KTH Computer Science
and Communication

Subtopic extraction using graph-based methods

Delämnesextraktion med grafbaserade metoder

MARKUS ÖSTBERG

markusos@kth.se

October 22, 2013

Degree Project in Computer Science and Engineering, Second Level
Supervisor: Jussi Karlgren (jussi@kth.se)
Examiner: Jens Lagergren (jensl@kth.se)
Project provider: Findwise AB, Simon Stenström (simon.stenstrom@findwise.com)

Abstract

This report presents a subtopic extraction method using search engine technology, combined with graph centrality ranking of topic candidates. The implemented system uses Wikipedia as a knowledge base to extract and rank topic candidates. Multiple methods of topic extraction are evaluated. Primary methods based on search engine technology are explored. The use of a state-of-the art annotation system of Wikipedia concepts for topic extraction is also explored as comparison to previous work. In this project, the system DBpedia spotlight is used for this comparison. It is also evaluated how graph centrality algorithms can be used to improve the results from the system. This includes using algorithms like PageRank, Degree centrality, Closeness centrality and Betweenness centrality to rank the extracted topic candidates. A good ranking is achieved when relevant topic candidates are given a higher rank than less relevant topic candidates. A topic candidate is considered relevant if it is relevant to the given input data text.

The system is implemented using the open source search engine ElasticSearch. The system is evaluated on abstracts from the Simple English Wikipedia. The results show that the system in 85% of the evaluated test cases finds the expected topic of the text in the top 10 ranked topic candidates. A survey was conducted where participants were asked to classify results from the system based on how well they matched a given text. The data from the survey shows that, in most cases, graph centrality ranking does improve the results noticeably, compared to only using the Term Frequency and Inverse Document Frequency (TF-IDF) ranking given by the search engine.

Referat

Delämnesextraktion med grafbaserade metoder

Denna rapport presenterar ett system för delämnesextraktion som använder sökmotorteknologi i kombination med grafcentralitet för att rangordna möjliga ämneskandidater. Det implementerade systemet använder Wikipedia som en kunskapsbas för att extrahera och rangordna möjliga ämnen. Flera metoder för ämnesextraktion undersöks. Primärt utforskas ämnesextraktion baserad på sökmotorteknologi. Som jämförelse med tidigare arbeten undersöks även hur ett ”state-of-the-art”-system för annotering av Wikipedia-koncept kan användas för ämnesextraktion. I detta projekt används systemet DBpedia spotlight för denna jämförelse. I rapporten utvärderas även hur grafcentralitet kan användas för att förbättra resultaten. Detta inkluderar användandet av algoritmer som PageRank, Degree-centralitet, Closeness-centralitet och Betweenness-centralitet för att rangordna de extraherade ämneskandidaterna. En bra rangordning motsvaras av att de relevanta ämneskandidaterna ges en högre rank än mindre relevanta resultat. En ämneskandidat är relevant om den kan anses vara ett representativt ämne för den givna indata-texten.

Systemet är implementerat med ”open source”-sökmotorn ElasticSearch. För att utvärdera systemet användes introduktionsstyckena från Simple English Wikipedia-artiklar. Resultatet visar att i 85% av de utvärderade testfallen finns det förväntade ämnet med bland de topp-10-rankade ämneskandidaterna. En undersökning genomfördes där deltagarna ombads klassifiera resultat från systemet baserat på hur väl de matchade en given text. Data från undersökingen visar att grafcentralitet, i de flesta fall, förbättrar resultaten märkbart, jämfört med att bara använda term-frekvens- och invers dokument-frekvens- (TF-IDF) rankingen given av sökmotorn.

Preface

This is a master's thesis in Computer Science at the School of Computer Science and Engineering at the Royal Institute of Technology (KTH). The project was provided by Findwise AB, a company that specializes in search solutions, and most of the work was conducted in their office in Stockholm. I would like to thank Findwise, and especially my project provider Simon Stenström, for all the support and always making me feel welcome. I also want to thank my supervisor at KTH, Jussi Karlgren, for all his valuable insights and advice during the project.

Contents

Chapter 1

Introduction

Maintaining correct keywords and other useful metadata is especially significant when searching through large archives of data. A survey on time spent on different tasks reveals that the average knowledge worker, i.e. people who produce or use information in their work, spends around 18 hours a week searching and gathering information. Over 1/6 of that time is spent searching but not finding what they are looking for [Feldman et al., 2005]. This means that improving findability of documents has the potential to improve the efficiency of information workers and thereby make companies more profitable.

Extracting topics and other metadata from texts is important in order to quickly gain an overview of a text. Today there exists a variety of different methods to extract topics. Subtopic extraction is a subcategory to topic extraction that is not extensively explored. The purpose of subtopic extraction is to not only determine the main topic that runs throughout the entire text, but also to distinguish what different parts of the text is about. The purpose of this thesis project is to examine the possibility of using a graph-based method to extract subtopics from text documents.

One approach to subtopic extraction is to use a graph to represent relationships between concepts. These relationships can be used to find appropriate candidate subtopics from the text and to rank how suitable the candidates are as subtopics of the text. The use of a graph-based method is not the conventional way of performing topic extraction. It is therefore of interest to explore and evaluate how well this works in practice, and if it is possible to use concept relationships for subtopic extraction.

1.1 Outline of This Report

- Chapter 1 introduces the problem and describes the projects goals. The system requirements as well as some use cases for the system are also described, along with a short introduction to the project provider.
- Chapter 2 contains the background theory for the project. Key concepts that are important in the project are introduced and different techniques for topic extraction are presented. The background chapter also contains a brief overview of what Wikipedia is, and how the encyclopedia has been used in topic extraction research. The chapter also explains the basics of statistical and graph-based topic ranking methods used in previous research.
- Chapter 3 presents the proposed system and describes the ideas used to try to solve the problem of subtopic extraction. The chapter further explains how the system's relationship graph is used to rank topic candidates in the system. It also describes the workflow of the system and how a user interacts with it.
- Chapter 4 describes in more detail how the system is implemented and the different tools used to do so. The chapter describes how the pre-processing of the Wikipedia data is conducted. It also explains how the data is used to build the system relationship graph structure. Implementation details of the different subtopic candidate extractors and the various graph ranking algorithms used in the project are also provided.
- Chapter 5 describes how the system is evaluated and what data is used to do so. It also describes the evaluation methods and the measurements utilized in the evaluation process.
- Chapter 6 presents the results from the evaluations of the system. The chapter shows how the system performance is affected by the use of different ranking methods.
- Chapter 7 contains discussion of the results and conclusions, as well as some suggestions for future work.

1.2 About Findwise

Findwise AB is an IT consultant company specializing in search driven solutions. Findwise focuses on improving what they call findability of their customers' data. Improving findability means to make information easy to find and quickly accessible. People do not want to spend hours searching for information; they want to find it as fast as possible. Efficient enterprise search is a critical part of many businesses, but many companies lack the in-house knowledge on how to implement it. This is where Findwise comes in and provides their expertise.

Findwise's interest in topic extraction lies in the possibility of using it to improve findability of their customers data. A topic extraction system could be used not only to automatically annotate topics of texts, but also as a tool for editors working for their client companies to make manual topic annotation faster. This project aims to evaluate the possibility of using automated subtopic extraction as a step towards better findability.

1.3 Goal of the Project

The goal of this thesis project is to evaluate if it is possible to use a graph-based method for subtopic extraction. This project aims to answer the following questions:

1. Is it possible to use search engine technology to identify topic candidates for texts of varying length?
2. Is a graph ranking method relevant and useful to rank possible topics?
3. Can the system, given a text containing several different subtopics, correctly identify these?
4. Is the result from the system relevant and useful enough to automatically create new topic metadata that could be utilized to improve findability in a search system?
5. Is the result from the system relevant and useful enough to use as a tool for editors to speed up manual annotation of subtopic metadata?

1.4 System Requirements Specification

Input:

- Plain text for subtopic analysis.
- Algorithm parameters for tweaking the performance and selecting ranking method.

Output:

- Input text segmented into smaller sections.
- For each section; a list of likely topics, ordered by the ranking given by the system.

1.5 Use Cases

- The system could be used as a tool to provide better metadata for search indexing. Indexing on subtopics could be done on a paragraph level, making it possible to present relevant paragraphs of text in the search results; e.g. if the user searches for the word ”tree”, the system could return all paragraphs annotated and indexed with the subtopic ”tree”.
- The system could be used as a tool for researchers who browse large amounts of texts. The system could then provide topics for each section, making it easier to determine if it is worth reading the entire section.
- The system could be used as an annotation tool for editors. The tool could give suggestions on topics that it perceives as relevant in the text, where the editor would decide which topic keyword suggestions to annotate the text with.

Chapter 2

Background

This chapter contains the background knowledge relevant for this project. Key concepts that are important in the project are introduced and different techniques for topic extraction are presented. The background chapter also contains a brief overview of what Wikipedia is, and how the encyclopedia has been used in topic extraction research. The chapter also explains the basics of statistical and graph-based topic ranking methods used in previous research.

2.1 What is a Topic?

A topic is the underlying subject of a text, speech, or any other form of communication. Topics tell us what is being discussed. By knowing the topic it is easier for us to understand the context of the communication. Knowledge of document topics has been known to be important throughout history. From old library collections built hundreds of years ago to the modern age and the creation of the Internet, people have spent massive amounts of time organizing documents. Document topics have always been important in this field, but with the explosion of new documents it is now impossible to manually categorize all documents. Automated topic extraction strives to solve this problem by making it possible for computers to determine the topic of a document. [Medelyan et al., 2008]

In more modern applications, topics are used not only as keywords when building search indexes of documents, but also for other information retrieval tasks like document classification and clustering. Topics are also useful when categorizing data, because the topics themselves often define possible categories for texts. To human readers, topics are still used to quickly get an overview of a document and to determine if it is worth reading or not.

In this report, *topic* will represent a concept that describes the subject of a text. It could be an object, like a Tree, a name of a person, like Stockholm, or a place, or an abstract concept, like a data structure. The term *subtopic* will be used to

represent a topic limited to a shorter section of the text.

2.1.1 Subtopic

A *subtopic* is a smaller subset of the main topic. In a longer text each paragraph usually contains its own subtopic. These subtopics are often related to the main topic that runs through the entire text, but depending on the type of text this is not always the case. Through extraction of subtopics we get a better understanding of what the text is about by distinguishing the topics of each section of the text. Extracted subtopics can be used to improve humans' understanding of the subjects of a text, create better search indexes, and summarize texts.

2.2 Topic Extraction

Topic extraction is the process of determining the topics of a text. Most research in topic extraction falls in one of two categories: latent topic models and ontology based methods.

2.2.1 Latent Topic Models

Latent topic models analyze the given texts and extract possible topics without using any external information. One of the currently common methods is Latent Dirichlet Allocation (LDA), which is a statistical model used to determine the topic of a text. It is one of the simpler statistical topic models. LDA is a further development of another topic model, probabilistic latent semantic analysis (pLSA). One of the base assumptions in both of these models is that all documents are made up of a mixture of several different topics. To utilize the model, a large set of documents is needed to build up the statistical data used for classification. Each document that needs to be classified is then compared to the model to find which combination of topics, according to the statistical model, fits best to the document. [Blei, 2012, Blei et al., 2003]

2.2.2 Ontology Based Methods

Ontology based extraction methods make use of large sources of structured data. An ontology is a formal representation of knowledge, often as concepts with relationships between them, e.g. dictionaries like WordNet or encyclopedias like Wikipedia. Generally an ontology is used to connect words that represent the same concept. This could be used to identify the words that belong to the same possible topic concept. Different methods are then used to evaluate which topic concept fits best on the input text. [Chahine et al., 2008]

One of the more used ontologies in topic extraction research is WordNet. WordNet is a large lexical database where words are grouped together in what are called synsets, or cognitive synonyms. Each synset represents a distinguished concept and words are linked to all other synsets in which these words appear. For example the synset for the meronym[1] of the word "Tree" contains, among others, the words: "stump", "crown", "treetop", "limb", "branch" and "trunk".

Both WordNet's and Wikipedia's article structures have been used in different ways to identify topics and as complements to topic models. See Section 2.7 for how Wikipedia has been used as an ontology for topic extraction in previous research.

2.3 Wikipedia

2.3.1 What is Wikipedia?

Wikipedia is one of the largest sources of encyclopedic knowledge in the world. It is also one of the largest websites on the Internet today with approximately 470 million unique visitors every month. The English edition of Wikipedia contains over 4 million articles and it continues to grow every day. Today there exists 285 different language editions of Wikipedia; 25 editions have more than 200,000 articles. [Wikimedia Foundation, 2013]

Wikipedia Article

A Wikipedia article is an encyclopedic entry that identifies and explains "a notable encyclopedic topic" [Wikimedia Foundation, 2013]. Each article contains a summary of the article's topic, links to related articles on Wikipedia and a list of reliable sources used for that specific article. Each article also has a title that describes its topic and makes the it distinguishable from other Wikipedia articles. This means that each title in Wikipedia is unique to only one article.

The fact that each article represents and explains a certain topic concept makes it ideal to use as a resource for topic extraction. Wikipedia has, for the last couple of years, been the focus of several research projects where the encyclopedic information and the articles' link structure have been used in a variety of different methods to improve topic extraction. Some of these projects are described in Section 2.7.

Linking

One of the biggest differences between Wikipedia and other encyclopedias is that important concepts in the Wikipedia articles are linked to the articles describing those concepts, whereas other encyclopedias are not. The Wikipedia links on each

[1] A meronym is a word that describes a part of the concept another word describes.

article page represent, in most cases, the major connections from one concept to the concept of another article. Terms that are unrelated or do not improve the understanding of the concept are in general not linked, e.g. the Wikipedia article 'Tree' that links to, among many other, the concepts: "Perennial plant", "Root", "Forest", "Bark" and "Botany". [Wikimedia Foundation, 2013]

Redirects

Redirects are a special type of pages. They are used to give articles an alternative title. For example, the word "Macrophanerophyte", meaning a wooden plant between 30 to 50 meters tall, redirects to the more common name, "Tree". Redirects are also used for plural forms, other closely related words, alternative spellings, common misspellings and many other reasons. [Wikimedia Foundation, 2013]

Wikipedia Namespace

Wikipedia uses 26 different namespaces to organize its data. Namespaces are structured by defined prefixes in the article title. Articles in the same namespace contain the same prefix in their title. The main namespace is the encyclopedic article namespace. All articles without a prefix belong to this namespace. Other common namespaces is the Category namespace, referenced to with the prefix "Category:". The category namespace consists of pages containing lists of articles that belong to the category and links to subcategories. Other namespaces include "Wikipedia:", which is used to store content about Wikipedia, "File:", which is used to store images, sounds and other files and "Template:", which is used to store article templates. [Wikimedia Foundation, 2013]

2.3.2 DBpedia

DBpedia is a community project created to extract all the structured information from Wikipedia. The project strives to make it possible to pose advanced queries on all the structured data from Wikipedia. Part of the project's goal is to make it easier to study new interesting ways of using Wikipedia data, to improve both the encyclopedia itself and other research areas. [DBpedia Project, 2013]

DBpedia provides, among other things, structured files with the data sets of many of the Wikipedia language versions. These data sets contain different parts of Wikipedia's encyclopedic data. For this project, the data set containing the Wikipedia page links is extra interesting. This data set contains a representation of all internal links between unique Wikipedia articles.

2.4 Disambiguation of Words

When using an ontology-based method for topic extraction, an important task is to extract and match concepts in the text with concepts in the ontology. The simplest method for extracting Wikipedia concepts from a text is to match each sequence of words to a list of all Wikipedia concepts. The problem with this is disambiguation of words. This means that the same word can mean different things depending on the context it occurs in. One example of this is the word "Tree". It usually means a tall woody plant, but it could also refer to a data structure, a last name, a river in Canada or a number of other things.

2.4.1 DBpedia Spotlight

A variety of tools exist for detecting and matching concepts in a text to corresponding Wikipedia concepts. One of the more configurable tools is *DBpedia Spotlight*, an open source tool for automatic identification and annotation of Wikipedia concepts in any text document [Mendes et al., 2011].

The disambiguation of words makes extraction of Wikipedia concepts from text a complex task. Previous research shows that a random method used to solve disambiguation was correct less than 25% of the time. In comparison, state-of-the-art systems like DBpedia Spotlight achieve an accuracy of around 80% [Mendes et al., 2011].

The method DBpedia Spotlight uses to detect and match concepts consists of several steps. First, the system identifies keywords in the input text. This is done by matching words in the input text with labels that correspond to one or more Wikipedia article titles. When a disambiguation is found, the algorithm selects the most likely articles in the candidate selection process. The final candidate is selected by looking at the context of the text. A ranking system using a measure called Inverse Candidate Frequency (ICF) is used with the Term Frequency (TF) measure to rank the candidates. ICF is a method to weight words based on how well a certain word can distinguish between candidate concepts for a disambiguated word. The highest ranking candidate is finally selected by the algorithm for annotation. [Mendes et al., 2011]

2.5 Graph-Based Ranking Methods

Some previous research explores the use of ontologies in combination with graph-based ranking methods to extract topics (See Section 2.7). In this section, the theory behind graph centrality is presented and several graph centrality measures are explained.

