NTNU
Norwegian University of
Science and Technology

Graph-Based Representations for Textual Case-Based Reasoning

Kjetil Valle

Master of Science in Computer Science
Submission date: June 2011
Supervisor: Pinar Öztürk, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

This project will investigate different representations using complex networks for Textual Case-Based Reasoning (TCBR). In TCBR case texts are compared in order to retrieve previously encountered cases that are similar to the new situation. Comparing two cases in free text format is a challenging task because cases may use different wording and also because natural language is ambiguous, e.g. words may have more than one sense. Network structures provide an intuitive representation to capture the relations and structure in text and provide efficient inference mechanisms.

The student will first briefly review how complex networks have been used for representing text. Different types of relations and their possible representations as network edges, as well as various inference mechanisms, will then be studied. The representation models and inference mechanisms will be tested empirically, using various datasets.

Assignment given: 17. January 2011
Supervisor: Pinar Öztürk, IDI

Graph-Based Representations for Textual Case-Based Reasoning

Kjetil Valle
IDI, NTNU
kjetilva@stud.ntnu.no

Abstract

This thesis presents a graph-based approach to the problem of text representation. The work is motivated by the need for better representations for use in textual Case-Based Reasoning (CBR). In CBR new problems are solved by reasoning based on similar past problem cases. When the cases are represented in free text format, measuring the similarity between a new problem and previously solved problems become a challenging task. The case documents need to be re-represented before they can be compared/matched. Textual CBR (TCBR) addresses this issue.

We investigate automatic re-representation of textual cases, in particular measuring the salience of features (entities in the text) towards this end. We use the classical vector space model in Information Retrieval (IR) but investigate whether graph-representation and salience inference using graphs can improve on the Term Frequency (TF) and Term Frequency-Inverse Document Frequency (TF-IDF) measures, *bag of words* approaches predominant in IR.

Our special focus is whether, and possibly how, the co-occurrence and the syntactic dependency relations between terms have an impact on feature weighting. We measure salience through the notion of graph centrality.

We experiment with two types of application tasks, classification and case retrieval. Although classification is not a typical TCBR task, it is easier to find datasets for this application, and the centrality measures we have studied are not specific to TCBR. The experiments on this task are therefore relevant to the second application task which is our ultimate target.

We test various centrality metrics described in the literature, make a distinction between local and global weighting measures and compare them for both application tasks. In general, our graph-based salience inference methods perform better than TF and TF-IDF.

Preface

This thesis describes the work done during my last semester at the Norwegian University of Science and Technology (NTNU). It has been carried out at the Department of Computer and Information Science during the period from January 2011 to June 2011. Parts of the thesis have also been submitted as a paper to the *India-Norway Workshop on Web Concepts and Technologies 2011*.

I would like to thank my supervisor during the period, associate professor Pinar Öztürk, for invaluable guidance throughout the project. I always felt inspired and confident after our meetings. Her feedback to my work was thorough and detailed, and I never had to wait long for answers to my questions.

I would also like to direct my thanks to PhD candidate Gleb Sizov for our discussions on various topcis and ideas, and for lending me the computational power needed for some of the heavier experiments.

Finally, my thanks go to fellow students Olav Bjørkøy, Øystein Carlsson and Marvin Lillehaug. I have valued their companionship, both academically and socially, through the project, and our discussions of everything from the best ways to express ideas to technical LaTeX-related questions have been helpful.

Kjetil Valle
Trondheim, June 2011

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Case-Based Reasoning (CBR) is a problem solving technique well suited for many types of problems. It is closely related to the way human beings solve problems, namely by remembering old problems and reusing solutions. This way of learning from experience is inspired by models from cognitive psychology about how humans reason, and has been proven to work very well in a wide range of domains (Leake, 1996).

CBR works on the assumption that similar problems have similar solutions. Experiences, i.e. previously solved problems, are stored as *cases* in a case base. For each new problem, the most similar cases are retrieved from the case base. When the best match is found, its solution is either used directly, or adapted to fit the new problem. After the new problem is solved, it too can be added to the case base to be used for problem solving in the future.

Despite CBR's proven success in many areas, it has not been widely adopted in the industry. The reason for this is, most likely, something at the very core of CBR: the *structured representation* of cases. Cases in classical CBR are represented as a collection of features organized as anything from attribute-value vectors to rich structured representations. The features and their organization represent important aspects of the case. They are used for reasoning processes such as computing the similarity between new and old problems, adapting old solutions to new situations, or explaining why a particular solution solves a given problem.

In the industry, most knowledge and experience is recorded as *textual documents*. To use these as cases in CBR, features must be extracted from the text. In traditional CBR, this has typically been done manually. This requires a knowledge engineer to review all the information, consult domain experts, identify the aspects of importance, and then build the case representation. In most situations this scales very poorly, since the number of available documents grows faster than cases can be hand-crafted. The process is, of course, also very costly. There is therefore clearly a need for automated methods for feature extraction.

Systems that attempt to do this are called Textual CBR (TCBR) systems. TCBR systems try to automatically extract from the text those parts that are central to the problem. These parts must then be stored in suitable case representations. The two central challenges in TCBR are to find ways to extract appropriate features from the text, and representations that are suited for the reasoning processes required in CBR.

An often used approach, borrowed from the field of Information Retrieval (IR), is

the so-called *bag of words* (BoW) model. In representations under this model, terms represent possible features, and their frequencies in the text are used as weights indicating their salience. In what is known as the Vector Space Model (VSM), the term weights constitute the elements of vectors representing the documents, which can be compared using a cosine measure. Although this method is frequently used due to its simplicity, we do not believe it is satisfactory for TCBR for two reasons:

- The BoW model throws away all information about the structure of the text, such as which words occur together and how they relate to each other.
- The frequency measure can often be a poor indicator of how central a term is to the topic discussed in a document.

This thesis is motivated by the need for better models for text representation. We propose a new approach to text representation, based on graph structures, to alleviate these problems. Graphs are ideal for representing relations between features, and should thus be able to retain some information about the internal structure of the documents. Graphs also enable new methods for identifying important aspects of the text to be used, such as graph centrality measures.

1.1 Goals and Objectives

In this thesis, we intend to explore how text can be represented using graphs, and how well such representations are suited as TCBR cases. Specifically, we will focus on the first step of the CBR process, the *retrieval phase*. In this phase, previously stored cases similar to the new problem are retrieved from the case base, for subsequently to be used to solve the problem at hand. In order to get the best suited solution, it is essential that the best possible candidate case is retrieved, and in order to find this case a good method for measuring similarity of cases is required.

We summarize our research goal as follows:

> *We intend to investigate and evaluate various graph-based representations for use in TCBR, with specific focus on case similarity measurement. The representations will capture the inherent structure of documents by retaining relationships between terms, and use graph centrality measures to weight terms as features. Our study will evaluate different text graph representations and a wide range of centrality measures.*

Our representations will also take the form of term-vectors, similar to BoW representations, except that document structure is also taken into consideration. Thus, our new representations are technically term weighting schemes for the VSM. Text networks will be used as intermediate representations, and term weighting will be based on the node centrality of the terms in these. The idea underlying this approach can be formulated as our first hypothesis.

