding relations between Noun Phrases specifically, such as "apple IS A fruit"

The full power of this tool can best be judged by members of the humanities and literary science areas. It should be tremendously helpful when used as an aid for different projects. As discussed in chapter 2.5, the relations calculated by the tool are written to file and the reader is encouraged to experiment with the data independently of the front end.

Appendix A

Acronyms

PMI Pointwise Mutual Information

JSON Javascript Object Notation

dragn Distant Reading And Graph Nodes

FMI Frequency Mutual Information

MVC Model-View-Controller

HTML Hypertext Markup Language

NP Noun Phrase

NLTK Natural Language Toolkit

POS Part-of-Speech

SPO Subject-Predicate-Object

SPOP Subject-Predicate-Object-Provenance

Appendix B

Code

B.1 Git repo

The code for the system developed as part of this thesis can be found at https://github.com/Madjura/dragn.

B.2 Documentation

The documentation for the system can be found at https://madjura.github.io/dragn/. Installation instructions are included as part of the documentation.

Bibliography

[Abn87] Steven P Abney. *The English noun phrase in its sentential aspect*. PhD thesis, Massachusetts Institute of Technology, 1987.

[Bou09] Gerlof Bouma. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40, 2009.

[CH90] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[Cro06] Douglas Crockford. The application/json media type for javascript object notation (json). 2006.

[Cyt] Cytoscape.js. http://js.cytoscape.org/. Accessed: 2017-08-28.

[FLH+16] Max Franz, Christian T. Lopes, Gerardo Huck, Yue Dong, Onur Sumer, and Gary D. Bader. Cytoscape.js: a graph theory library for visualisation and analysis. *Bioinformatics*, 32(2):309, 2016.

[Lan06] Thomas K Landauer. *Latent semantic analysis*. Wiley Online Library, 2006.

[MMS93] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[Mor13] Franco Moretti. *Distant Reading*. Konstanz University Press, 2013.

[Mor16] Franco Moretti. *Distant Reading*. Konstanz University Press, 2016.

[NB14] Vít Nováček and Gully APC Burns. Skimmr: Facilitating knowledge discovery in life sciences by machine-aided skim reading. *PeerJ*, 2:e483, 2014.

[PGB] Project Gutenberg, The King James Version of the Bible. https://www.gutenberg.org/ebooks/10. Accessed: 2017-08-23.

[RJ09] Gabriel Recchia and Michael N Jones. More data trumps smarter algorithms: Comparing pointwise mutual information with latent semantic analysis. *Behavior research methods*, 41(3):647–656, 2009.

[RKA12] Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi. Semantic cosine similarity. In *The 7th International Student Conference on Advanced Science and Technology ICAST*, 2012.

[Sin01] Amit Singhal. Modern information retrieval: A brief overview. 2001.

[Tre] Penn Treebank tagset. https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html. Accessed: 2017-10-02.

[Zad65] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kentlich gemacht.
Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum

Unterschrift