Graph centrality measures are used to determine how important a node is in a graph. They are often applied on social network graphs to determine how influential a node, or person, is in the network. Graph centrality is also used in web search where the nodes represent web pages and the edges represent the links between pages. Researchers have also experimented with graph centrality to determine central concepts in texts, and some progress has been made using graph centrality in topic extraction systems.

Depending on the purpose and the criteria for what is important in the graph network, different centrality measures are used. This section presents a number of graph centrality measures and explains how they can be used to rank nodes in graph structures. Figures 2.1, 2.2 and 2.3 show how different graph centrality measures rank the same graph.[2]

[2]Image source: `http://en.wikipedia.org/wiki/File:Centrality.svg`

2.5.1 Degree Centrality

In degree centrality, a node is important if it has a high degree, i.e. if it has many edges connected to and from it. The key idea is that a node that is directly connected to many other nodes is more important than a node connected to only a few nodes.

Definition 1 *Degree centrality:*

$$C(u) = deg(u)$$

Where u is a node in the graph, and deg(u) is the number of edges to and from node u.

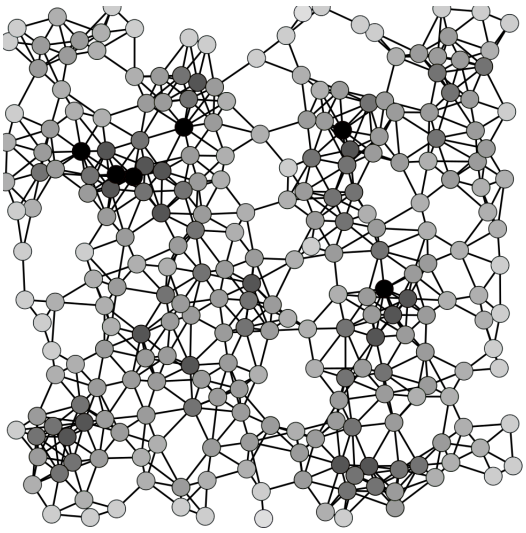

Figure 2.1. Illustration of nodes centrality according to a Degree centrality ranking. Dark gray nodes are more central in the graph than light grey nodes.

2.5.2 Closeness Centrality

In closeness centrality, a node is important in the network if it is close to all other nodes in the network. This means that the most important node is the one whose sum of the shortest paths to all other nodes is smallest. [Hulpus et al., 2013]

The closeness measure in its original form requires that the graph does not have any disconnected components. This is because the distance between two nodes in different disconnected components is seen as infinite, which results in an infinite closeness for all nodes if disconnected components exist in the graph. To work around this problem a modified closeness measure is used. The modified measure does not take into account the distance between nodes in disconnected components.

Definition 2 *Closeness centrality:*

$$C(u) = \sum_{v \in V(G)} \frac{1}{\delta(u,v)}$$

Where u is a node in the graph G, and $\delta(u,v)$ is the distance from node u to v, i.e., the minimum length of any path connecting u to v.

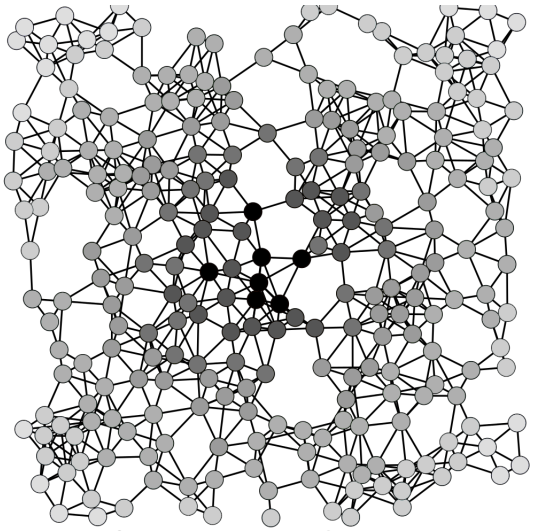

Figure 2.2. Illustration of nodes centrality according to a Closeness centrality ranking. Dark grey nodes are more central in the graph than light grey nodes.

2.5.3 Betweenness Centrality

In betweenness centrality, a node is important if it is a common part of the shortest paths between many pairs of other nodes. The most important node is the node that connects most other nodes as the shortest path in the network [Hulpus et al., 2013].

Betweenness centrality is a complex measure to compute. Until a few years ago, the complexity of the best known algorithms required $\Theta(n^3)$ time, where n is the number of nodes in the graph. Newer research shows that this can be reduced to $\mathcal{O}(nm)$. This makes it possible to use betweenness centrality on larger graphs. [Brandes, 2001]

Definition 3 *Betweenness centrality:*

$$C(u) = \sum_{s \neq u \neq t \in V(G)} \frac{\sigma_{st}(u)}{\sigma_{st}}$$

Where u, s and t are nodes in the graph G. σ_{st} is the number of shortest paths between nodes s and t, and $\sigma_{st}(u)$ is the number of shortest paths between s and t that passes through node u. [Brandes, 2001]

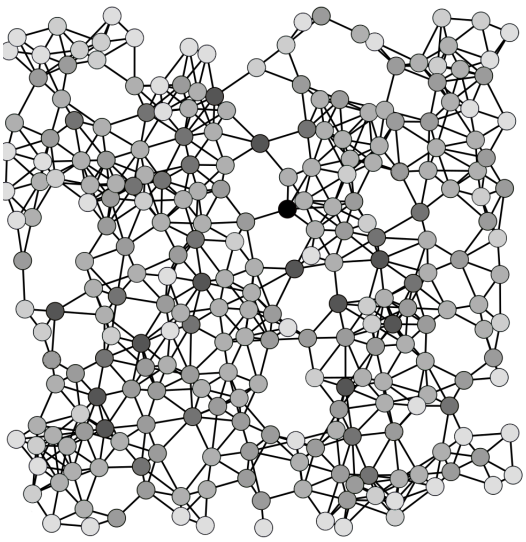

Figure 2.3. Illustration of nodes centrality according to a Betweenness centrality ranking. Dark grey nodes are more central in the graph than light grey nodes.

2.5.4 PageRank

PageRank is a web page ranking algorithm developed at Stanford University. The basic idea behind PageRank is that a web page is as valuable as the pages that link to it. A typical analogy for this is that a link from one page to another essentially can be seen as a vote being cast by one page onto another. Pages that are voted on by many other pages receive a higher rank. The page is therefore ranked by the value of its backlinks, i.e., the links from other pages to that page. If a page is backlinked by many highly ranked pages, the page is given a higher ranking than a page with low valued backlinks. [Page et al., 1999]

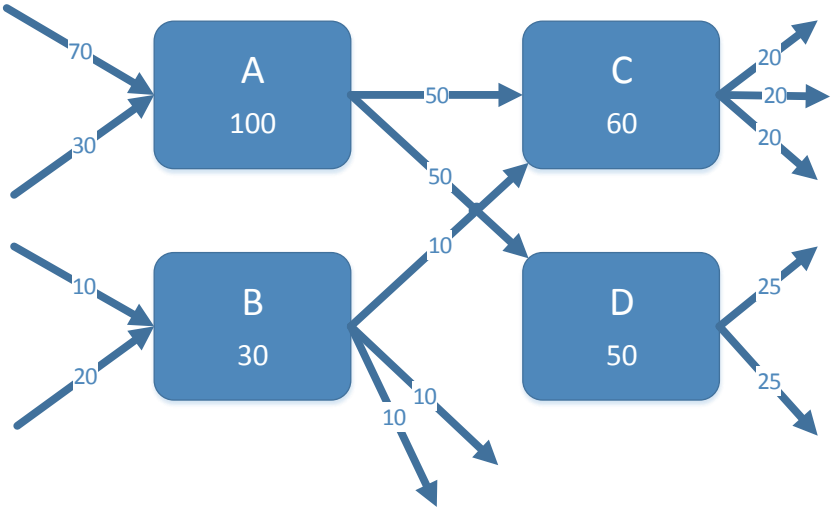

Figure 2.4. Simple illustration of the PageRank calculations

PageRank also takes into account how many out links a backlinked page has when assigning the ranking. E.g. if a page with the rank 100 links to two different pages, each of those pages receives a 50 point score increase. The total ranking is the sum of all the scores received from all the backlinked pages. See Figure 2.4.

Definition 4 *A simplified version of the PageRank definition: R(u) is the rank of page u. B_u is the set of pages that links to u, the backlinks of u. N_u is the number of links from the page u and c is the normalization factor. [Page et al., 1999].*

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

Random Surfer Model

PageRank can also be seen as a *random surfer model*. The rank corresponds to how likely it is that a random surfer ends up on a certain page. Pages with a higher PageRank are more likely to be visited than pages with a low rank. In the simplified model, a surfer that ends up in a loop of pages only connected to each other would continue to just visit the pages in the loop. This kind of loop is called a rank sink, because it drains the rank from all of its backlinks that are not in the loop. To solve this problem and to make the random surfer model better, a rank source vector E(u) is inserted into the model. The rank source could be used to weigh nodes differently for the ranking, but in most applications, it is defined to be uniform over all nodes. In the random surfer model, this definition corresponds to the surfer randomly jumping between any pages on the web. [Page et al., 1999]

Definition 5 *PageRank definition: E(u) is the rank source for page u and R(u) is the rank of page u. B_u is the set of pages that links to u, the backlinks of u. N_u is the number of links from the page u and c is the normalization factor. [Page et al., 1999].*

$$R(u) = c * E(u) + c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

2.6 Statistical Ranking Methods

This section presents a method used in previous work to rank possible topic candidates. The ranking determines if the topic candidate is a good match to the document text and gives a metric that can be compared to other topic candidates.

2.6.1 TF-IDF

Term Frequency–Inverse Document Frequency (TF-IDF) is a classical metric in information retrieval. It is mainly used as a measure of a term's importance in a document. *Term Frequency* (TF) is the number of occurrences of a specific term in a text document, sometimes normalized by dividing by the frequency of the most common term in the document or the size of the document. *Inverse Document Frequency* (IDF) is calculated by dividing the total number of documents by the number of documents containing the specific term. By multiplying the term frequency with the inverse document frequency, we get the TF-IDF metric. [Manning et al., 2008]

Definition 6 *The $TF \times IDF$ ranking for a term T in a document D is calculated by:*

$$TF \times IDF = \frac{freq(T, D)}{size(D)} \times log \frac{N}{count(T)}$$

Where freq(T,D) is the frequency of the word T in document D, size(D) is the number of words in document D, count(T) is the number of documents containing the term T and N is the total number of documents. [Manning et al., 2008]

Search engines often use TF-IDF to rank how relevant different documents are to a search query. This is normally done by calculating a normalized sum of the TF-IDF value for each term in the search engine query.

2.7 Wikipedia Topic Extraction Methods

The development of Wikipedia has reached a point where it provides good coverage of most subjects. This makes it useful for general topic extraction in a way that was not possible ten years ago. The good coverage, in combination with Wikipedia's structured open data, has made it subject to several research projects for the last couple of years. This section describes some of this work and distinguishes their differences in methodology.

2.7.1 Wikipedia Graph Ranking

The Wikipedia graph structure is created from Wikipedia's articles and link data. Each article in Wikipedia is represented as a node in the graph, and the links between articles are represented by directed edges between the nodes of the graph. Varieties of this graph have been used in a number of different research projects, for both topic extraction and other purposes. Methods that use the Wikipedia graph structure to extract topics often utilize some type of graph ranking algorithm. These algorithms are often inspired by the PageRank algorithm. Some recent research projects have worked with ranking of concepts in the Wikipedia graph [Coursey et al., 2009, Zhou et al., 2009]. The methods used in these projects are further developments and adaptations of the ideas behind PageRank.

Coursey et al. [2009] extends the PageRank algorithm by introducing a bias towards nodes matched in the input text. Zhou et al. [2009] WikiRank algorithm uses the assumption that a concept is important if it occurs in the input document with high frequency and is heavily linked to other significant concepts in the Wikipedia graph. In their research, they use a predecessor to DBpedia Spotlight called *Wikify!* to identify Wikipedia concepts in the input texts. Wikify! uses a method called *Keyphraseness* to annotate Wikipedia concepts in any input text [Mihalcea and Csomai, 2007].

Hulpus et al. [2013] explores how graph centrality measures can be used on the Wikipedia graph structure to identify topic labels, given the core topic concepts. The work is part of a larger system called *Canopy*, which makes use of LDA for extraction of core concepts in the input text. The system also contains a word-sense disambiguation component used to clarify the core concepts before they are given to the topic labeling system. The researchers present a novel method using a graph centrality algorithm that is biased towards the identified core topic concepts. This is used to find other concepts central in a sub graph of the Wikipedia graph. The idea is that central concepts in the sub graph describe the set of identified core concepts from the text. The central concept is therefore a reasonable label for the topic that the core concepts represent.

2.7.2 Statistical Ranking

Statistical ranking methods have also been explored. In these methods, identified topic candidates are often ranked based on word frequencies, Wikipedia graph link counts and other quantifiable measures.

Medelyan et al. [2008] developed a method that uses a number of statistical measures to rank possible topic candidates. Their system selects Wikipedia articles that match keywords in the input text as candidate topics. The candidates are thereafter ranked using TF-IDF, node connectivity to other identified nodes in the Wikipedia graph and a few other parameters. When compared to topic assignments done by human annotators, the consistency between the system's annotations and those of the humans were similar to the consistency between different human annotators. This means that a topic assignment made by the system is comparable with that of a human indexer.

Schonhofen [2006] evaluates a statistical method that uses Wikipedia's categories as topic labels. The method matches words in the input text to stemmed titles of Wikipedia articles. Each stemmed title is then linked to one or several Wikipedia articles. The Wikipedia categories belonging to all the found articles are then extracted as possible topics for the input text. In each step weights are applied to the concepts. The weights of the previous step are used to calculate the new weights. The weights in each step are also normalized by different measures, e.g. number of categories that word W occurs in, number of titles containing a word W, number of articles pointed to by title T, etc. The final weights, for the categories that are possible topics, are therefore dependent on weights from all the words and articles that linked to that category. In 86% of the cases, when running the system on Wikipedia articles, the system returned one or more of the articles' actual Wikipedia categories in the top 20 results. In 48% of the cases, the highest ranked category that the system returned was a match to one of the article's categories.

Chapter 3

Proposed System

This chapter introduces a proposed system for subtopic extraction. The key ideas on how the system is envisioned are described and the system workflow is presented.

3.1 Key Ideas

3.1.1 Concept Relationships

The key idea used in this project is that concepts in languages have relationships to other concepts. For this project, a concept is defined as anything from a physical object, like a tree in the forest, to something as abstract as a tree data structure. Language concepts can be used to describe other concepts. This means that a relationship exists between these concepts. For example, the concept ”Tree (plant)” can be described with the concepts ”Root”, ”Trunk”, ”Branch”, ”Plant” and so on. This indicates that there is a relationship between the concept ”Tree (plant)” and all these concepts.

3.1.2 Knowledge Graph

If a set of all the different concepts in a language and all the relationships between them are collected, it is possible to construct a graph structure that describes the language. The nodes in the relationship graph correspond to the concepts and the edges between the concepts represent the relationships between different concepts. This graph structure represents the knowledge graph used in the proposed system. The knowledge stored in this graph needs to be extensive enough to cover a large portion of the language for it to be useful in a topic extraction system.

In this project, the knowledge graph is constructed from the English Wikipedia. The Wikipedia Graph is not a perfect representation of the concepts and their relationships in the English language, but the scope of this project does not allow

for the construction of a complete knowledge graph. Wikipedia articles make out the nodes in the knowledge graph, and the links between articles are represented with directed edges.

3.1.3 Extracting Topics

Another key idea is that it is possible to extract subtopics by examining the relationships between the concepts in a text. The relationships between different concepts are looked up in the knowledge graph. The topic extraction is then done by finding concepts that are central in the graph, which is constructed from the concepts in the text. If a text contains the concepts ”Root”, ”Trunk”, ”Branch” and ”Plant” and all of these have relationships to the concept ”Tree (plant)”, there should be a certain probability that ”Tree (plant)” is a good subtopic for that text.

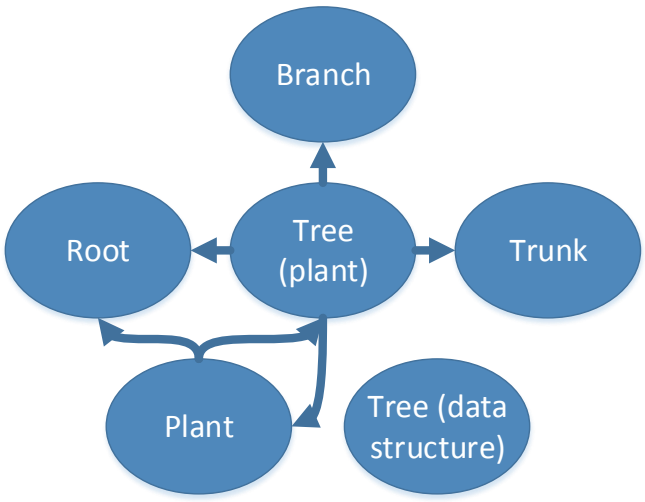

Figure 3.1. Illustration of how the text is matched to the concepts of the knowledge graph, creating a candidate relationship graph.