Hypothesis 1: Graph centrality forms a better basis than term frequencies for term weighting.

In IR, a measure known as Term Frequency-Inverse Document Frequency (TF-IDF) is often used as an improvement over simple Term Frequency (TF) for term

weighting. TF-IDF incorporates information about the frequencies of the terms among all the documents, in addition to their frequencies within the relevant document. We will similarly develop a weighting measure using node centrality over the entire corpus, in order to test our second hypothesis.

Hypothesis 2: Including information about overall term centrality in the corpus will improve the term weighting.

We intend to explore two network models: *co-occurrence networks* and *dependency networks*. Co-occurrence networks have predominated most of the research done on text network representations for document similarity measurement. The reason for this is likely that co-occurrence networks are rather simple, both to implement and conceptually. Dependency networks have, however, to the extent of our knowledge, not previously been used for the purpose of computing feature weights. The Dependency networks are based on parsing of natural language using dependency grammars, and is thus more restrictive in the relations captured by the networks. Whereas co-occurrence networks consider all co-occurrences relevant, only those fitting predefined structural relationships are included in dependency networks. This is the basis for our third hypothesis.

Hypothesis 3: Since dependency networks are based on more structurally defined relationships between terms than co-occurrence networks, they will be better able to capture meaningful relations and thus better facilitate document similarity measurement.

Although neither of the network models are novel as far as text representation is concerned, we will evaluate both in more detail than have previously been done with focus on document representation. We will also experiment with a wide range of centrality measures, covering all the main groupings, in order to find those best suited for term weighting.

In order to evaluate the performance of our representations in facilitating document similarity measurement, we define and use two evaluation tasks. The first is a classification task based on nearest-neighbour classification, and the second is a retrieval task inspired by TCBR retrieval, albeit with a somewhat simplified evaluation criteria. These tasks will be used both while developing the network representations, and in the final experiments comparing them to each other and to frequency-based BoW representations.

1.2 Limitations

While the previous section outlined the goals of the thesis, it is equally important to specify clearly what we will *not* try to do.

First, although the work is motivated by TCBR, and aims to study better case retrieval methods in particular, we will not develop a working CBR system. We focus on methods for comparing documents, or more specifically comparing a new case query with past case documents, using graph-based representations. It is a long way from this to realizing the idea of automatically training CBR systems on

collections of textual documents, and this work should only be seen as the first steps towards such a goal.

Second, although we hope to develop good network representations for text, we do not aim to apply natural language processing in order to actually understand the contents of the documents. Such understanding, and the possibilities it would open for the representations, would certainly be beneficial, but is beyond the scope of this thesis.

Finally, it is important to note that even though we seek representations of text suited for use in TCBR, only one aspect is evaluated in this study, namely how well they facilitate similarity measurement. The study does not, and does not intend to, say anything about the usefulness of graph-based cases in other aspects of TCBR.

1.3 Outline

The thesis starts with two chapters presenting background theory serving as both a foundation and a motivation for the work described in the rest of the thesis. Chapter 2 presents TCBR as a problem solving technique, and describes some of the issues related to representation and document similarity assessment. Graph theory and complex networks are subsequently the focus of Chapter 3, where several models and concepts are described.

In Chapter 4 we move on to look at how graphs can be used to represent text. After a review of related research on text networks, we present our approach to the problem. This chapter also describes the evaluation methods and the experimental framework. The datasets used throughout Chapters 5 and 6 are also presented here. Chapters 5 and 6 explore the co-occurrence network representations and dependency network representations, respectively. Each of the chapters study various aspects of the respective representation, using the evaluation methods introduced in Chapter 4.

The representations are compared to each other and to BoW models in Chapter 7. A case study of terms considered important by the representations opens the chapter, before the comparative experiments are described. The evaluations done here also employ the methods presented in Chapter 4, but are performed on new datasets. Chapter 8 discuss the results of the experiments, and summarize the main findings and conclusions. The final chapter presents ideas and possibilities for further work.

There are two appendices. The first describes our implementation of the graph-representations and experiments. In the second appendix, detection of power-law distributions in graph data, and our search for these in our representations, is discussed.

Chapter 2

Textual Case-Based Reasoning

This chapter presents background theory about Case-Based Reasoning (CBR), and focuses in particular on Textual CBR (TCBR). CBR as a problem solving technique, and its process of solving problems, is first described. TCBR is then introduced, and the role of text in CBR explained. We look at the motivations for using textual representations, and discuss the relationship between TCBR and Information Retrieval.

There is of course much more to CBR and TCBR than can be covered in this rudimentary description. The aim of the chapter is merely to provide an introduction to TCBR as a problem solving method, and motivate the search for new and better representations for cases in TCBR.

2.1 Case-Based Reasoning

Case-Based Reasoning (CBR) is an approach to problem solving motivated by reuse of solutions to previously solved problems. When a new problem is encountered, the set of previously solved problems is examined, and the most similar problems are retrieved. If a sufficiently similar problem has been encountered before, that solution might be reused directly. Otherwise, solutions from one or more of the old problems might be adapted to the new situation in order to create an adequate solution. Once solved, the new problem can be stored for use in solving of new problems at a later time.

In CBR terminology, the set of previously solved problems are called the *case base*. The case base consists of *cases*, which are problem instances consisting of a *problem description* part and a *solution* part. The new problem to be solved is called the *query case*, or simply the *query*. The query consist of only a problem description. The goal of CBR can be stated as follows:

> *Presented with a query problem, derive based on the set of cases in the case base a solution that solves the query.*

CBR is in a narrow sense a reasoning technique, comparable to, for example, deductive inference. In a broader sense it can also be seen as a methodology for designing and implementing computer systems that realize the principles of this reasoning technique.

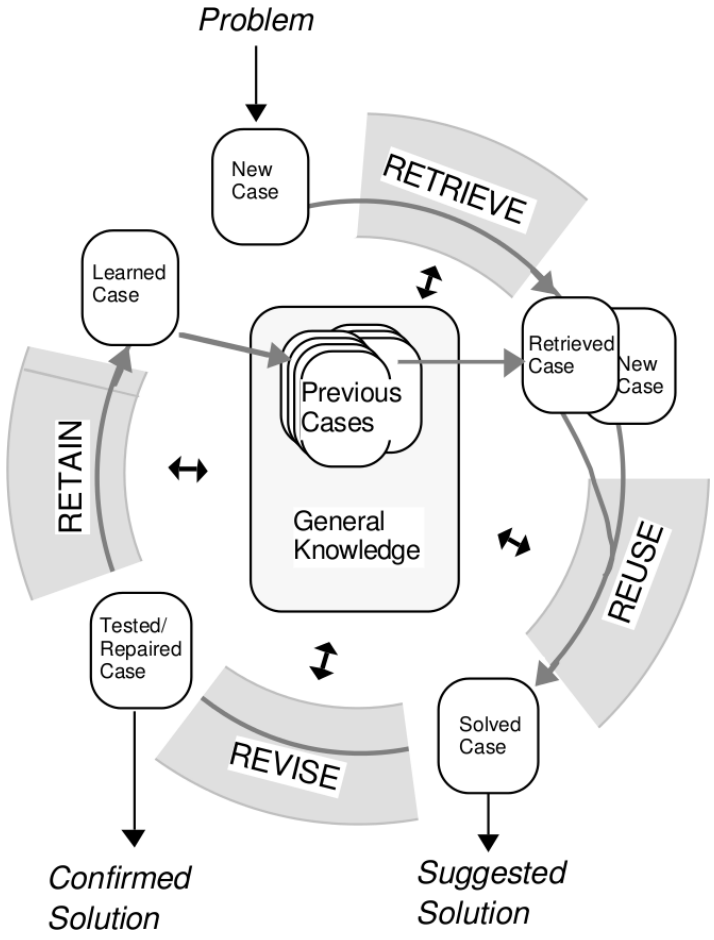


Figure 2.1: The CBR cycle. From Aamodt and Plaza (1994).