Figure 3.1 shows how a text string could be matched with nodes in the relationship graph and that the concepts ”Tree (plant)”, ”Plant” and ”Tree (data structure)” are identified. The concept ”Tree (plant)” is connected to the concepts ”Root”, ”Trunk”, ”Branch” and ”Plant”. Because ”Tree (plant)” has more common neighbors than the other concepts extracted from the text, and is therefore more central in the candidate relationship graph. As a result, we can assume that the text, to a certain extent, is more about the plant ”Tree” rather than the data structure ”Tree” or the broader concept ”Plant”.

As seen in Figure 3.1, "Tree (data structure)" is not connected to any other concept in the graph, and is therefore probably not a relevant topic for the text. The concept "Plant" is related to some of the other extracted nodes and thus might also be a relevant topic for the text. The hope is that the system should be able to handle disambiguation of words as well as broader topics, and rank them accordingly. The concept "Tree (data structure)" should get a low rank by the system while "Plant" and "Tree (plant)" should get higher ranks.

As opposed to previous research, the algorithm uses only the selected candidates and their relationships to rank the result. The works by Coursey and Mihalcea [2009] and Zhou et al. [2009] analyze concept relationships in the entire Wikipedia graph to rank topic concepts. In this project, the possibility to use graph centrality to rank the candidate topics in the smaller candidate relationship graph is explored. Other previous research, such as Schonhofen [2006], uses Wikipedia categories as topic labels. In this project, the focus lies on labeling shorter sections of text with subtopics. Therefore, the system utilizes the articles titles as labels. This gives the system a possibility to give a much more precise subtopic label.

3.2 System Workflow

The input to the system is a plain text file with paragraph structure. It should also be possible to give the system an XML-structure with short texts and annotated topics for evaluation purposes. The system should also take a number of parameters to define which candidate selection method and ranking algorithm is used. Figure 3.2 illustrates an overview of the system workflow.

The first step is for the system to read the input text file and split the input into sections. A section could in this case be either a paragraph in the text, a specified number of consecutive words (e.g. 20 words in a row) or consecutive sentences (e.g. 5 sentences in a row).

The second step in the system workflow is to extract and select topic candidates from the text sections. A topic candidate is a concept from the relationship graph that is identified in the text section. This can be implemented in different ways. Candidate selection is the process of matching concepts, from the input text to the knowledge graph, and selecting the best matches as candidate topics.

In this project, two different methods of candidate selection are explored. The first method is a novel approach using a search engine to find relevant topic candidates, and the second is a well-proven method based on the Wikipedia annotation system DBpedia Spotlight (See Section 4.3.2 for more details). A Wikipedia annotation system similar to DBpedia Spotlight was used in research by Zhou et al. [2009]. DBpedia Spotlight is therefore also used as a baseline to evaluate the usefulness of the novel approach using a search engine for candidate selection, as this is part of research question 1 of this project.

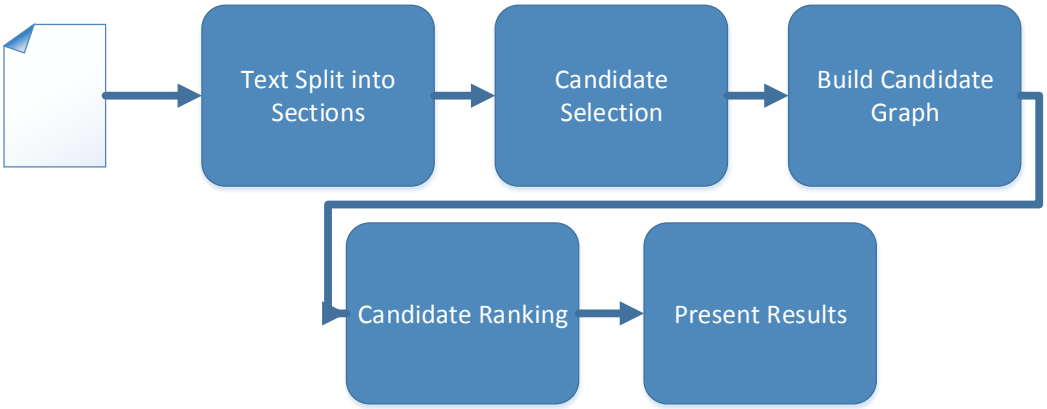

Figure 3.2. System workflow

From the identified topic candidates, a smaller candidate relationship graph is constructed. The graph's main nodes are the concepts identified as topic candidates, but it might also be extended to include neighbors to those nodes from the global concept relationship graph.

The system then uses the candidate relationship graph to rank the topic candidates by using a ranking algorithm. The ranking algorithms use different graph-based centrality measures to determine which concepts are the most central in the graph. The hypothesis is that a concept that is central in the candidate relationship graph also is a central concept in the input text section that the graph corresponds to. Research question 2 of this project is to determine if the graph centrality can be used to improve the results.

In the last step, the system builds a presentation HTML file containing the input text split into smaller sections, and for each section, a list of the top ranked topic candidates in order of ranking score.

The system is built in modules so that the topic candidate selection algorithms and the ranking algorithms are easily interchangeable. By modifying or creating new modules, it is also easy to change how the result is presented as well as how the text is split into sections.

Chapter 4

Implementation

This chapter contains details about the setup, data pre-processing and implementation of the system. It also contains an example of how the topic extraction works and the results from a sample topic extraction.

4.1 Setup

This section is about the environment setup in the project. It presents different software and libraries used in the project and summarizes their main functionality in the project.

4.1.1 MongoDB

MongoDB is a high performance, open source, NoSQL database. In contrast to that in traditional databases, data in MongoDB is saved as documents. Each document in the database is represented as a JSON structure. MongoDB has the capability to index on any document attribute to improve the speed of common database queries.

4.1.2 ElasticSearch

ElasticSearch is an open source search engine built on top of Apache Lucene software. It is designed to be fast, stable and scalable. ElasticSearch stores data in the JSON format, which makes it easy to index data stored in MongoDB.

The following sections describe how ElasticSearch works and how it is set up in this project.

Queries

ElasticSearch handles many different types of queries. The basic query is a term query, where the system is given a text string and one or more fields to search in. Documents that contain the term in the specified field are ranked using TF-IDF, but it is also possible to boost results to improve their score. The top ranked search responses are then returned. The number of search responses returned is defined in the query, but by default the top 10 results are returned.

Boosting

Boosting in ElasticSearch can be done in two different ways, either during indexing or in each query. Search time boosting is done by adding boosts to different parts of the query. Either a specific word or all search hits in a certain field can be boosted. Index time boosting can either be per document, or to boost fields or terms in the document. The index time boosting values are static while the search time boosting values can be changed depending on the query.

Stop Words

Stop words are words ignored by the search engine. When the search engine parses and analyzes the query, the stop words are removed and do not affect the query result. Because of the number of non typical documents indexed in this project, a large stop word list with over 600 stop words is used.[1] This assures that common words in the English language, but not in the indexed documents, are not given a skewed score by the ElasticSearch scoring algorithm.

4.2 Pre-Processing

This section explains the pre-processing of the Wikipedia graph data and how it is used as the topic knowledge in the system. All pre-processing steps use a MongoDB database as storage for processed documents.

4.2.1 Parsing the Wikipedia Link Structure

The knowledge graph used in the system is constructed from the Wikipedia link structure. The link structure is extracted from DBpedia, which provides XML structured files containing snapshots of all Wikipedia data. The "Wikipedia Pagelinks" file contains an XML list where each row corresponds to a link from one DBpedia resource to another. Each DBpedia resource is then labeled to correspond to an article in Wikipedia.

[1] List of stop words used: `http://www.nada.kth.se/~markusos/thesis/data/stopwords.txt`

The Wikipedia Pagelinks file is parsed line by line. The names of all article concepts are extracted, giving the names of both the article containing the link and the linked article. For each article, a JSON document is created containing the names of article concepts for all the outgoing links and the title of the document. See Listing 4.1 for the JSON structure. All documents are then saved in an MongoDB database for processing.

4.2.2 Merging Redirects

The next step in the pre-processing stage is to merge all redirect pages to their main articles. In most cases, a document with only one link is a redirect page. It is therefore assumed that all documents with only one link should be merged with the linked document. The algorithm then goes through the database and merges these documents. This is done by adding the title of the redirect page, the page with only one link, to the one page they link to. All redirects to a page are stored in a special array in the JSON structure.

Listing 4.1. JSON structure, corresponding to the Wikipedia article "Tree".

```
{
  "_id" : ObjectId("511cf9667e2b684578787db4"),
  "title" : "Tree",
  "redirectCount" : 9,
  "redirects" : ["Trees", "Sapling", ...],
  "linkCount" : 173,
  "links" : ["Perennial plant", "Woody plant", ...],
  "backLinkCount" : 3421,
}
```

4.2.3 Backlink Counting

Another important metric is the backlink count. A document that is linked to by many other documents is potentially more important than one that is linked to by just a few. To add the backlink count value to all the documents, a two-pass algorithm goes through all documents and first counts the number of links to each concept. In the second pass, each article is matched to the corresponding concept and the count is added to the document in the backLinkCount field (See Listing 4.1).

4.2.4 Indexing

When all the other pre-processing steps are completed, the processed documents are indexed into ElasticSearch. During this step, all documents saved in MongoDB are indexed into ElasticSearch. This makes it possible to use ElasticSearch for fast identification of topic candidates by sending search queries to the search engine.

4.3 Topic Extraction

This section explains how the different steps in the topic extraction workflow are implemented. Figure 4.1 shows the system workflow, and some details about the interchangeable components in the ElasticSearch candidate selection and the concept ranking step.

4.3.1 Reading Input Data

The first step of the topic extraction is to read the input data and split it into sections. Depending on the settings for how to split the data into sections, one of several methods is used. The default setting splits the text at paragraph breaks, where a blank line is left in the text. It is also possible to give the system an XML structure with evaluation as input.

4.3.2 Candidate Selection

During the candidate selection step, possible topic candidates are extracted from the split input text. A topic candidate could either be a concept mentioned in the text or a concept that is strongly connected to concepts mentioned in the text.

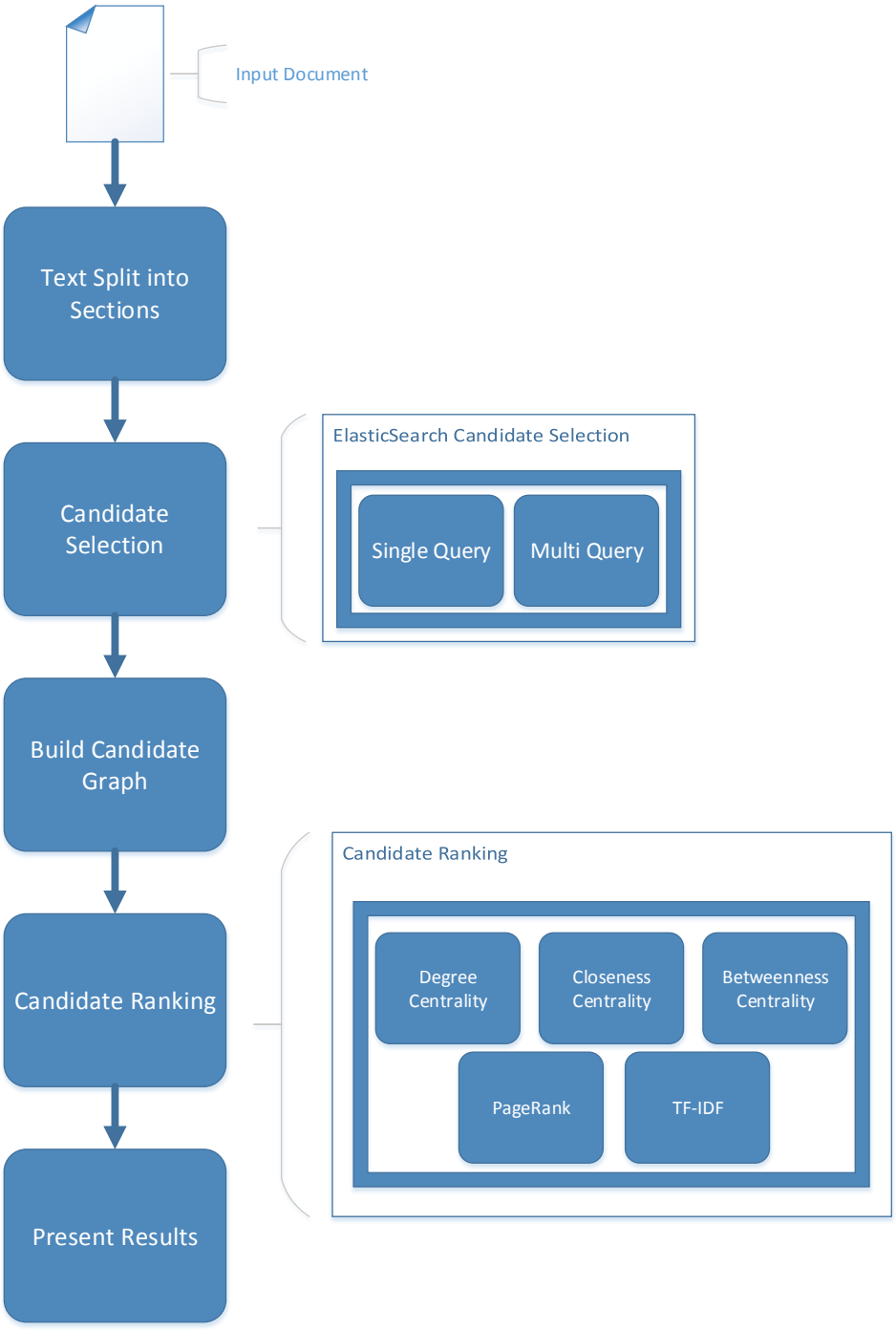

Figure 4.1. System workflow

ElasticSearch Candidate Selection

In this project, ElasticSearch is not used in the conventional way. Instead of indexing full text documents and using keywords to find matches, documents containing keywords are indexed and segments of full texts are used in the search queries. As a result, the system returns keyword-documents that best match the text segment. Because each document contains a topic and keywords that describe the topic, it is possible to use the search engine to extract topic candidates.

It is possible to use ElasticSearch for candidate selection in many different ways. How queries are used, and what data the queries are used on, are some of the potential ways to tweak the candidate selection. It is of interest to explore the impact on the result if the search is not done on the entire section, but instead split into multiple queries. In this project, some of these combinations are evaluated to determine a good practice. The focus is on the search method (i.e, what data is used for the search).

Search method:

- Single Search: The search is done as a single query on the entire section.
- Multi Search: The section is split up into multiple queries, one for each sentence in the section. All queries are then used to build a single candidate graph.

The search query is executed on data fields ”title”, ”links” and ”redirects” with lowering boosts of 0.1 on the fields ”title” and ”redirects”. This boost is chosen based on the idea that a article topic is relevant if many of the concepts that describe the topic are mentioned in the text. It is also important when the concept itself is mentioned, but it is the combination of many related concepts being mentioned that makes it more likely for the concept to be the topic of the text. Tests run on smaller evaluation sets show that the result using a different boost on the links and title fields improves the recall noticeably on that test set. To limit the scope of the project, all evaluations of the ElasticSearch candidate selection are run with this particular boost.

The ElasticSearch candidate selection step returns the top ranked matches from the search queries as the selected candidates. It is possible to set the number of search results returned from each search. It is to be evaluated how many search results are useful when performing topic extraction.

DBpedia Spotlight Candidate Selection

The DBpedia Spotlight service can also be used to extract topic candidates. The service returns a JSON object containing links to all DBpedia resources mentioned in the text. From these links it is possible to extract the corresponding Wikipedia resource. To extract the concept relationships between the found topic candidates, lookups are made in the MongoDB database structure of the knowledge graph. The relationship data is then processed to build the topic candidate graph used to rank the topics.

DBpedia Spotlight is highly configurable, but in this project it is not explored what the best configuration is. Basic evaluation on a smaller data set shows that a configuration where the settings for confidence and support were set to 0 gave the best recall. Setting the confidence to 0 instructs DBpedia Spotlight to return all annotated matches, even if the system is unsure of the accuracy. The support setting is a way to filter the annotated articles to those with more than the set number of backlinks. Setting support to 0 results in that all articles can be annotated, even if they are not supported (linked to) by any other article in the Wikipedia graph. To limit the scope of the project, all evaluations are run with these settings.

4.3.3 Candidate Relationship Graph

The next step is to take the candidate topics and create a graph of the relationships between these concepts. In this report it is called the *candidate graph*. The selected topic candidates are used to construct the nodes of the graph. The ontology is used in the system in the form of the knowledge graph. The candidate topics matches concepts in the knowledge graph. Relationships are then added between those nodes in the candidate graph that are related in the knowledge graph. Candidate concepts not related to any other concepts in the graph are seen as not related to the text topic. They are therefore not included in the candidate graph. Figure 4.2 shows an example of a candidate relationship graph constructed from a text about bicycles.

4.3.4 Topic Concept Ranking

The main purpose of the topic ranking steps is to determine the main concepts in the input text. The ranking algorithms use the candidate topic relationship graph to determine which concepts that are central in the text. A number of different methods to determine central concepts are evaluated.