2.1.1 Assumptions in CBR

CBR, as a reasoning technique, makes certain assumptions about the nature of problem solving. According to Leake (1996), there are two tenets that form the basis of CBR. These are

1. The world is regular: similar problems have similar solutions.
2. The types of problems that an agent encounters tend to recur.

The consequence of the first tenet is that solutions for similar problems are good starting points for new problem-solving. A result of the second tenet is that future problems are likely to be similar to current and past problems. Thus, when the two tenets hold, it should be useful to remember and reuse reasoning, making case-based reasoning an effective reasoning strategy.

2.1.2 The CBR Cycle

The process of CBR is best described using the *CBR cycle*, a process of four steps introduced by Aamodt and Plaza (1994). The cycle is illustrated in Figure 2.1. The steps, often referred to as *the four REs*:

Retrieve: The case or cases most similar to the query is retrieved from the case base. What is meant by similarity among cases depends on the domain and how cases are represented.

Reuse: Information and knowledge from the retrieved case(s) is used to solve the problem. Solutions may in the simplest case be reused directly, and in more complicated scenarios be adapted to meet criteria specific to the new problem. Both actual solutions and the methods by which the case solutions were derived, depending on what is stored in the cases, may serve as the basis for reuse.

Revise: The proposed solution is tested against the problem, and revised if necessary. Possible ways to test the solution is to apply it in the real world environment or to have an expert evaluate it. This step could be implemented as part of a CBR system, or could be a process external to the system.

Retain: The query case with the newly created solution is stored with the rest of the cases in the case base. This way new experiences are retained with the old knowledge, and can contribute to future problem solving.

2.1.3 Motivation for CBR

CBR is obviously inspired by the way humans solve problems — not by arriving at every conclusion by logical reasoning, but by remembering what happened in similar circumstances in the past. That people do not solve every new problem from scratch, but base decisions on previous experiences, have a sound basis in cognitive science. According to Roger Schank, a cognitive scientist at Yale University, people are most of the time too lazy to think. Instead we base our actions on memories of what we have thought or done before. Based on this idea of *reasoning by remembering*, with the goal of replacing reasoning with the recall and adaption of episodic memories, Schank and his colleagues established the foundations of CBR during the early 1980s.

CBR is an example of a lazy problem solving technique, which means that no attempt is made to generalize or learn anything before an actual problem instance needs to be solved. Reasoning is thus demand-driven and happens only when information about the specific problem is available. Eager learning, conversely, tries to construct a general, input independent target function during training of the system. While an eager learners must create global approximations to problems, lazy learner can create local approximations that corresponds to actual queries.

2.1.4 Case Representations

A CBR system is heavily dependent on the way its cases are represented. The case representation needs to enable the system to easily reason with concepts in the problem domain, and facilitate efficient execution of each of the four REs. Aamodt and Plaza (1994, p.9) define the problem as follows.

> *The representation problem in CBR is primarily the problem of deciding what to store in a case, finding an appropriate structure for describing case contents, and deciding how the case memory should be organized and indexed for effective retrieval and reuse.*

The case contents, and the way in in which to structure them, chiefly depend on the type of problems to the system is intended to solve. For example, in a medical diagnosis system typical case features would include symptoms of the patient and results of various tests.

As for the memory models, there are many ways to organize the case base. Two early influential memory systems are the Dynamic Memory model of Schank and Kolodner and the exemplar-based model of Porter and Bareiss. The Dynamic Memory model (Kolodner, 1993) is based on the idea that specific cases which share similar properties are organized under a general structure: a *generalized episode*. A generalized episode holds features common to its set of cases, and contain indices pointing to more specific generalized episodes or directly to cases. The exemplar-based described by Porter et al. (1990) refer to cases as *exemplars*. In this model, the case memory consist of a network structure of categories, semantic relations, cases, and index pointers. Cases are associated with categories, and ordered by their degree of *prototypicality* for the category. Indices link cases, features and categories together, describing their relations and differences. Categories are interlinked by semantic relations, which contain the general domain knowledge of the model.

Cases are usually divided into problem descriptions and solutions, either explicitly or indirectly as part of the same case structure. Representations vary from relatively simple feature-value vectors, to complex data-structures specially tailored to a particular domain.

2.2 Textual CBR

Traditional CBR, as described in the previous section, rests on the fundamental idea of solving problems by collecting experiences from earlier problem solving episodes. In real life, many of the most valuable experiences are stored as textual documents, such as medical records, technical manuals and documentations, collections of frequently asked questions (FAQs), memos, reports, and informal notes. It is thus only natural to consider how these resources can be used from a CBR perspective, which is done in the field of TCBR.

The idea of using textual documents as problem experiences is tempting because they are so readily available. Traditionally, structured CBR cases have in many cases been demanding to construct for various domains, and required a lot of effort from knowledge engineers. This, perhaps most of all, has been the limiting factor in the adoption of CBR systems in the industry.

TCBR is concerned with figuring out how textual information can be used in CBR, either directly or by creating structured cases automatically from documents. The field is young, and many questions still remains unsolved.

2.2.1 Assumptions Made in TCBR

Working with documents, there are some assumptions that are commonly made in TCBR. The following four are identified by Mustafaraj (2007, p.47) as the most common:

1. Each case has at least a problem description and a problem solution part.

2. One document is equal to one case.
3. A document is a bag of unordered words.
4. One relevant document is sufficient if it answers a query.

These assumptions are by no means universal, but are often accepted because they simplify both the development and evaluation of TCBR systems. Some researchers, among them Mustafaraj herself, deliberately break with some or all of them. Mustafaraj and Freisleben (2006) argue that an event-oriented perspective is needed in modelling of cases. They argue that the prevailing object-oriented perspective is caused by the influence of technical documents on current TCBR research, and that objects are suited as features for retrieval but does not serve well for reasoning.

We shall, in the representations presented in Chapter 4, break with the third assumption by incorporating information about document structure.

2.2.2 Cases from Text

One of the key decisions in TCBR is what to do with the textually recorded experiences, the collection of documents. These must in some way be made to represent cases in a form the system is able to work with, and the way this is done has a huge impact on the TCBR system.