TF-IDF

This ranking method uses the TF-IDF score calculated by ElasticSearch. This means that the ranking step applies no modifications to the order of the candidate topics and returns them in the same order as the search engine does. The TF-IDF

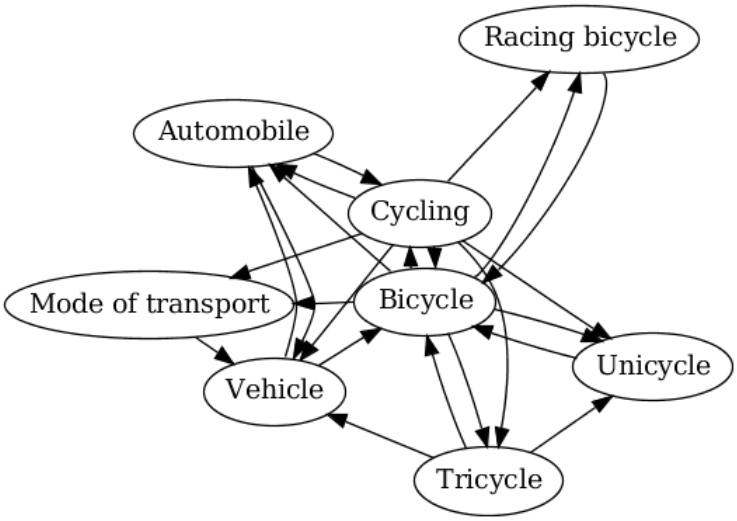

Figure 4.2. Illustration of a candidate relationship graph for a text about bicycles. The edges represents a relationship from one concept to another.

ranking is used as a baseline to compare the graph centrality ranking methods to. This is to determine if graph centrality is effective in improving the topic candidate ranking process.

Degree Centrality

Degree centrality ranking counts links and backlinks from each topic candidate node to another node in the candidate graph. A high count indicates that the node represents a central concept in the text.

Listing 4.2. Degree centrality algorithm

```
1 calculateDegreeCentrality(graph) {
2   foreach node n ∈ graph {
3     n.score = n.nrOfBackLinks + n.nrOfLinks;
4   }
5 }
```

Closeness Centrality

Closeness centrality ranking is based on the distance from a node to all other nodes in the graph. In this implementation, a node that is in a disjoint set has distance 0 to all nodes in other disjoint sets. Therefore, it is not considered when scoring nodes in other disjoint sets.

Listing 4.3. Closeness centrality algorithm

```
calculateClosenessCentrality(graph) {
  foreach node n ∈ graph {
    closeness = 0;
    distances = distances(n);
    for (node to ∈ graph) {
      if distances[to] > 0 {
        closeness += 1.0 / distances[to];
      }
    }
    n.score = closeness;
  }
}

distances(from) {
  distances[n] = null, node n ∈ graph;
  queue = empty queue;
  queue.append(from);
  distances[from] = 0;
  while (queue not empty) {
    n = queue.poll();
    distance = distances[n] + 1;
    foreach neighbor node r of node n {
      if (distances[r] == null) {
        distances[r] = distance;
        queue.add(r);
      }
    }
  }
  return distances;
}
```

Betweenness Centrality

Betweenness centrality ranking takes into account the number of shortest paths that pass through each node. A node is central if many of the shortest paths go through it. Brandes [2001] algorithm for fast calculation of betweenness centrality is used.

Listing 4.4. Betweenness centrality algorithm

```
calculateBetweennessCentrality(graph) {
  centrality[n] = 0, node n ∈ graph;
  foreach node s ∈ graph {
    p[n] = empty list, node n ∈ graph;
    d[n] = -1, node n ∈ graph; d[s] = 0;
    o[n] = 0, node n ∈ graph; o[s] = 1;
    stack = empty stack
    queue = empty queue
    queue.add(s);
    while queue not empty {
      v = queue.poll();
      stack.push(v);
      foreach neighbor node w of node v {
        if (d[w] < 0) {
          queue.add(w);
          d[w] = d[v] + 1;
        }
        if (d[w] == d[v] + 1) {
          o[w] = o[w] + o[v];
          p[w].append(v);
        }
      }
    }
    delta[n] = 0, node n ∈ graph;
    while stack not empty {
      w = stack.pop();
      foreach v ∈ p[w] {
        delta[v] = delta[v] + (o[v]/o[w])*(1 + delta[w]);
      }
      if w ≠ s {
        centrality[w] = centrality[w] + delta[w]; }
    }
  }
  foreach node n ∈ graph{
    n.score = centrality[n]
  }
}
```

PageRank

PageRank ranks the nodes by the importance of their neighbor nodes. An important node is one that is linked to by other important nodes in the candidate graph. A high PageRank score is achieved when a node represents a central concept in the text.

Listing 4.5. PageRank algorithm

```
calculatePageRank(graph) {
  d = 0.85;
  maxChange = 1;
  foreach node n ∈ graph {
    n.score = 1;
  }
  while (maxChange > 0.1) {
    maxChange = 0;
    for (node : graph) {
      currentScore = node.score;
      scoreSum = 0;
      for (n : node.BackLinks) {
        scoreSum += n.score / n.nrOfLinks;
      }
      score = (1 - d) + d * scoreSum;
      node.score = score;
      change = |score - currentScore|;
      if (change > maxChange){
        maxChange = change;
      }
    }
  }
}
```

4.4 Example of Topic Extraction

This section contains an example of how the typical results from the system can look, given a short text focused on a specific topic. In this example, the encyclopedic text about trees (the plant), from Encyclopedia Britannica is used to illustrate the topic extraction process.

Definition of a Tree from Encyclopedia Britannica: *"Tree, woody plant that regularly renews its growth (perennial). Most plants classified as trees have a single self-supporting trunk containing woody tissues, and in most species the trunk produces secondary limbs, called branches."*[2]

4.4.1 Candidate Selection Results

Since this text only consists of one paragraph, the text is not split before the candidate selection. The number of candidates returned by the selection algorithms is narrowed down to the 10 best matches. This is to limit the size of the graphs shown in Figure 4.3 and 4.4. The candidates with no connection to any of the other candidate nodes have not been included in the graphs.

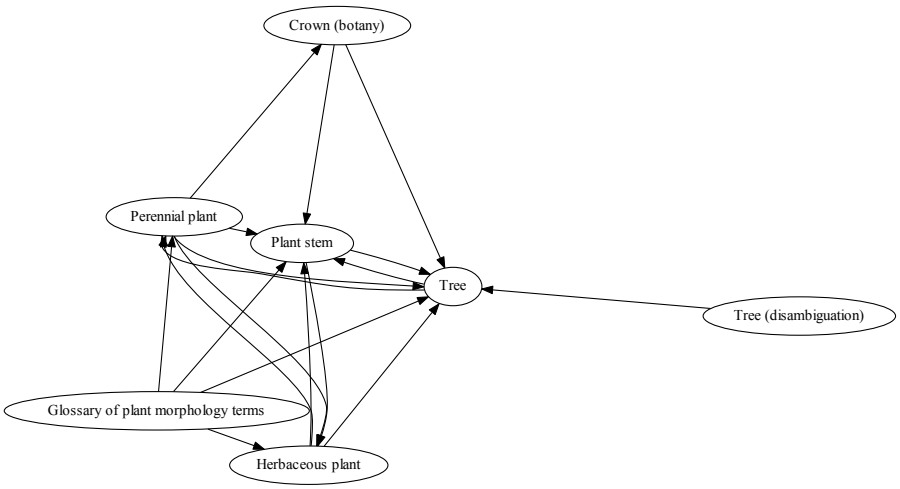

Figure 4.3. Candidate graph created from ElasticSearch (Section+Single Search) candidate selection.

[2] `http://global.britannica.com/EBchecked/topic/603935/tree`, Accessed: 2013-05-10

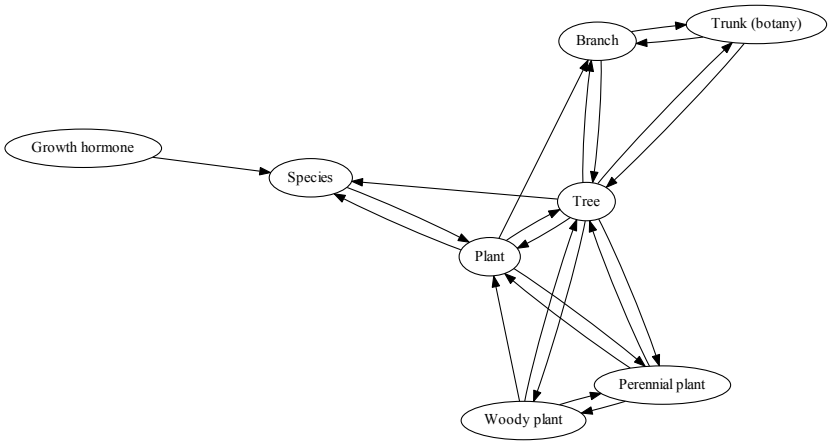

Figure 4.4. Candidate graph created from DBpedia spotlight candidate selection.

4.4.2 Candidate Ranking Results

In Table 4.1 and 4.2 the 5 top ranked results from using closeness centrality to rank the topic candidates are presented.

Rank	Concept	Score
1	Tree	7.00
2	Perennial plant	6.50
3	Plant stem	6.50
4	Herbaceous plant	6.00
5	Glossary of plant morphology terms	6.00

Table 4.1. 5 top ranked topic candidates using closeness centrality on the ElasticSearch candidate selection graph.

Rank	Concept	Score
1	Tree	7.50
2	Plant	7.00
3	Species	6.00
4	Perennial plant	5.83
5	Woody plant	5.83

Table 4.2. 5 top ranked topic candidates using closeness centrality on the DBpedia spotlight candidate selection graph.

As shown, the results are representative for the topic of this text. "Tree" is the highest ranked concept from both algorithms. Most of the other returned top ranked topic candidates are considered as broader or related topics of the text. The scores shown in the tables are the closeness centrality score received by the ranking algorithm.

Chapter 5

Evaluation

This chapter describes how the system is evaluated and how the evaluation dataset is created. It also describes the evaluation methods and the measurements used in the evaluation process.

5.1 Evaluation Data

5.1.1 Simple English Wikipedia Abstracts

Simple English Wikipedia is a wiki containing shorter, simpler versions of the same type of concepts as the larger English Wikipedia. This makes it suitable to evaluate the topic extraction system. The articles cover topics that exist in the larger Wikipedia site, but the texts are different. It is important not to evaluate the system on the exact same texts that constitute the core of its background knowledge. Using the same data that is used to construct the system's knowledge to evaluate the system could possibly give biased results. It would not show if the system is able to extract topics from texts other than those used to construct the knowledge graph.

When using Simple Wikipedia abstract texts for evaluation, the following defines the input and expected output from the system.

- Input: Text of several Wikipedia article introduction paragraphs/abstracts.
- Output: Text of each article abstract with a list of proposed topics. The list of proposed topics should contain the actual name of the article. A higher ranking position of the expected topic suggests a better performance of the ranking algorithm.

50 Article Abstracts Dataset

The first evaluation dataset used is constructed from the first section of 50 randomly selected Simple English Wikipedia articles. The criteria used to select articles are that the article contains three or more sentences and that it has a corresponding article in the normal English Wikipedia. The expected topic for each of these short sections is the title of the article. This dataset is built manually by using the random article function on Wikipedia. The first section, in a Wikipedia article, is often the definition of the main topic of the article. The hope is that the topic of the first section of these article's is limited to the articles main topic.

329 Article Abstracts Dataset

This evaluation dataset is constructed from the Wikipedia Simple English abstract database dump[1]. Abstracts with a text length of over 400 characters are initially selected. This limits the set to 430 different article abstracts. To ensure that the system has the potential to find the correct answer, the set is limited to articles which the titles matches the titles of existing articles in the regular English Wikipedia. This limits the set to 329 articles. As in the other evaluation dataset, the title is used as the expected topic for the text.

5.1.2 Topic Ranking Survey

To determine how the system performs on a more varied collection of texts, a survey was conducted. By conducting a survey it is possible to determine how users of the system would perceive the topic suggestions the system produces. Human judging of the results is important in order to understand how well the system performs. The survey contained eight different texts from four different sources with different text types.

To see how well the system performed on other random encyclopedic texts, two texts were selected from Encyclopedia Britannica. Two article abstracts were randomly selected; one from a computer science paper and one from an economics paper. Two texts from articles on Scholarpedia[2] were also used. The last two texts were from a simple English news site.

For each of these eight texts, several configurations of the system are used to rank topic candidates. For each configuration, the top five ranked topics were saved for evaluation. The set of all ranked topics from all configurations represented the selection of topics to be ranked in the survey. This gave a set of possible topics for each text of sizes varying between 20 to 36 topics. The participants in the survey were asked to mark each of these possible topics with the options, 'Good match',

[1]Database dump used for evaluation: `http://dumps.wikimedia.org/simplewiki/20130227/`, Accessed: 2013-02-30

[2]A wiki for peer reviewed scientific articles: `http://www.scholarpedia.org/`

'Broader topic', 'Related topic' or 'Not a match'. There was also a text field where the participants were asked to describe the topic of the text. This data is used to determine what the participants perceived as well fitting topics for the text.

In total, 21 individuals participated in the topic survey. The topic data collected from the survey is used to evaluate the performance of each of the system configurations utilized to build the survey data. This makes it possible to compare how the system ranked the topic candidates with how well the survey participants marked their relevance. Table 5.1 describes each of the used texts, and the most commonly used topic description from the text field in the topic survey.[3]

Text	Source	Topic
1	Encyclopedia Britannica	Bicycle
2	Simple English News	St Patrick's day
3	Computer Science paper	Spam filtering / Spam detection
4	Scholarpedia	Language
5	Simple English News	Space travel / International Space Station
6	Encyclopedia Britannica	Banana
7	Economics paper	Internet marketing / Sponsored links
8	Scholarpedia	Reinforcement learning

Table 5.1. Description of the texts used in the topic ranking survey. The topic field shows the most common description of the texts topic used by the participants of the survey. Text 3, 5 and 7 had disagreement on a descriptive topic, so the two most used is presented

5.2 Evaluation Methods

5.2.1 Precision and Recall

In information retrieval, two of the most commonly used measures are precision and recall. Combined, they are used to measure how good a search result is. Precision is calculated by counting the number of relevant results and dividing it by the total number of results retrieved in the search. This measure is used to determine how precise the search is.

Recall is calculated by counting the number of relevant results and dividing it by the total number of relevant documents in the search index. This is used to measure the accuracy of the system in finding the relevant documents.

[3] All evaluation data, including the survey results are available at:
`http://www.nada.kth.se/~markusos/thesis/`

Definition 7 *The Precision is calculated by:*

$$Precision = \frac{|\{Relevant\ documents\} \bigcap \{Retrieved\ documents\}|}{|\{Retrieved\ documents\}|}$$

Definition 8 *The Recall is calculated by:*

$$Recall = \frac{|\{Relevant\ documents\} \bigcap \{Retrieved\ documents\}|}{|\{Relevant\ documents\}|}$$

In the case of evaluating a topic extraction system, a relevant document is defined as the expected topics of the input texts and a retrieved document is the topics suggested by the topic extraction system. With this definition, it is more interesting to look at the recall of the system than the precision. The recall shows how many of the texts were matched to their expected topics, in the set of returned topics. Looking at precision is not as relevant, when there is only one acceptable correct answer per text.

5.2.2 Average Score

The average score measure is constructed to take not only the ranking position into account, but also the recall of the results. This makes it useful to compare results from different ranking algorithms when the recall is not equivalent.

For each evaluation case, the score function gives a score between 0 and 10. The score is based on where the expected topic is found in the result-ranking list. An expected topic found at ranking position 1 is given a score of 10. The score is lowered linearly by how far down in the result list the topic is located. Topics not found or located below the 10th position are given a score of 0.

The average score only takes into account the top 10 ranked topics. This is based on the assumption that it is unlikely that a human editor, using a topic system as a tool for annotating articles, would examine more than 10 suggested topics.

Definition 9 *The Score function*

$$S(t) = \begin{cases} 11 - R(t) & \text{if } R(t) < 11 \\ 0 & \text{if } R(t) \geq 11 \end{cases}$$

Where t is the expected topic for the text. R(t) is the ranking position of topic t in the result list. $R(t) = \infty$ *if the result list does not contain t.*

The average score can be used on evaluation data where one expected topic exists, e.g. the Simple Wikipedia abstract dataset. The average score can then be used as a measure to compare how different candidate ranking methods perform on the evaluation dataset. A perfect system would have an average score of 10. This would means that the system has 100% recall and that all expected topics are ranked highest in the list of suggested topics. A change in both recall and the ranked position has an affect on the average score, which makes it useful when comparing combinations of topic ranking algorithms and candidate selection algorithms.

Definition 10 *The Average score is calculated by:*

$$Average\ score = \frac{1}{|T|} \times \sum_{t \in T} S(t)$$

Where T is the set of topic evaluation data, S(t) is the score of the expected topic t.

5.2.3 Rank Sum Score

The rank sum score is a scoring method that builds on ideas from the Wilcoxon Rank-Sum Test [Wild, 2010]. The ranking method can be used to compare groups of measurements to determine if differences between the groups exist.