> *In TCBR, the representation of the text source is key because it is used as the basis for computing the similarity between cases, which ultimately determines which cases are retrieved. (Cunningham et al., 2004, p.581):*

As we see it, there are two main approaches to constructing the cases. The first, and perhaps seemingly the simplest, is to store the documents directly in the case base. Alternatively, features can be automatically extracted from the text sources, and complex structured cases as traditionally used in CBR can be created.

The first approach have the advantage of not having to do complex feature extraction from natural language, which is a difficult task. Consequently, it is imposed restrictions in which kinds of reasoning it is possible to do with the problem cases. These approaches really represents the extremes of a spectrum where most existing systems fall somewhere in between the two as intermediate solutions.

2.2.3 Aspects of Written Text

There are many challenges related to using text as a basis for case representations. One of these is that the vocabulary used in the documents depends on their domain. Because of this, it is often difficult to find available resources that structure and define a vocabulary in an organized way. Thus, many resources, such as WordNet[1], prove too general for practical use in TCBR. Another source of difference in vocabulary used in documents is the different users participating in their creation. Both level of knowledge and experience and writing style can influence the way information is expressed in documents.

[1] `http://wordnet.princeton.edu`

Another problem is that it can often be difficult to distinguish the parts of the text that constitute the problem description from those representing the solution. This is related the explicit structure of the documents, which may differ between different document types, and sometimes between documents in the same document collection. If the relevant documents have a specified, or sufficiently consistently, structure it may be possible to use this information when constructing cases. In the FAQFINDER system, for example, Burke et al. (1997) use the inherit structure in question-answer pairs as problem descriptions and solutions, respectively.

Also the grammatical quality of written text can vary greatly. Some types of documents are usually well written in grammatically correct language, while others may consist of keywords, incomplete sentences, or simply contain many misspelled words and lack punctuation and capitalization. The degree of grammaticality influence the degree in which TCBR is able to utilize NLP techniques. Statistical NLP typically use elements such as punctuation, capitalization, or part-of-speech, and in the absence of these researchers are confined to the use of simpler representations such as n-grams.

2.2.4 TCBR and Information Retrieval

Due to the concern with textual documents, TCBR is naturally closely related to Information Retrieval (IR). Particularly, many methods used in IR also prove useful in TCBR. The Vector Space Model (VSM) in particular, which we describe shortly, has been used in much TCBR research.

The similarities are based on the shared focus on textually represented information. Both require methods for representing and processing text. Especially essential for IR is measurement of similarity between documents, and this is also important for TCBR if text is used in the case representation.

There are, however, also fundamental differences between TCBR and IR. The main difference lies in the domain specific focus of TCBR. A TCBR system is supposed to solve problems within one domain, and is usually not required to be applicable to problems in other domains. One of the central ideas in CBR is that domain knowledge should be used to find good candidate cases and construct the best possible solution. IR techniques, conversely, is expected to be largely domain independent.

Another difference between the two is that IR is, as the name implies, purely a retrieval process. CBR, as discussed previously, include retrieval as one of several steps. The goal of a CBR system is not merely to identify and return to the user one or more documents relevant to an information need, but to find a solution to a problem.

For good discussions and comparisons of TCBR and IR, consult Lenz et al. (1998) and Lenz (1998).

The Vector Space Model

The VSM is an important representation technique widely used in IR. Under the VSM, documents are represented using vectors in a n-dimensional feature space with n equal to the size of the vocabulary. Each dimension thus corresponds to a

unique term, and the document vectors are weighted according to the frequencies of terms in the documents.

Different weighting schemes can be used to create the weights. The most commonly used are *term frequency* (TF) and *term frequency-inverse document frequency* (TF-IDF). The TF is the frequency of a particular term in a specific document. It is measured as the number of occurrences of the term t in document d, the *term count* $\text{TC}_{t,d}$, divided by the total number of words in d,

$$\text{TF}_{t,d} = \frac{\text{TC}_{t,d}}{|d|} \tag{2.1}$$

The TF measure has a tendency to value highly some terms that are common in many documents, and thus might describe the particular document d poorly. These are typically small function words such as *the*, *and*, *of*, etc. The TD-IDF measure takes this into account, and give higher weights to terms that are frequent in document d but occur in few other documents. This is achieved by multiplying the TF value by the inverse of term t's *document frequency* (DF). DF_t is defined as the fraction of documents that contain term t. The inverse document frequency IDF_t is obtained by taking he logarithm of the inverse of DF_t. The TF-IDF measure is thus found as

$$\text{TF-IDF}_{t,d} = \text{TF}_{t,d} \times \log\left(\frac{|D|}{|\{d \in D \cap t \in d\}|}\right) \tag{2.2}$$

where D is the collection of documents.

When documents are represented as vectors in this way, a multitude of vector similarity metrics can be applied to calculate the distance between them, or equivalently their similarity. The most commonly used is the *cosine similarity* measure, which simply is the cosine of the angle between two document vectors.

$$\text{sim}_{cos}(d_1, d_2) = \frac{\vec{v}(d_1) \cdot \vec{v}(d_2)}{|\vec{v}(d_1)|\,|\vec{v}(d_2)|} \tag{2.3}$$

Benefits of this similarity measure is that two identical documents will get a similarity of 1.0, and two documents that share no common terms at all will have a similarity of 0.0. The measure also ignores the norm of the vectors, which makes it suited to compare documents of different lengths.

2.3 Summary

This chapter has presented CBR, a reasoning technique and problem solving methodology useful for many types of problems. CBR finds solutions to new problems based on a body of previous experiences — the case base. The ability of a CBR system is directly dependent on the number and quality of the experiences that constitute this case base.

One of the major challenges with traditional CBR is the construction of the cases. In classic CBR systems, cases have typically been structured consisting of features from the problem domain. Such cases require much effort in the form of knowledge engineering, which might be why the use of CBR is not more widespread.

One of the best sources of problem solving experiences are textual documents. People and organizations regularly make notes, records and reports about episodes useful as problem cases. The use of such documents in CBR is the focus of the field of Textual CBR (TCBR).

TCBR share many methods and representations with Information Retrieval (IR), including the Vector Space Model and term frequency vectors, but is also fundamentally different. While IR focuses on retrieving specific nuggets of information as a response to an users information need, TCBR is focused on solving specific problems. The goal in TCBR is not merely to retrieve information, but to reuse and adapt the retrieved information into something that forms the solution to a given problem.

Chapter 3

Graph Theory

This chapter presents the theory of graphs and networks. The aim is to provide an understanding of the theoretical foundations of networks and their uses, before their application to text representation is discussed in the following chapter.

Section 3.1 starts the chapter by briefly explaining the basic concept of graphs. Section 3.2 then builds upon this, introducing the field of network theory. What networks are, how they can be used, and some of their most important properties are described here. In Section 3.3 a special focus on Graph Centrality is made, which will prove a central component in the representation presented in Chapter 4. Section 3.4 discuss how networks relate to and can represent natural language text in different ways. A brief summary concludes the chapter.

3.1 Graphs

A graph is an abstract mathematical representation. It consist of a collection of objects, called *nodes* or *vertices*, and a collection of relations between these, the *edges*, *arcs* or *links*. Formally a graph G is an ordered pair $G = (V, E)$, where V is the set of vertices and E the set of edges connecting these.

We will be using the terms *node* and *vertex* interchangeably. The term *edge* will be preferred for the relations, and we say that a node *links* to another if there is an edge from the former to the latter.

Although nodes may represent various entities (more on this in the following section), they are treated as featureless and indivisible in the graph. The *order* N of a graph is the number of nodes $|V|$. Edges represent abstract relations between the nodes. They may be either directed or undirected, and may or may not be weighted. The *size* of a graph is defined as its number of edges.

The number of edges connected to a node is known as its *degree*. If the edges in the graph are directed, the *in-degree* and *out-degree* is defined as the number of incoming and outgoing edges, respectively.

Two nodes are said to be *connected* if there is a *path* between them. A path is a sequence of nodes where each node links to the next node in the sequence. A *connected graph* is one in which every pair of nodes are connected.

The path between a pair of nodes with the least number of intermediate nodes is called their *geodesic path*. In a graph with unweighted edges, this also constitute the shortest path; with weighted edges, the shortest path is the path where the sum

of edge weights have the lowest value. The average geodesic path length between node pairs in the graph is called the graph's *characteristic path length*.

Two nodes are said to be *adjacent* if they are connected by a single edge. The *neighborhood* of a node is the set of adjacent nodes. A graph where all nodes are adjacent is called *complete*, or *fully connected*. Such a graph contains all the $N(N-1)/2$ edges possible (in an undirected graph), where N is the order of the graph. In a complete graph, all node pairs have geodesic paths of 1.

An *adjacency matrix* is often used to represent a graph. Each row and column in the adjacency matrix represents a node. The value of element a_{ij} in the matrix is 1 if node i is connected to node j, and 0 otherwise. In a weighted graph, a non-zero value of a_{ij} describes the weight of the edge.

Graphs are mathematical abstract representations, used to solve a wide range of problems in graph theory, a field in discrete mathematics. The origin of the concept is the well-known problem of the Seven Bridges of Konigsberg, where Leonhard Euler, in 1735, used the properties of a graph representation of the city to prove that it was impossible to traverse the seven bridges of the city without crossing any of them twice.

Graphs are still used to represent a wide range of interconnected systems of things or people. We call such systems, where physical entities are represented by graph models in order to better understand them and their properties, networks. This is the focus of the following section.

3.2 Networks

Network theory can be seen as a part, or rather an application, of graph theory. It is concerned with the study of complex systems that is composed of entities and relations. Anything that can be seen as a set of discrete interrelated objects can be modelled using network theory.

Networks are highly useful, both in social sciences and for studying natural phenomena. Examples of applications of network theory include semantic networks, neural networks, computer networks, and social networks of many kinds, to name but a few. The study of network theory is a very active and growing field. In the words of Barabási (2003, p.222):

> *Network thinking is poised to invade all domains of human activity and most fields of human inquiry. It is more than another helpful perspective or tool. Networks are by their very nature the fabric of most complex systems, and nodes and links deeply infuse all strategies aimed at approaching our interlocked universe.*

Albert and Barabási (2002) explain the recent boom in network research with certain technical and social developments. On the technical side there is the rapid increase in available computational power. New technology enables both computerized data acquisition leading to large databases of real networks, and the computational power to explore such networks containing millions of nodes. The technical development has been aided by weakening boundaries between disciplines, enabling researches to access and study a range of different networks, and a heightened focus on understanding systems as a whole.

The understanding of whole systems, rather than trying to understand systems by studying individual components, is also motivated by the sheer complexity of many systems. Because most systems studied are very large, often with millions of nodes or more, it becomes problematic to investigate the details of the network separately. To deal with this, probabilistic models for representing large networks have been devised.

The following sections describe the three most prominent such probabilistic models. Parts of these sections have previously been presented as part of our earlier research (Valle, 2010). First described is *Random Graphs*, the model that initialized the field of network theory and still influence our thinking about how to represent large networks. It is based on networks emerging when nodes are randomly connected with a given probability. The *Small-World Networks* model is then introduced. This model solves some problems that cannot be represented by simple random graphs, such as short characteristic path lengths and high clustering coefficients found in many real systems. Finally the model of *Scale-Free Networks* is discussed. The power of this model is that it is able to describe why many networks display degree distributions that follow power-laws, leasing to the presence of hub nodes.

3.2.1 Random Graphs

The theory of random graphs was first described by Erdős and Rényi in the late 1950s. Until this time, graph theory concerned itself with smaller graphs with specific structures, and their properties. Many networks are too large for us to be able to represent or describe them completely and explicitly. Random graphs introduced *probabilistic models* into network theory, and enabled us to represent such networks by means of local rules without considering a complete global structure of the network.

Random graphs, as first described by Erdős and Rényi (1960), consists of N vertices, each pair connected with a probability p. The result is a graph with approximately $pN(N-1)/2$ edges randomly distributed in the graph. This simplest form of the random graph model is named after its inventors, *Erdős-Rényi random graphs*.

The connectivity of a graph is described by its degree distribution. This is a function $P(k)$, defining the probability that a given node in the network have degree k. In a random graph, this function follows a Poisson distribution. The average degree of a node in a random graph is $\langle k \rangle = pN$.

An interesting and well known property of random graphs is the emergence of *giant components*. While the probability of edges p is low, the network consist of small isolated graphs, typically trees. At a critical probability $p_c \simeq 1/N$ the average degree $\langle k \rangle$ approaches 1. This means that every node on average is connected to at least one other. When this happens the network drastically changes its characteristics. All the small previously disconnected graphs then starts connecting to each other, forming a giant component within the network. It is also found that if $\langle k \rangle \geq ln(N)$, almost every graph is connected (Albert and Barabási, 2002). This is useful because it allows us to predict whether any two nodes will be connected in a network by only estimating the average degree of the network.

Random graph models, while obviously simplifications of how real networks behave, have guided the thinking about large networks, and serve as a good ap-

proximation to modelling some systems. While many real world networks are more complex than the Erdős-Rényi random graphs, they serve well as a baseline to which networks can be compared.

3.2.2 Small-World Networks

The *small-world model* was first presented by Watts and Strogatz (1998). They noted that the structure of most real world networks lies somewhere in between the completely regular and completely random. Some such networks can be seen to have both *(1)* a high clustering coefficient C, much larger than that of a comparable random graph with the same number of nodes and edges, and *(2)* a short average length of the shortest path between pairs of nodes, i.e. characteristic path length, comparable to that of a corresponding random graph. They called these systems small-world networks. Such systems display enhanced signal-propagation speed, computational power and synchronizability.