The results from the survey are used to rank topic concepts. Each concept is assigned a value determined by the distribution of classifications in the survey. A topic concept that a large percentage of the participants classified as a "Good match" receives a higher score than a topic classified as "Not a match".

Definition 11 *The survey score is calculated by:*

$Survey\ Score = 4 * C(Good\ match) + 2 * C(Broader\ topic) + 1 * C(Related\ topic)$;

Where C(a) is the percent of participants that ranked the topic concept in category a.

To calculate the rank sum score, all measurements are then ranked according to the survey score values. Topic concepts that share the same score are given the average rank of all concepts with that score. See Table 5.2 for an example of how this is calculated.

This is calculated for all the concepts belonging to each of the texts used in the survey. The rank sum score is then calculated as the sum of rank for each of the five concepts that the ranking algorithms return.

E.g. a ranking algorithm that produces the concepts A, C, D, E, I receives the rank sum score of $1+2.5+4+5+9.5 = 22$, compared to an algorithm that produces

Concept	A	B	C	D	E	F	G	H	I	J
Score	3.7	3.5	3.5	3	2.5	2	1.7	1.5	1	1
1 to n	1	2	3	4	5	6	7	8	9	10
Rank	1	2.5	2.5	4	5	6	7	8	9.5	9.5

Table 5.2. Example on how the ranking is calculated given the score

the concepts B, F, G, H, J that receives the score $2.5+6+7+8+9.5 = 33$. The first algorithm has a better score in this case, because a lower score is the result of more highly ranked concepts. This gives a comparable value for each ranking algorithm and text.

The average rank sum score, over all eight texts, can then be used to compare the performance of the ranking algorithms on all survey texts.

5.3 Evaluating Candidate Selection

The evaluation of topic recall from the candidate selection algorithm is important. If the topic candidate graph does not contain the expected topic of the text, the system will not be able to rank that topic high in the topic ranking phase. The perfect solution would have a high recall in the output from the candidate selection algorithm, making it possible for the ranking phase to always have access to the correct response. To evaluate the recall of the candidate selection components, the Simple English Wikipedia abstract datasets are used. Table 5.3 shows the configurations used for the evaluation.

The recall for each text in the evaluation set is either 1 or 0, depending on if the search results contains the article title (the expected topic). The value presented in Figure 6.1 and Figure 6.2 is the recall of articles where the topic is found. Recall 100% would be if the correct topic was found for all articles in the evaluation dataset.

Candidate Selection	Dataset	Search method	# search results
DBpedia Spotlight	50 Samples	n/a	n/a
ElasticSearch	50 Samples	Single Search	5, 10, 20, 30, 40, 50, 100
ElasticSearch	50 Samples	Multi Search	5, 10, 20, 30, 40, 50, 100
DBpedia Spotlight	329 Samples	n/a	n/a
ElasticSearch	329 Samples	Single Search	5, 10, 20, 30, 40, 50
ElasticSearch	329 Samples	Multi Search	5, 10, 20, 30, 40, 50

Table 5.3. Cases used to evaluate the Candidate Selection.

5.4 Evaluating Topic Ranking

5.4.1 Simple English Wikipedia Abstracts

To determine which of the ranking methods performs best, the topic ranking position of an expected topic is evaluated. For this, the Simple English Wikipedia abstracts data sets are used. Each ranking algorithm is to return the top 10 ranked candidates for each text. The recall and average score are then calculated over the entire evaluation set. This gives each ranking algorithm two measures that can be used for comparisons. Table 5.4 shows the configurations used for the evaluation. The results are presented in Section 6.3 and 6.4.

Precision is not used, because it is more informative to examine the order of the results with the average score measure, than it is to examine the precision. As mentioned before, the precision is not a good measure when only one possible relevant topic is taken into account for each text.

Candidate Selection	Ranking	Search method	# search results
DBpedia Spotlight	Closeness	n/a	n/a
DBpedia Spotlight	Betweenness	n/a	n/a
DBpedia Spotlight	PageRank	n/a	n/a
DBpedia Spotlight	Degree	n/a	n/a
ElasticSearch	Closeness	Single Search	5, 10, 20, 30, 40, 50, (100)
ElasticSearch	Closeness	Multi Search	5, 10, 20, 30, 40, 50, (100)
ElasticSearch	Betweenness	Single Search	5, 10, 20, 30, 40, 50, (100)
ElasticSearch	Betweenness	Multi Search	5, 10, 20, 30, 40, 50, (100)
ElasticSearch	PageRank	Single Search	5, 10, 20, 30, 40, 50, (100)
ElasticSearch	PageRank	Multi Search	5, 10, 20, 30, 40, 50, (100)
ElasticSearch	Degree	Single Search	5, 10, 20, 30, 40, 50, (100)
ElasticSearch	Degree	Multi Search	5, 10, 20, 30, 40, 50, (100)
ElasticSearch	TF-IDF	Single Search	5, 10, 20, 30, 40, 50, (100)

Table 5.4. Cases used to evaluate the Topic Ranking using the 50 and 329 Samples Simple English Wikipedia dataset. The 100 search results setting was only used on the 50 samples dataset.

5.4.2 Topic Ranking Survey

The topic ranking is also evaluated using the survey results. A method using rank sum to score the algorithm's results is used to compare the performances. Each ranking algorithm's top 5 ranked topics are considered, and are used to calculate the rank sum. Table 5.5 presents the cases used to construct the survey. The ranking of each concept is determined by the responses from the participants in the topic ranking survey. The ranking is performed using the method described in Section

5.2.3. The rank sum is calculated for each algorithm on all of the eight evaluation texts. For comparison, the average rank sum over all texts is also calculated.

The survey score is also used to compare the algorithms. The average of the survey scores of the texts is used to compare how well the algorithms performed on individual texts. The theoretical maximum of this measure is 4, but given that there were usually at most one topic from each text that was close to that score, a reasonably good algorithm would have an average score around 1.5. The average over all texts is used to compare the algorithms. All results are presented in Section 6.6.1.

Candidate Selection	Ranking	Search method	# search results
ElasticSearch	Closeness	Single Search	20
ElasticSearch	Closeness	Multi Search	20
ElasticSearch	Betweenness	Single Search	20
ElasticSearch	Betweenness	Multi Search	20
ElasticSearch	PageRank	Single Search	20
ElasticSearch	PageRank	Multi Search	20
ElasticSearch	Degree	Single Search	20
ElasticSearch	Degree	Multi Search	20
ElasticSearch	TF-IDF	Single Search	20

Table 5.5. Cases used to evaluate the Topic Ranking. The top 5 results from these cases were used to construct the survey where the participants classified the results.

Chapter 6

Results

In this chapter, all evaluation results are presented. TF-IDF is used as a baseline to compare to the graph centrality algorithms. All tables and diagrams contain the corresponding score or recall from ElasticSearch Single Search TF-IDF results. It is important to note that the TF-IDF used as a baseline here uses the same background knowledge as the other algorithms.

- Recall is presented in the range 0 to 1, where 1 corresponds to 100% recall.
- Average Score is presented in the range 0 to 10, where 10 corresponds to all expected topics that are ranked by the candidate ranking algorithm.
- Rank Sum is the sum of the rank of the concepts returned by each candidate ranking algorithm. A lower rank sum indicates that, according to the survey, more of the highly relevant topics are returned by that ranking algorithm.
- The survey score represents the average survey score of the 5 topic concepts returned by each ranking algorithm. The average of these from all 8 texts are used for comparison. A high survey score indicates that more of the highly relevant topics are returned by that ranking algorithm.

6.1 DBpedia Spotlight Candidate Selection Recall

Dataset	Recall
50	0.98
329	0.84

Table 6.1. Recall for all extracted candidates of the 50 and 329 data set using DBpedia spotlight candidate selection algorithm.

6.2 ElasticSearch Candidate Selection Recall

6.2.1 50 Samples Dataset

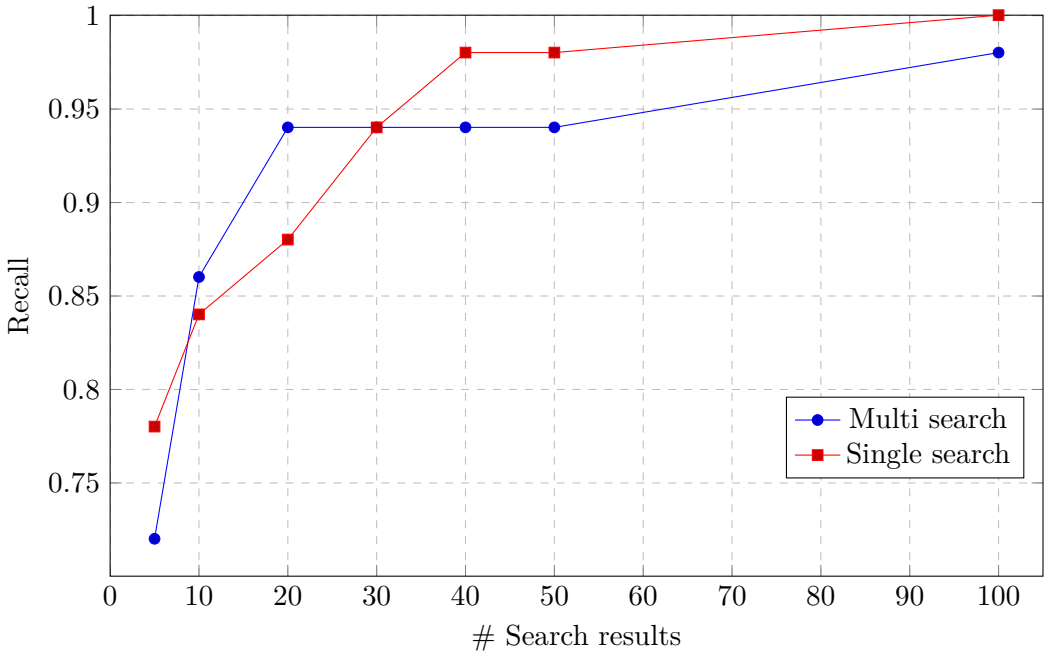

Figure 6.1. Recall for all extracted candidates of the 50 data set with different parameters for the candidate selection algorithm.

SearchResults	MultiSearch	SingleSearch
5	0.72	0.78
10	0.86	0.84
20	0.94	0.88
30	0.94	0.94
40	0.94	0.98
50	0.94	0.98
100	0.98	1.00

Table 6.2. Recall for all extracted candidates of the 50 data set with different parameters for the candidate selection algorithm.

6.2.2 329 Samples Dataset

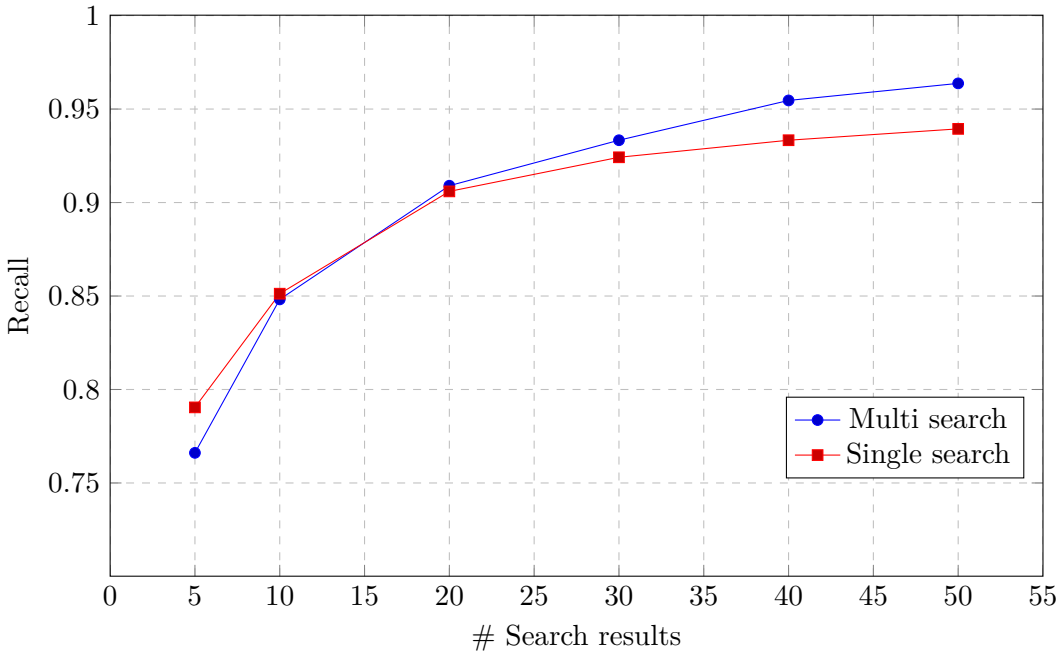

Figure 6.2. Recall for all extracted candidates of the 329 data set with different parameters for the candidate selection algorithm.

SearchResults	MultiSearch	SingleSearch
5	0.77	0.79
10	0.85	0.85
20	0.91	0.91
30	0.93	0.92
40	0.95	0.93
50	0.96	0.94

Table 6.3. Recall for all extracted candidates of the 329 data set with different parameters for the candidate selection algorithm.

6.3 Simple Wikipedia Abstracts (50 Samples Dataset)

6.3.1 Single Search Candidate Selection

Score

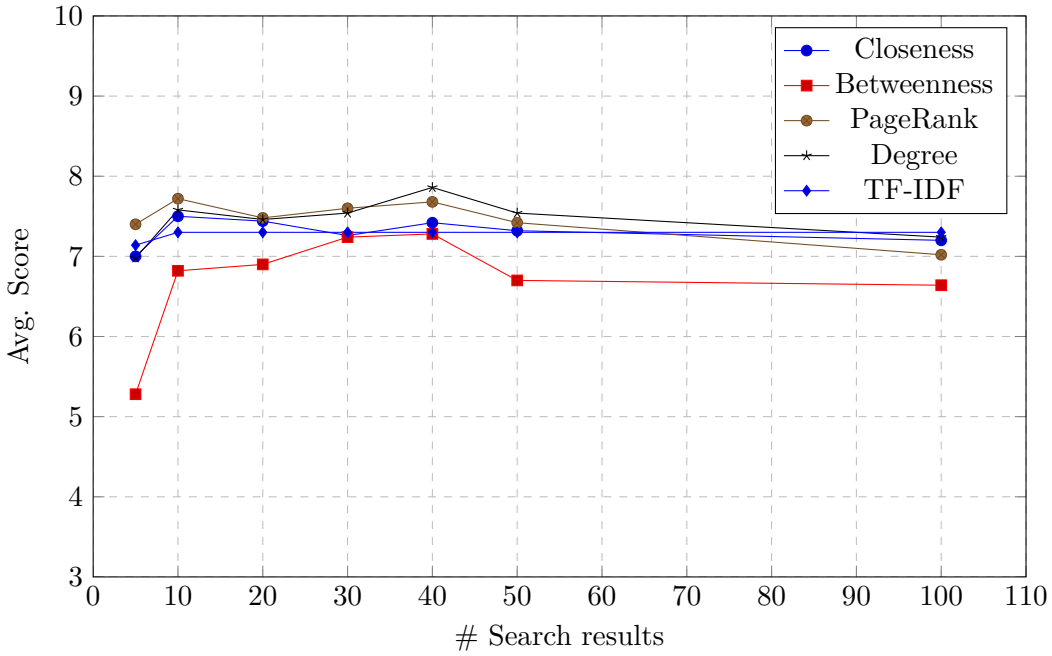

Figure 6.3. Result score for different ranking algorithms using the single search candidate selection.

SearchResults	Closeness	Betweenness	PageRank	Degree	TF-IDF
5	7.00	5.28	7.40	6.98	7.14
10	7.50	6.82	7.72	7.58	7.30
20	7.44	6.90	7.48	7.46	7.30
30	7.26	7.24	7.60	7.54	7.30
40	7.42	7.28	7.68	7.86	7.30
50	7.32	6.70	7.42	7.54	7.30
100	7.20	6.64	7.02	7.24	7.30

Table 6.4. Result score for different ranking algorithms using the single search candidate selection where SearchResults is the number of search results for each search query.

Recall

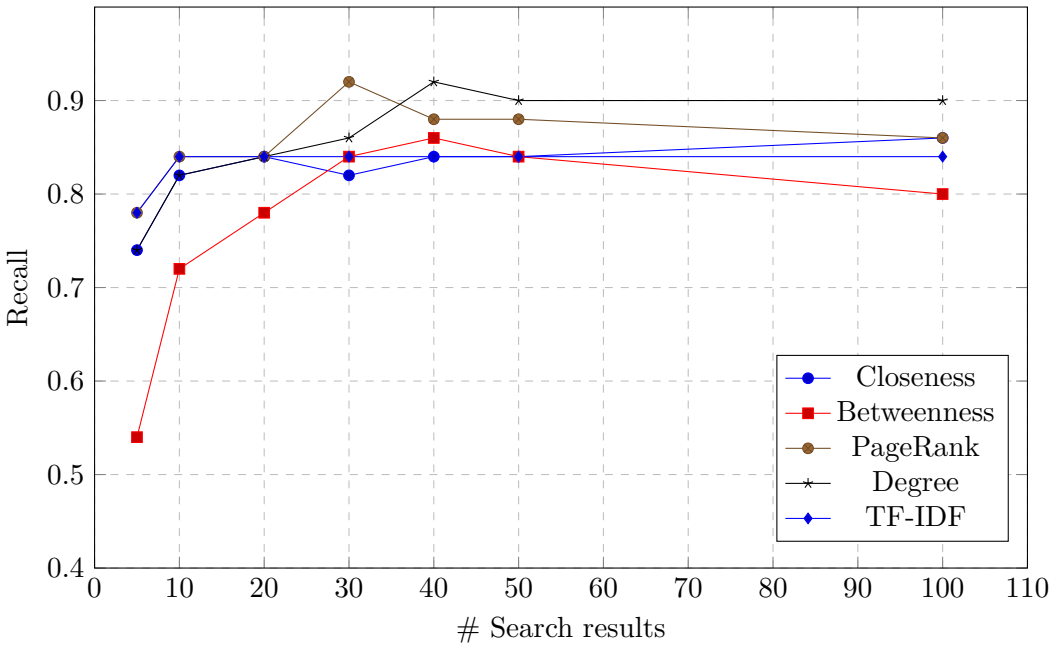

Figure 6.4. Result recall for different ranking algorithms using the single search candidate selection.