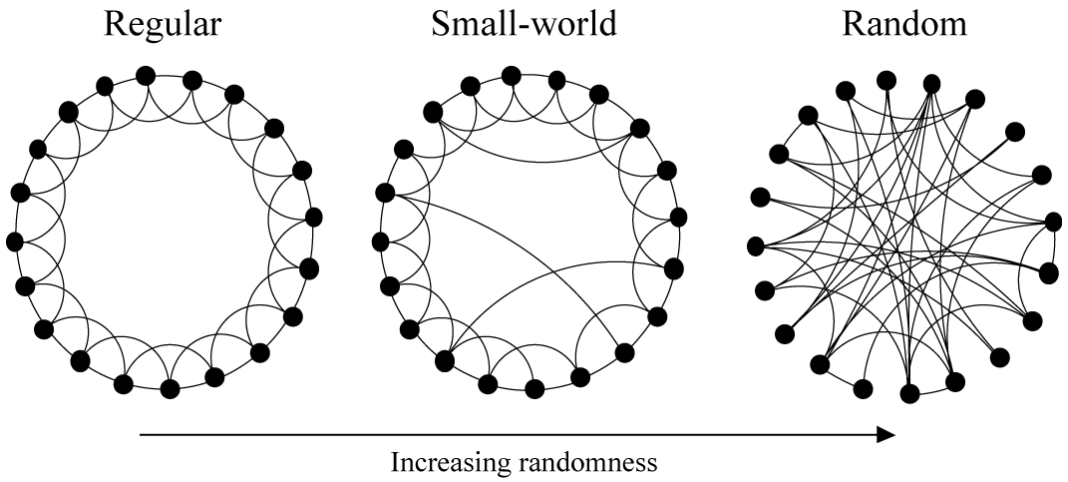


Figure 3.1: The types of networks explained by the Watts-Strogatz model (Watts and Strogatz, 1998).

The model proposed by Watts and Strogatz enables us to explain the clustering observed in many real systems. Just like the Erdős-Rényi model, it is fundamentally random. The difference is that instead of adding edges randomly to a set of nodes, the network initially starts out as completely regular. Edges are then rewired with a probability p, equivalent to that of adding new edges in the random graphs, thus introducing randomness into the network. The discovery made by Watts and Strogatz was that even a handful of such rewired edges drastically reduce the average distance in the network, while they hardly affect the clustering at all.

The two properties defining such networks, *small worlds* and *clustering*, are described subsequently. For a more detailed analysis of the properties of small-world networks, see (Albert and Barabási, 2002, Section VI).

The Small-World Phenomenon The concept of the *small-world phenomenon* was initiated by the observation that it is often a surprisingly short average minimum distance between pairs of vertices in graphs. This was first noted by Stanley Milgram (1967) in a study examining the average path length in social networks. The

study concluded that there was a typical length of about six links between pairs of people in the United States. This effect is popularly known as the *six degrees of separation*. Such small-world effects have later been shown both in nature and in man-made systems.

Even though the idea of six degrees of separation originated with Milgram's work, he never used the term. It was first used by Hungarian writer Frigyes Karinthy and popularized in a play with the same name by John Guare (1990). The game of *six degrees of Kevin Bacon* is based on the concept. The goal is here to find a link from any actor to Bacon within six steps, where actors are considered connected if they have starred in the same movie. Another version of the game is the *Erdős number*, which is the number of coauthor relations needed to connect a researcher to Paul Erdős.

The effect seems to be true for most kinds of complex networks, and is also evident in random graphs. Newman (2001, p.4) explains this intuitively:

> *In almost all networks, the number of kth nearest neighbors of a typical vertex increases exponentially with k, and hence the average distance between pairs of vertices l scales logarithmically with N the number of vertices.*

Since this property is found to hold true, also in random graphs (Watts and Strogatz, 1998; Newman, 2001), the small-world phenomenon cannot be an indication of any particular organizing principle. Rather it is an important property for describing the behaviour of a network. It was hypothesized by Jeong et al. (2000) that the small-world phenomenon make networks robust, and that this is the reason why many biological networks show this property.

Clustering A common property of networks is the formation of clusters. One of the most familiar examples of this is circles of friends and cliques in social networks. These are groups of people within which almost everyone knows almost everyone else. The tendency of a network to contain such clusters is described by its *clustering coefficient*.

For a given node i in the network, there are e_i edges connecting it to other nodes. If these nearest neighbors constituted a complete subgraph in the network there would be a total of $e_i(e_i - 1)$ edges between them. The clustering coefficient of node i is defined as the ratio of actual edges in this subgraph, E_i, to the total possible. Thus, the clustering coefficient is defined as

$$C_i^{\text{directed}} = \frac{E_i}{e_i(e_i - 1)} \tag{3.1}$$

With undirected graphs, the number of possible edges in the subgraph is only half, and the equation is adapted correspondingly:

$$C_i^{\text{undirected}} = \frac{2E_i}{e_i(e_i - 1)} \tag{3.2}$$

For the network as a whole, the clustering coefficient is defined as the average of that of each node.

$$C = \frac{1}{N}\sum_{i}^{N} c_i \tag{3.3}$$

where N is the number of nodes in the network.

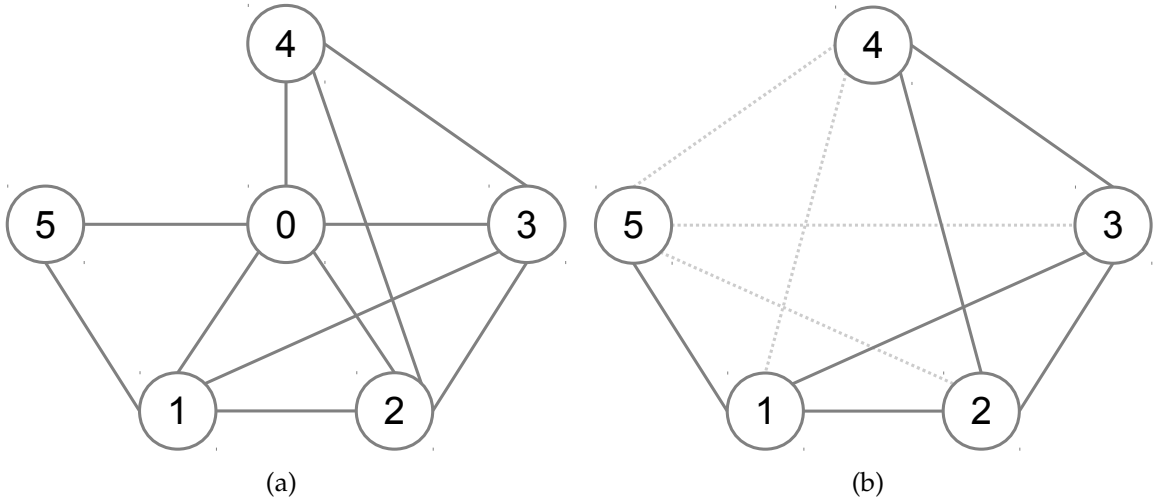


Figure 3.2: Illustration of graph clustering. (a) shows a small example graph. (b) shows the $e_0 = 5$ neighbors of node 0, and their connectedness, with dotted lines indicating missing connections. Node 0 have clustering coefficient $C_0 = 6/10 = 0.6$. The rest are $C_1 = 0.66$, $C_2 = 0.83$, $C_3 = 0.83$, $C_4 = 1.0$, $C_5 = 1.0$.