SearchResults	Closeness	Betweenness	PageRank	Degree	TF-IDF
5	0.74	0.54	0.78	0.74	0.78
10	0.82	0.72	0.84	0.82	0.84
20	0.84	0.78	0.84	0.84	0.84
30	0.82	0.84	0.92	0.86	0.84
40	0.84	0.86	0.88	0.92	0.84
50	0.84	0.84	0.88	0.90	0.84
100	0.86	0.80	0.86	0.90	0.84

Table 6.5. Result recall for different ranking algorithms using the single search candidate selection where SearchResults is the number of search results for each search query.

6.3.2 Multi Search Candidate Selection

Score

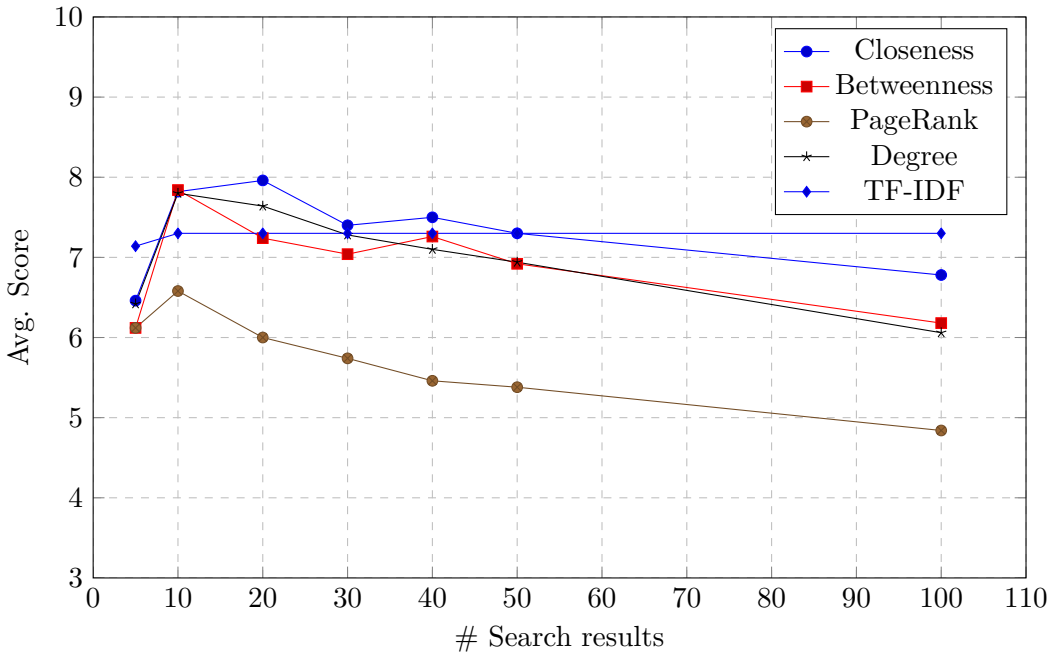

Figure 6.5. Result score for different ranking algorithms using the multi search candidate selection.

SearchResults	Closeness	Betweenness	PageRank	Degree	TF-IDF
5	6.46	6.12	6.12	6.42	7.14
10	7.82	7.84	6.58	7.80	7.30
20	7.96	7.24	6.00	7.64	7.30
30	7.40	7.04	5.74	7.28	7.30
40	7.50	7.26	5.46	7.10	7.30
50	7.30	6.92	5.38	6.94	7.30
100	6.78	6.18	4.84	6.06	7.30

Table 6.6. Result score for different ranking algorithms using the multi search candidate selection where SearchResults is the number of search results for each search query.

Recall

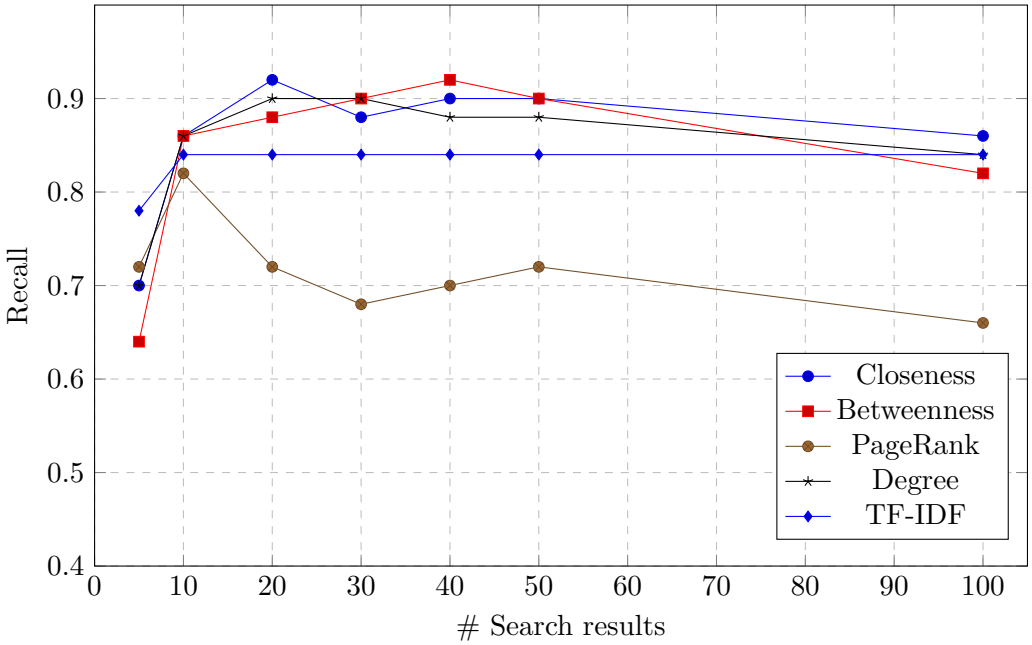

Figure 6.6. Result recall for different ranking algorithms using the multi search candidate selection.

SearchResults	Closeness	Betweenness	PageRank	Degree	TF-IDF
5	0.70	0.64	0.72	0.70	0.78
10	0.86	0.86	0.82	0.86	0.84
20	0.92	0.88	0.72	0.90	0.84
30	0.88	0.90	0.68	0.90	0.84
40	0.90	0.92	0.70	0.88	0.84
50	0.90	0.90	0.72	0.88	0.84
100	0.86	0.82	0.66	0.84	0.84

Table 6.7. Result recall for different ranking algorithms using the multi search candidate selection where SearchResults is the number of search results for each search query.

6.4 Simple Wikipedia Abstracts (329 Samples Dataset)

6.4.1 Single Search Candidate Selection

Score

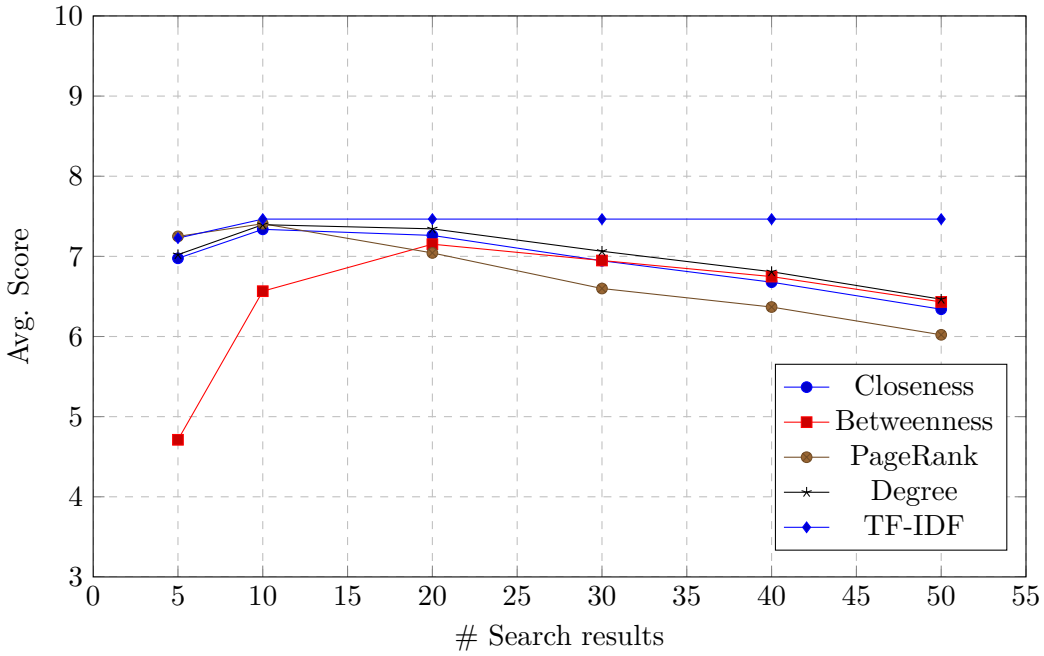

Figure 6.7. Result score for ranking algorithms using the single search candidate selection.

SearchResults	Closeness	Betweenness	PageRank	Degree	TF-IDF
5	6.98	4.71	7.25	7.02	7.23
10	7.34	6.57	7.41	7.40	7.47
20	7.26	7.15	7.04	7.34	7.47
30	6.95	6.95	6.60	7.06	7.47
40	6.68	6.75	6.37	6.81	7.47
50	6.34	6.43	6.02	6.47	7.47

Table 6.8. Result score for ranking algorithms using the single search candidate selection. SearchResults is the number of search results for each search query.

Recall

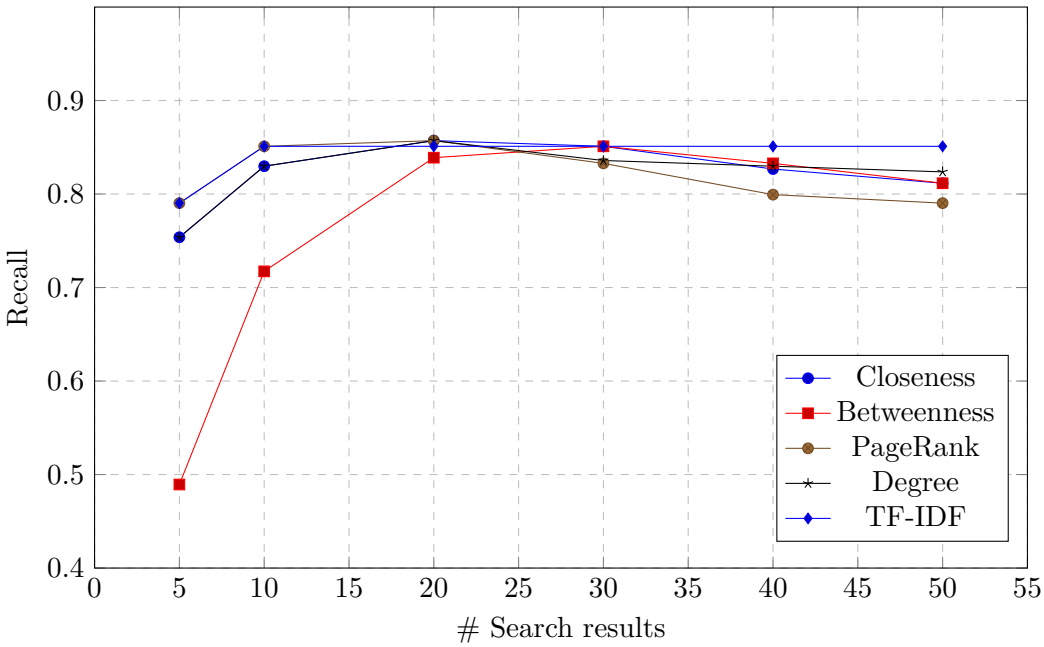

Figure 6.8. Result recall for different ranking algorithms using the single search candidate selection.

SearchResults	Closeness	Betweenness	PageRank	Degree	TF-IDF
5	0.75	0.49	0.79	0.75	0.79
10	0.83	0.72	0.85	0.83	0.85
20	0.86	0.84	0.86	0.86	0.85
30	0.85	0.85	0.83	0.84	0.85
40	0.83	0.83	0.80	0.83	0.85
50	0.81	0.81	0.79	0.82	0.85

Table 6.9. Result recall for different ranking algorithms using the single search candidate selection where SearchResults is the number of search results for each search query.

6.4.2 Multi Search Candidate Selection

Score

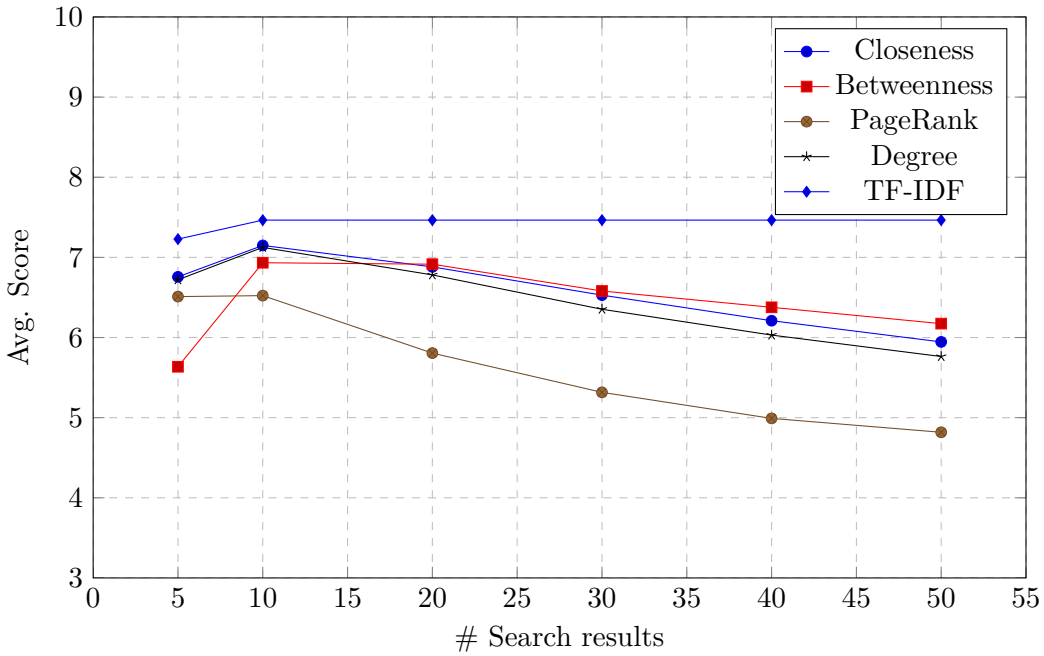

Figure 6.9. Result score for different ranking algorithms using the multi search candidate selection.

SearchResults	Closeness	Betweenness	PageRank	Degree	TF-IDF
5	6.76	5.64	6.51	6.72	7.23
10	7.15	6.93	6.52	7.12	7.47
20	6.88	6.91	5.81	6.78	7.47
30	6.53	6.58	5.32	6.35	7.47
40	6.21	6.38	4.99	6.03	7.47
50	5.95	6.17	4.82	5.76	7.47

Table 6.10. Result score for different ranking algorithms using the multi search candidate selection where SearchResults is the number of search results for each search query.

Recall

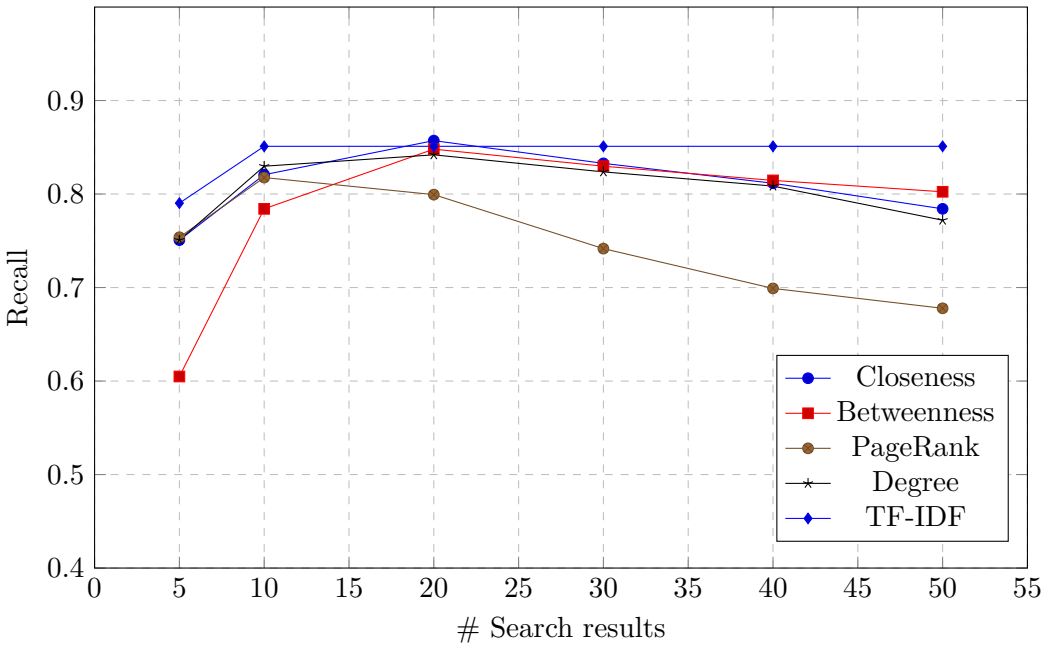

Figure 6.10. Result recall for different ranking algorithms using the multi search candidate selection.

SearchResults	Closeness	Betweenness	PageRank	Degree	TF-IDF
5	0.75	0.60	0.75	0.75	0.79
10	0.82	0.78	0.82	0.83	0.85
20	0.86	0.85	0.80	0.84	0.85
30	0.83	0.83	0.74	0.82	0.85
40	0.81	0.81	0.70	0.81	0.85
50	0.78	0.80	0.68	0.77	0.85

Table 6.11. Result recall for different ranking algorithms using the multi search candidate selection where SearchResults is the number of search results for each search query.

6.5 DBpedia Spotlight

6.5.1 50 Samples Dataset

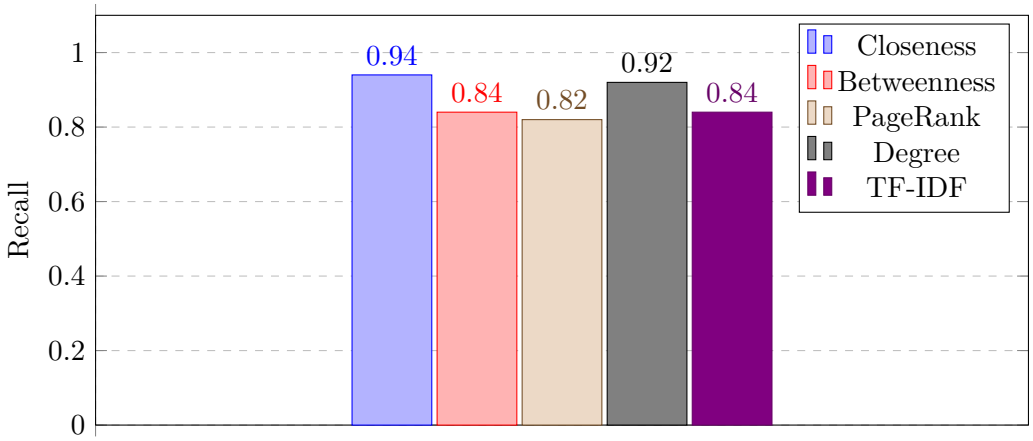

Figure 6.11. Result recall for different ranking algorithms using DBpedia spotlight candidate selection. A higher value is better. TF-IDF is used as a baseline.

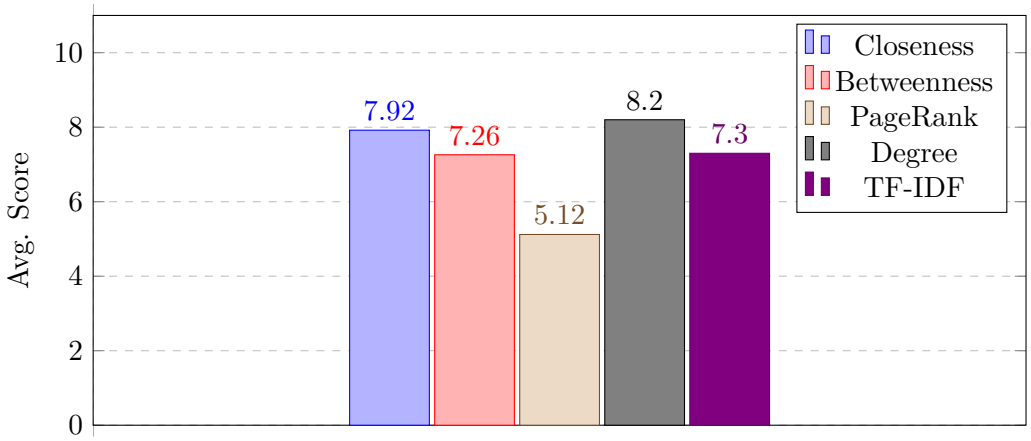

Figure 6.12. Result score for different ranking algorithms using DBpedia spotlight candidate selection. A higher value is better. TF-IDF is used as a baseline.

Measure	Closeness	Betweenness	PageRank	Degree	TF-IDF
Recall	0.94	0.84	0.82	0.92	0.84
Avg. Score	7.92	7.26	5.12	8.20	7.30

Table 6.12. DBpedia spotlight recall and score (50 samples dataset).

6.5.2 329 Samples Dataset

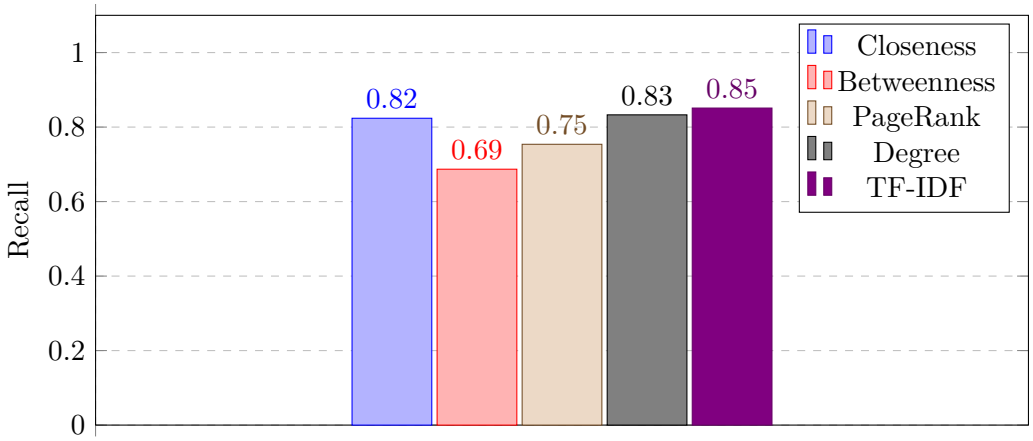

Figure 6.13. Result recall for different ranking algorithms using DBpedia spotlight candidate selection. A higher value is better. TF-IDF is used as a baseline.

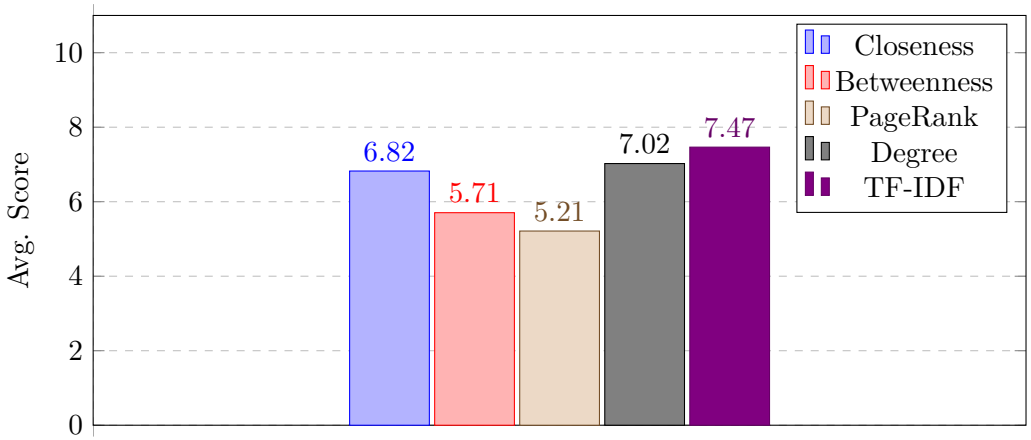

Figure 6.14. Result score for different ranking algorithms using DBpedia spotlight candidate selection. A higher value is better. TF-IDF is used as a baseline.

Measure	Closeness	Betweenness	PageRank	Degree	TF-IDF
Recall	0.82	0.69	0.75	0.83	0.85
Avg. Score	6.82	5.71	5.21	7.02	7.47

Table 6.13. DBpedia spotlight recall and score (329 samples dataset).

6.6 Survey Results

6.6.1 Average Rank Sum

Single Search

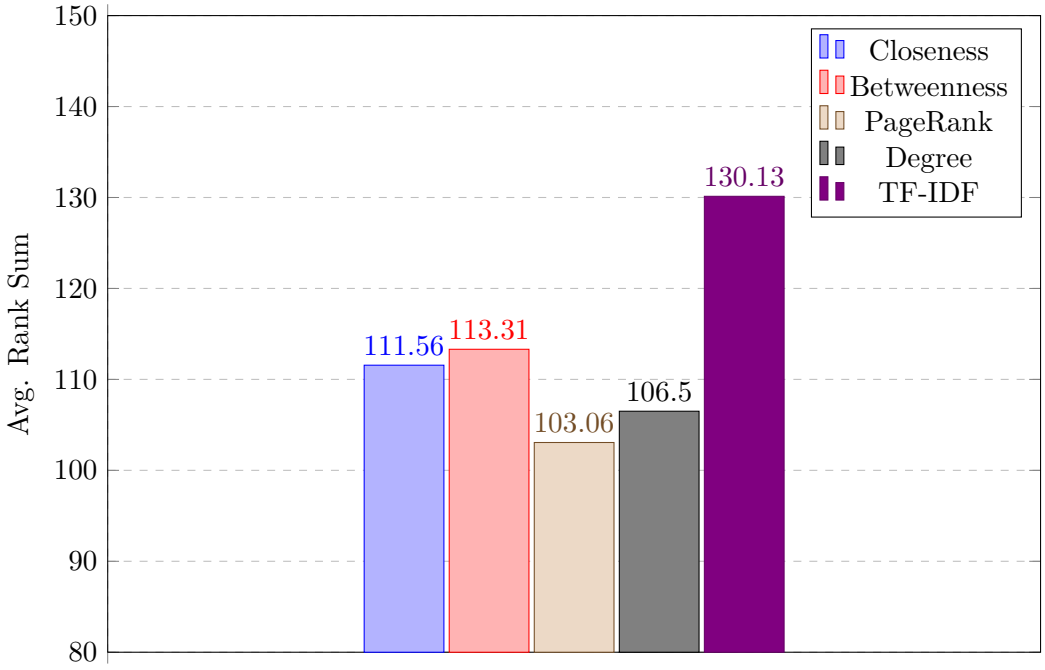

Figure 6.15. Rank sum from survey result using ElasticSearch Single Search. A lower score is better. TF-IDF is used as a baseline.

Text	Closeness	Betweenness	PageRank	Degree	TF-IDF
Text 1	137.50	113.50	108.00	120.00	152.50
Text 2	88.50	116.00	82.00	88.50	116.00
Text 3	90.50	114.50	90.50	90.50	146.00
Text 4	89.00	87.00	91.00	89.00	126.00
Text 5	92.50	125.00	89.50	92.50	168.50
Text 6	127.00	127.00	115.00	127.00	121.00
Text 7	142.00	132.00	147.50	132.00	112.00
Text 8	125.50	91.50	101.00	112.50	99.00
Avg.	111.56	113.31	103.06	106.50	130.13

Table 6.14. Rank sum from survey result using ElasticSearch Single Search. A lower score is better. TF-IDF is used as a baseline.

Multi Search

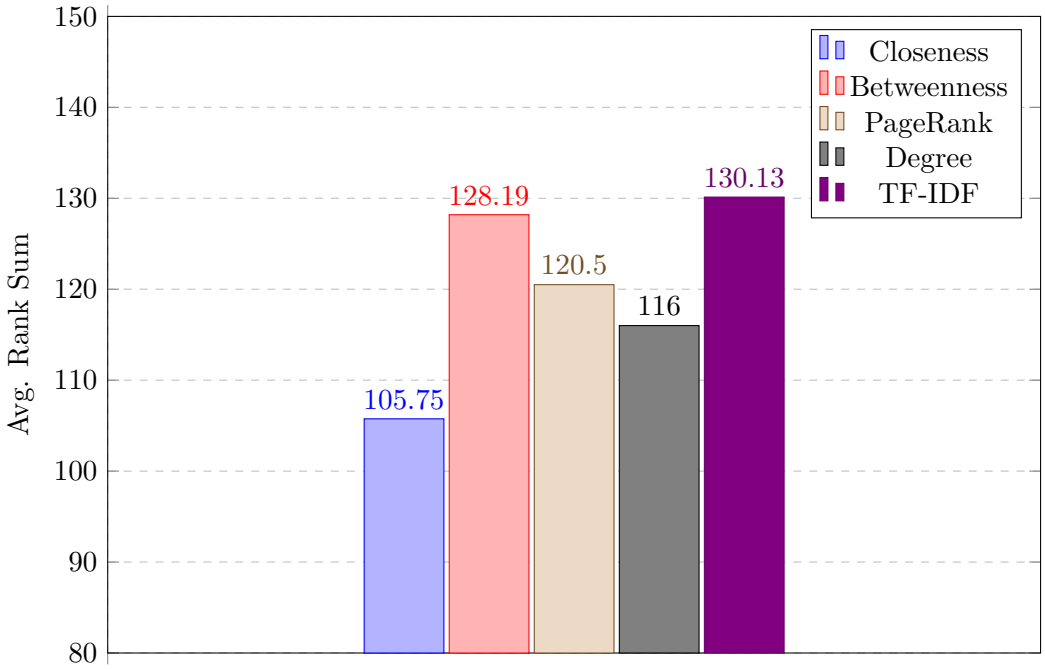

Figure 6.16. Rank sum from survey result using ElasticSearch Multi Search. A lower score is better. TF-IDF is used as a baseline.

Text	Closeness	Betweenness	PageRank	Degree	TF-IDF
Text 1	86.00	124.00	107.50	86.00	152.50
Text 2	131.00	131.00	141.00	141.00	116.00
Text 3	116.00	164.00	111.50	111.50	146.00
Text 4	127.00	142.00	142.00	142.00	126.00
Text 5	100.50	147.00	106.50	113.00	168.50
Text 6	104.50	104.50	104.50	104.50	121.00
Text 7	94.50	91.50	120.50	63.00	112.00
Text 8	86.50	121.50	130.50	167.00	99.00
Avg.	105.75	128.19	120.50	116.00	130.13

Table 6.15. Rank sum from survey result using ElasticSearch Multi Search. A lower score is better. TF-IDF is used as a baseline.

6.6.2 Survey Score

Single Search

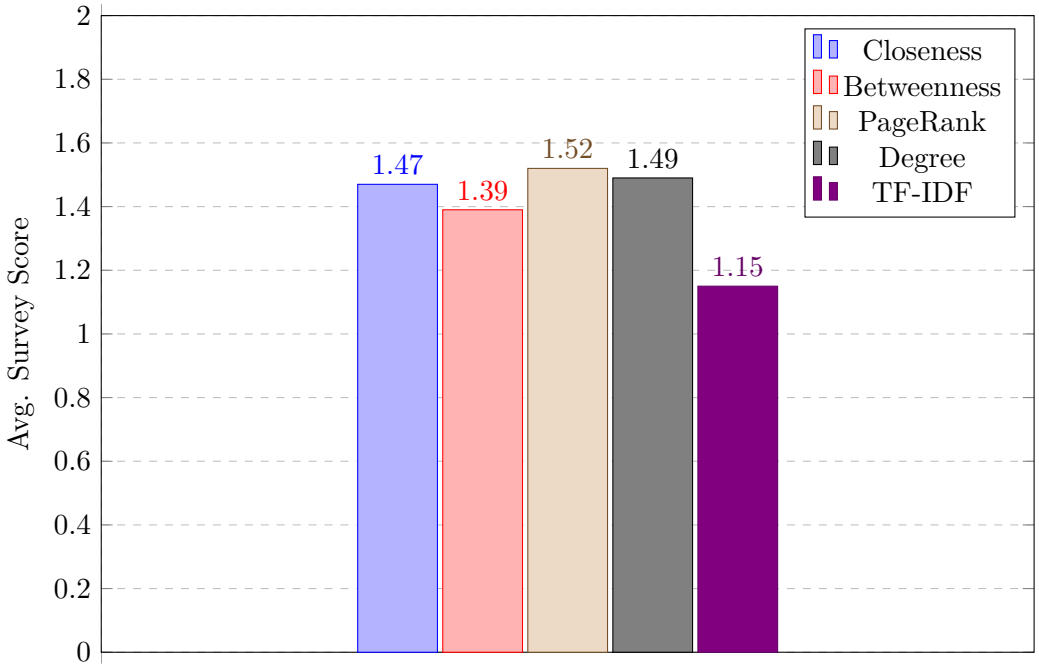

Figure 6.17. Average survey score from survey result using ElasticSearch Single Search. A higher score is better. TF-IDF is used as a baseline.