Figure 3.2 exemplifies the calculation of clustering coefficients in a small undirected graph. The number of possible edges between the nodes in 3.2(b) is 10, while the number of actual edges $E_i = 6$. This gives node 0 in 3.2(a) a clustering coefficient of $C_0 = 0.6$. By Equation (3.2), the rest of the nodes have clustering coefficients $C_1 = 0.66$, $C_2 = 0.83$, $C_3 = 0.83$, $C_4 = 1.0$, $C_5 = 1.0$, which gives the network an average clustering coefficient of $C = 8.2$.

Since the probability of two neighbors of a node in a random graph being connected is the same as the probability p that any random pair of nodes is connected, the clustering coefficient in such networks is $C_{rand} = p$. It is a typical property of many real networks that they have a higher clustering coefficient than comparable random networks.

3.2.3 Scale-Free Networks

While studying the topology of the World Wide Web, Jeong et al. (1999) discovered a phenomenon that could not be explained by the prevalent network models at the time. This phenomenon was that there existed a relatively small number of nodes, with an extremely high number of connections to other nodes. Such nodes, called *hubs*, are practically forbidden by the degree distributions in both the Erdős-Rényi random graphs and Watts-Strogatz small-world networks. This led to further study of the degree distribution, and the discovery of *power-laws* governing the degree distribution of the Web.

Such hubs were also found in many other networks, and it was soon discovered that a wide range of natural and artificial networks display such degree distributions.

For a degree distribution to follow a power-law means that $P(k) \sim k^{-\gamma}$ for some constant γ specific to the network. $P(k)$ is here the probability that a random node

in the network will have degree k. Power-laws are scale invariant, which means that scaling the argument x of a power-law function $f(x) = ax^k$ by a constant factor c only causes a proportionate scaling of the function itself.

$$f(cx) = a(cx)^k = c^k f(x) \propto f(x) \tag{3.4}$$

This is why such networks are called *scale-free*. For a more detailed explanation of power-laws, see Appendix A.

The long tail in the degree distribution of scale free networks are what makes possible the occurrence of hubs in the network. Described by the power-law, almost all the nodes in the network have a very low degree, at most connected to a few other nodes. Hubs belong to a special group of nodes defying this trend and connecting to a high number of other nodes. These connectors hold the network together and are very important to the topology of scale-free networks.

It turns out that most hubs usually are connected to hubs with a lower degree, and that these in turn are connected to even smaller hubs. This *hierarchical hub-structure* gives scale-free networks one of their most important properties, a high fault tolerance. Since there is a very low number of hubs compared to the number of nodes, the removal of any random node is very unlikely to have almost any effect on the overall connectedness of the network. This is also a weakness, unfortunately, as the targeted removal of hubs will have a huge impact on the network.

In order to explain how scale-free networks emerge, Barabási and Albert (1999) proposed a theory for such networks not unlike to that of the previous models. The model is based upon the idea of developing, or growing, the networks from an initially small set of nodes. What they realized was that the edges were not, in fact, random, as supposed by all previous models. Instead, there was a trend that those nodes with already a high number of connections would get even more edges; a *rich get richer* phenomenon. The nodes showed a clear *preferential attachment* when adding new edges.

These two ingredients, growth and preferential attachment, forms the basis of what is known as the *Barabási-Albert model*. The model starts out with a small number of nodes, and then iteratively add new nodes with a few edges to those already present. The nodes to which a new node connects are selected with a probability dependent on the degree of the node, $\Pi(k_i)$, defined as

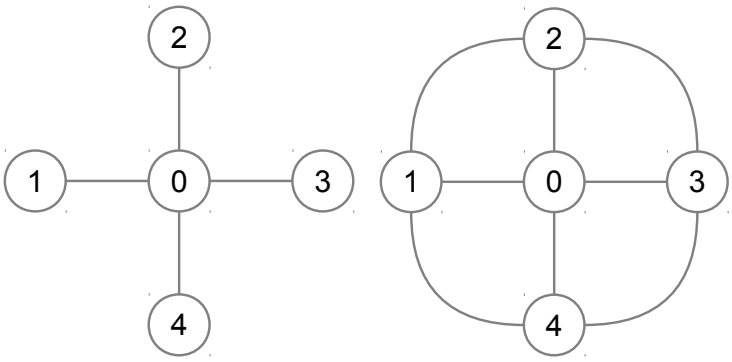
$$\Pi(k_i) = \frac{k_i}{\sum_j^N k_j}$$

where k_i is the degree of node i.

For a further discussion of this topic, Albert and Barabási (2002, Section VII) provides a detailed description of scale-free networks.

3.3 Centrality Measures

Node centrality is the measure of how important a node is within a network. Centrality can be used for node ranking, where the goal is to capture the relative importance of the nodes. The *rank* of a node is the node's position in a list of nodes ordered



(a) Star graph (b) Wheel graph

Figure 3.3: Two small graphs illustrating the concept of node centrality. Node 0 is the hub of both graphs, and clearly the most central node. This observation is the basis of the arguments for node centrality measures presented by Freeman (1978).

by importance. What it means to be important is not generally agreed upon, and as we shall see, there are many ways in which to measure a node's centrality.

Freeman (1978) reviewed the intuitive bases of the then existing measures for centrality. As he observed, the node forming the center of a star graph or the hub of a wheel graph (see Figure 3.3) is obviously more central than the other nodes. He showed that this node has the maximum possible *degree* in the network. It also falls on the shortest path *between* the largest possible number of other nodes, and, since it has the minimal possible distance to the other nodes, is maximally *close* to them as well. Each of these three properties led to the definition of a measure of node centrality. The three notions, *degree centrality*, *betweenness centrality* and *closeness centrality* still form the basis of most centrality measures.

Degree centrality is a totally local measure, concerned only with the number of neighbors a node has. Betweenness and closeness, conversely, take into account the complete graph, and how central nodes are in light of their paths to other nodes. A fourth measure that in some sense occupies the middle ground between the local degree-based centrality and the global path-based centralities has later been proposed by Bonacich (1972). He suggested that the eigenvector of the largest adjacency matrix could serve as a centrality measure. This is based on the idea that the importance of nodes are recursively related to the centralities of the nodes to which they connect.

One application of node centrality is by search engines on the Web to identify how important web pages are. Two of the most popular ranking algorithms for this are Google's PageRank and Kleinberg's HITS. These are variations on the eigenvector centrality, and are based on the idea of "recommendation" or "voting" among nodes (Mihalcea and Tarau, 2005). An edge from one node to another is seen as a vote for that node. The higher the number of votes, the higher the ranking will be. Each vote is not equal — votes from other important nodes are seen as more important than votes from unimportant nodes.

A weakness of the centrality measures presented by Freeman (1978) is that they are designed for binary connections; neither of them takes into account the fact that edges may be weighted. The degree centrality may easily be generalized to

weighted graphs by summing the connection strengths instead of the number of connections. Newman (2001) discusses the generalization of the betweenness and closeness measures to weighted graphs in the context of scientific collaboration networks. Opsahl et al. (2010) suggest that these generalizations should not be completely focused on weights, and that a middle ground must be found. They suggest a tuning parameter α by which the relative importance of number of connections and the connection strengths may be determined.