Text	Closeness	Betweenness	PageRank	Degree	TF-IDF
Text 1	1.20	1.36	1.40	1.34	0.68
Text 2	1.67	1.40	1.74	1.67	1.40
Text 3	2.28	1.74	2.28	2.28	1.37
Text 4	2.02	2.10	2.00	2.02	1.30
Text 5	1.60	1.29	1.61	1.60	0.87
Text 6	1.56	1.56	1.59	1.56	1.58
Text 7	0.10	0.10	0.10	0.10	0.60
Text 8	1.30	1.57	1.46	1.38	1.44
Avg.	1.47	1.39	1.52	1.49	1.15

Table 6.16. Average survey score from survey result using ElasticSearch Single Search. A higher score is better. TF-IDF is used as a baseline.

Multi Search

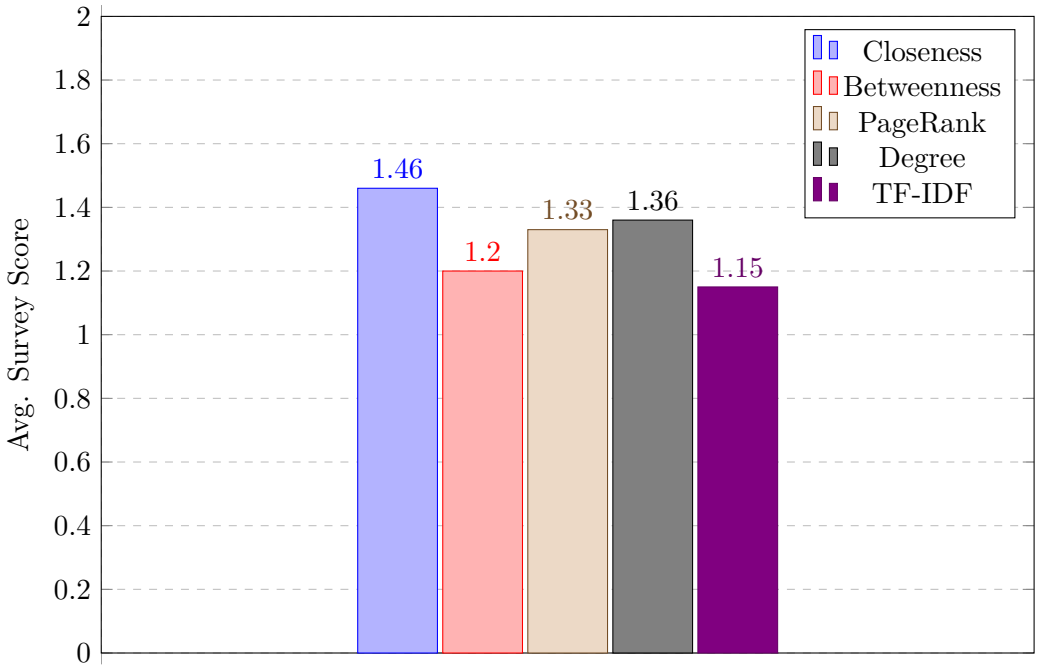

Figure 6.18. Average survey score from survey result using ElasticSearch Multi Search. A higher score is better. TF-IDF is used as a baseline.

Text	Closeness	Betweenness	PageRank	Degree	TF-IDF
Text 1	1.51	1.31	1.36	1.51	0.68
Text 2	1.34	1.34	1.25	1.25	1.40
Text 3	1.83	1.05	1.88	1.88	1.37
Text 4	1.45	1.25	1.25	1.25	1.30
Text 5	1.60	1.09	1.50	1.45	0.87
Text 6	1.73	1.73	1.73	1.73	1.58
Text 7	0.69	0.48	0.34	0.81	0.60
Text 8	1.51	1.39	1.30	1.00	1.44
Avg.	1.46	1.20	1.33	1.36	1.15

Table 6.17. Average survey score from survey result using ElasticSearch Multi Search. A higher score is better. TF-IDF is used as a baseline.

Chapter 7

Discussion and Conclusion

In this chapter, the results are discussed and the conclusions are presented, as well as some suggestions for future work.

7.1 Candidate Selection Recall

The ElasticSearch Multi search, Single search, and the DBpedia Spotlight candidate selection method all performed similarly when comparing recall on the evaluation datasets. Using the DBpedia spotlight selection method on the smaller evaluation data set had a recall of 98% and a recall of 84% on the 329 evaluation set. These results are comparable to the results achieved when using ElasticSearch for candidate selection, with 20 or more search results, as can be seen in Section 6.2.

It is interesting to compare the results from the ElasticSearch topic candidate selection with the results when using a state-of-the-art disambiguation system like DBpedia spotlight. It shows that using a search engine like ElasticSearch with TF-IDF as a candidate selection algorithm is similar to using DBpedia spotlight. Even an advanced system like DBpedia spotlight has problems with extracting correct concepts from some texts.

7.2 Simple Wikipedia Evaluation

The results from the evaluation on the two simple Wikipedia datasets show that the use of a search engine with TF-IDF results in a high recall and highly ranked expected topics on average.

It is clear that all graph centrality ranking algorithms perform worse, given larger graph structures. This indicates two things. First, it shows that the ElasticSearch candidate selection step performs well already with just a few candidates. Secondly, the graphs, constructed from the ElasticSearch candidate selection step,

are probably biased towards the concepts containing words highly ranked by the search engine's TF-IDF score. This affects the possibility to improve the results with graph centrality ranking. The quality of the knowledge graph, in this case the Wikipedia graph, also affects the quality of the extracted candidates and, in turn, the candidate graph.

On the smaller dataset, graph centrality measures, like closeness centrality and degree centrality, outperform the TF-IDF score when parameter for the *number of search results* was set to return around 20 results. Looking at the results from the larger Simple Wikipedia dataset, all of the centrality ranking algorithms perform slightly worse than the single search TF-IDF implementation. The difference is not large enough to say anything conclusive, but there might be several reasons behind this. TF-IDF returns results that are hard to improve, both because they already have a recall of 85% and also because the graph structure itself is affected by the TF-IDF ranking. The scoring does not take into account the correctness of the other results provided, but only scores based on the expected topic's rank. This might also affect the result in favor of TF-IDF. The conclusion drawn is that the graph ranking in some cases gives noticeably better results compared to TF-IDF, but in other cases the difference is not significant, when measuring recall and average score.

7.3 Survey Data Evaluation

The data used for the survey is the results from ElasticSearch candidate selection in combination with the different ranking algorithms. The ElasticSearch candidate selection is configured to use the top 20 search results. This is because the recall, for most algorithms, peaked at around 20 search results as can be seen in figure 6.8 and 6.10. To get a greater distinction of the differences between the ranking algorithms, the top 5 ranked results from each algorithm was used in the survey.

The survey shows that using graph centrality seems to give some advantage to only using TF-IDF. In the single search results, all graph centrality ranking algorithms perform noticeably better than the TF-IDF ranking. For multi search the results show that both closeness centrality and degree centrality still performs noticeably better, while the other graph centrality algorithms only get marginally better results than TF-IDF. Both Closeness centrality and Degree centrality seem to have stable better results than TF-IDF, despite the settings of the ElasticSearch candidate selection.

As the results in Figure 6.15 and 6.17 show, when using single search and graph centrality to rank the concepts, the result is noticeably better than using only TF-IDF for the ranking. The small dataset with 21 responses only gives an indication that it might work. It is not ensured that this is the case for all types of texts, as the results show a variation of the performance on different types of texts.

Table 6.16 and 6.17 shows that for most of the texts, the TF-IDF ranking algorithm received lower average survey score results than the graph ranking algorithms. When looking at the results for each text individually in those tables, text 7 stands out with significantly worse results than on the other texts. Text 7 is also the text where fewest participants in the survey had overlapping topic classifications when asked to describe the topic of the text. The conclusion from this is that it might be harder to classify the topic of text 7, even for humans, and the automatic system also has troubles with the text.

7.4 Sources of Errors

In this section possible sources of errors in the results are discussed. All evaluation results are dependent on a number of different parameters. It is almost impossible to determine the impact from each of them. To limit the scope of the project some parameters have been set arbitrarily.

7.4.1 Candidate Selection Recall

The problem of getting a perfect recall stems from a few main difficulties. The Wikipedia graph contains concepts and relationships that are problematic or misleading, especially when using TF-IDF as a model for extracting topic candidates. Concepts that contain relatively common words in the English language, but not common in Wikipedia article names, get an unrealistically high score in some cases. This is partially solved by using a more extensive stop word list. A problem with using an extensive stop word list is that it is hard to determine if a word is important enough, when determining context and topic, to not have it on the stop word list.

7.4.2 Simple Wikipedia Evaluation

The graph ranking seems to be biased towards concepts that are central in the Wikipedia graph. Concepts like 'United States' that link to an enormous amount of other concepts, even if they are not necessarily defining the concept, lead to problems when they connect otherwise unconnected graph components. Concepts that are central in the Wikipedia graph therefore end up being central in the candidate selection graph.

Another possible problem with the evaluation is that it uses Wikipedia data to evaluate a system built on Wikipedia data. By using Simple English Wikipedia some of the bias is avoided, but since Simple Wikipedia articles often are based on articles from Wikipedia it is still to be seen as a possible problem. The use of Simple Wikipedia data is justified with the problem of creating a large evaluation set within the scope of this project. A good evaluation set needs to be created by several domain experts, and often requires a lot of time and resources to create.

7.4.3 Survey Data

It is hard for humans to classify topics on short texts, especially without the context of the text. In the survey it was also difficult to determine the context of the suggested topics. E.g. A concept that a human perceives as relevant might mean something else in the knowledge database, leading to lower scores for some algorithms that correctly not returned that concept. Another example of this is topic candidates that represent disambiguation pages in Wikipedia. They tend to get a high TF-IDF score, but a correctly lower score from the centrality algorithms. By just classifying the topics on the article name this might give an advantage to TF-IDF ranking.

Another example of the difficulties of topic annotation is from text 6, the text about bananas. Many of the suggested topics from the system was latin names for different species of bananas. Because of the lack of domain experts on bananas (or people accustomed with Latin) in the participating group, many marked these as not a matching topic even though the are highly relevant. This further shows the difficulty of assigning good topics to texts.

7.5 Conclusion

In this project, the use of search engine technology to extract topic candidates from text it has been evaluated. The results of using graph centrality, to improve the quality of the returned topics, has also been studied.

TF-IDF is a robust ranking and topic extraction method, when used with extensive background knowledge like Wikipedia. When looking at all of the top ranked proposed topic candidates, it is clear that using only TF-IDF has a tendency to fall short. The data from the survey shows that using graph centrality measures improves the ranking order to include other broader and relevant topics in the top ranked results. This leads to fewer bad topic suggestions by the system, than when using only TF-IDF. This answers the two first research questions of this project; ”Is it possible to use search engine technology to identify topic candidates for texts of varying length?” and ”Is a graph ranking method relevant and useful to rank possible topics?”. The recall from the search engine candidate selection method is comparable and sometimes even better then the results from a system like DBpedia Spotlight. The results from the survey clearly shows a positive trend when using graph centrality to rank the candidate topics.

The third research question: ”Can the system, given a text containing several different subtopics, correctly identify these?” can be answered by looking at the evaluations using the Simple Wikipedia datasets. It shows that the system is able to identify several different subtopics from text. It has been evaluated how well the system can distinguish and identify topics when each section of text contains a well defined topic. The recall for identifying the correct topic for the evaluation sections

is 85% on the larger Simple Wikipedia evaluation dataset.

Centrality measures works well on some types of text. The system seems to work best on texts where the topic is broader and more well defined in the background knowledge graph. Small specific topics without many related topics in the Wikipedia graph generally seems to be harder to classify. Both degree centrality and closeness centrality seems to be stable methods for improving the results. Both ranking algorithms get reasonably good results in both the tests using ElasticSearch and DBpedia spotlight for candidate selection. PageRank is also good, but the results are not as stable when varying the method of extracting topic candidates. PageRank performed significantly worse with the use of DBpedia spotlight candidate selection and when using ElasticSearch with the multi search settings. Betweenness centrality performs worse than the other centrality measures in almost all evaluation cases, and is therefor not recommended for use in this type of topic extraction system.

Even if the results are promising, the system is not ready to be used for automatic topic extraction and labeling. The system is only evaluated on a small dataset and both the results from the simple Wikipedia evaluation and the topic survey shows that the system is not precise enough for automatic labeling. The system returns to many unrelated concepts in the top 5 for it to be useful in automatic topic labeling. This conclusion concurs with the conclusions drawn in similar research in this area [Hulpus et al., 2013]. This answer to the fourth research question: "Is the result from the system relevant and useful enough to automatically create new topic metadata that could be used to improve findability in a search system?".

7.6 Future Work

A possible use of the system, that it is almost ready for, is to assist editors in the annotation process. The topics suggested seems to be relevant enough to be useful for editors. The final decision to use the suggested topics is then up to the editor. From the results so far, a good starting point for such a system would be to use the developed system with ElasticSearch candidate selection. Using the single search setting for the candidate selection, together with either degree centrality or closeness centrality ranking, seems to give the best results. When using ElasticSearch for candidate selection, it is recommended to use around 20 of the top ranked search results. This is based on that they produce more stable results on a variation of candidate graphs. To further improve the system, the ranking algorithm needs to consider the top ranked topic results from the previous section of text, when ranking topics. How this should be implemented needs further investigations. This answers the last research question: "Is the result from the system relevant and useful enough to use as a tool for editors, to speed up manual annotation of subtopic metadata?".

Wikipedia is a good start to use as background data, but a knowledge base adapted to the type of texts that the system is used on would most likely improve the results. It might be possible to create a small knowledge graph custom made for

specific types of text. E.g. To classify the topic of corporate documents, it might be possible to specify concepts unique to specific types of documents and then use this as a knowledge base for the system. This is also an area that needs further investigation.

To better understand subtopic classification better evaluation data is needed. Using data from Wikipedia has its setbacks, because it only consists of one specific type of texts. The development of a larger dataset consisting of several text types with topics annotated on paragraph level by domain experts would improve the possibility to get a deeper understanding on how to develop better topic models.

Bibliography

David M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, April 2012. ISSN 0001-0782. doi: 10.1145/2133806.2133826. URL `http://doi.acm.org/10.1145/2133806.2133826`.

D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.

Carlo Abi Chahine, Nathalie Chaignaud, Jean-Philippe Kotowicz, and Jean-Pierre Pécuchet. Context and keyword extraction in plain text using a graph representation. In Richard Chbeir, Albert Dipanda, and Kokou Yétongnon, editors, *SITIS*, pages 692–696. IEEE Computer Society, 2008. ISBN 978-0-7695-3493-0. URL `http://dblp.uni-trier.de/db/conf/sitis/sitis2008.html#ChahineCKP08`.

Kino Coursey and Rada Mihalcea. Topic identification using wikipedia graph centrality. In *HLT-NAACL (Short Papers)*, pages 117–120. The Association for Computational Linguistics, 2009. ISBN 978-1-932432-42-8. URL `http://dblp.uni-trier.de/db/conf/naacl/naacl2009s.html#CourseyM09`.

Kino Coursey, Rada Mihalcea, and William Moen. Using encyclopedic knowledge for automatic topic identification. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 210–218, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-29-9. URL `http://dl.acm.org/citation.cfm?id=1596374.1596407`.

DBpedia Project. The DBpedia Knowledge Base. ., 2013. URL `http://dbpedia.org/About`. Accessed: 2013-02-15.

Susan Feldman, Joshua Duhl, Julie Rahal Marobella, and Alison Crawford. The Hidden Costs of Information Work. Technical report, IDC, 05 2005.

Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. Unsupervised graph-based topic labelling using dbpedia. In *6th ACM International Conference on Web Search and Data Mining (WSDM'13)*, 2013.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

Olena Medelyan, Ian H. Witten, and David Milne. Topic indexing with Wikipedia. In *Proceedings of the Wikipedia and AI workshop at AAAI-08*. AAAI, 2008.

Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011*, pages 1–8, 2011.

Rada Mihalcea and Andras Csomai. Wikify!: linking documents to encyclopedic knowledge. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 233–242, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-803-9. doi: http://doi.acm.org/10.1145/1321440.1321475.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. URL `http://ilpubs.stanford.edu:8090/422/`.

Peter Schonhofen. Identifying document topics using the wikipedia category network. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 456–462, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2747-7. doi: http://dx.doi.org/10.1109/WI.2006.92.

Wikimedia Foundation. Wikipedia:about. ., 2013. URL `https://en.wikipedia.org/wiki/Wikipedia:About`. Accessed: 2013-02-20.

Chris Wild. The wilcoxon rank-sum test. ., 2010. URL `http://www.stat.auckland.ac.nz/~wild/ChanceEnc/Ch10.wilcoxon.pdf`. Accessed: 2013-05-10.

Baoyao Zhou, Ping Luo, Yuhong Xiong, and Wei Liu. Wikipedia-graph based key concept extraction towards news analysis. In *Commerce and Enterprise Computing, 2009. CEC '09. IEEE Conference on*, pages 121 –128, july 2009. doi: 10.1109/CEC.2009.54.