The following sections describe various centrality measures in more detail. Degree based centralities are first described, followed by the path-based betweenness and closeness centralities, and variations on these. Finally, eigenvector-based measures are presented, including PageRank and HITS. Figure 3.4 groups the various centrality measures according to these distinctions.

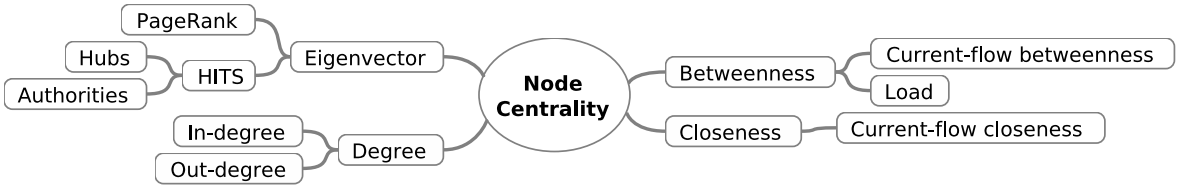


Figure 3.4: Mind-map of various node centrality measures. Closeness and betweenness, on the right, are path based measures. Degree, with its weighted and direction sensitive variants are based on nodes' direct neighbourhoods. The eigenvector-based measures are also based on neighbors, but take the importance of each neighbour into account. The measures are described in detail in Sections 3.3.1 through 3.3.9.

3.3.1 Degree Centrality

Degree centrality is the simplest and most intuitive centrality measure presented by Freeman (1978) — all that matters is how well connected each node is. In its most basic form, for binary connections, a node's degree centrality is simply the number of edges it has to others in the network, i.e. its degree. It is defined as

$$C_d(i) = \sum_j^N a_{i,j} \tag{3.5}$$

where $a_{i,j}$ is 1 if node i and j are adjacent, and 0 if they are not. The measure is naturally extended to weighted networks by letting $a_{i,j}$ represent the weight between the nodes.

This measure is affected by the size of the network over which it is calculated. To prevent this, it is possible to normalize by the maximum possible degree of the network.

$$C_d(i) = \frac{1}{N-1} \sum_j^N a_{i,j} \tag{3.6}$$

where N is the number of nodes in the network.

The rationale for this measure is that the nodes with most connections, or in the weighted case, strongest ties to other nodes, have many channels for communication with the rest of the network, and are thus influential. As observed in the context

of scale-free networks in Section 3.2.3, those nodes with a very high number of connections, the hubs, are responsible for holding the network together and ensuring robustness. Conversely, if nodes with high degree are removed, the network soon collapses.

Opsahl et al. (2010) argues that while the binary degree measure only values number of connections, the weighted version only cares about the sum of connection strengths. As a more balanced measure, they suggest

$$C_d^{\alpha}(i) = k_i^{1-\alpha} \times s_i^{\alpha} \tag{3.7}$$

where k_i is the degree of node i, s_i the total weight strength, and α a parameter determining the relative importance of the two. With α values of 0.0 and 1.0, the measure behaves as the purely unweighted and weighted degree centralities, respectively.

3.3.2 Betweenness Centrality

The second of the measures presented by Freeman (1978), betweenness centrality, is also based on ideas about communication networks. Nodes with high betweenness centrality are those that fall on the shortest path between a large number of other nodes. Because of their location, such nodes are in a position to control or influence the communication between others in the network.

The betweenness centrality of a node i can be expressed as

$$C_b(i) = \sum_{s \neq i} \sum_{t \neq i} \frac{\sigma_{s,t}(i)}{\sigma_{s,t}} \tag{3.8}$$

where s and t are nodes in the graph, $\sigma_{s,t}$ is the number of geodesic paths linking the two, and $\sigma_{s,t}(i)$ is the fraction of these paths that go through node i.

This value is affected by the size of the graph. To remove this effect, the centrality values can be normalized by $(N-1)(N-2)$, where N is the number of nodes in the graph. This number is the maximum possible pairs of nodes in the graph not including i.

$$C_b(i) = \frac{1}{(N-1)(N-2)} \sum_{s \neq i} \sum_{t \neq i} \frac{\sigma_{s,t}(i)}{\sigma_{s,t}} \tag{3.9}$$

Since Freeman operates on binary networks, his shortest paths are in practice geodesic paths between the node pairs. Newman (2001) suggests that Dijkstra's algorithm for shortest paths (Dijkstra, 1959) should be used instead. This way the weights of the network's edges can be taken into account, and the shortest path will not necessarily be the one visiting the least number of nodes.

Unconnected graphs pose a possible problem to this formulation of betweenness centrality, since some node pairs have no shortest path. This problem can be circumvented by applying the measure to each component separately.

3.3.3 Current-Flow Betweenness

Betweenness centrality is a measure of the control a node has over the spread of information through the network. From the description above, it is clear that this

model assumes that all communication pass along the shortest paths. This is of course a gross simplification in many situations.

Current-flow betweenness centrality is a measure that relaxes this assumption (Newman, 2005). With this measure contributions from all paths between nodes, not just the shortest, are counted. The shortest paths are still most influential, but all paths contribute to the overall betweenness.

The name, current-flow, is derived from the model used for flow of information through the network: that of current-flows in electrical networks. Newman (2005) presents the electrical network model, defining the current-flow betweenness centrality to be the current flowing through a node when one unit of current is injected into and extracted from the network, averaged over all source-terminal pairs. He proves that this is mathematically equivalent to the betweenness on random walks between nodes. Because of this, current-flow betweenness is also known as random-walk betweenness.

The betweenness centrality based on shortest paths, as described by Freeman, and this measure can be seen as extremes in a spectrum of possibilities. The former assumes that information flowing between nodes knows exactly where the optimal path is, and that it chooses to follow it. In the latter it is assumed that information have no way of knowing where optimal paths are, and instead flows around at random.

Calculation of current-flow betweenness is described by Newman (2005, p.9) as follows.

1. *Construct the matrix $D - A$, where D is the diagonal matrix of vertex degrees and A is the adjacency matrix.*
2. *Remove any single row, and the corresponding column. For example, one could remove the last row and column.*
3. *Invert the resulting matrix and then add back a new row and column consisting of all zeros in the position from which the row and column were previously removed (e.g. the last row and column). Call the resulting matrix T, with elements T_{ij}.*
4. *Calculate the betweenness using Equation* (3.10).

The betweenness values are calculated as the average of the current flowing through node i over all the source-target pairs s, t:

$$C_{cfb}(i) = \frac{\sum_{s<t} I_i^{st}}{\frac{1}{2}N(N-1)} \tag{3.10}$$

where I_i^{st} is calculated as

$$I_i^{st} = \frac{1}{2}\sum_j A_{ij}|T_{is} - T_{it} - T_{js} + T_{jt}|, \quad \text{for } i \neq s, t \tag{3.11}$$

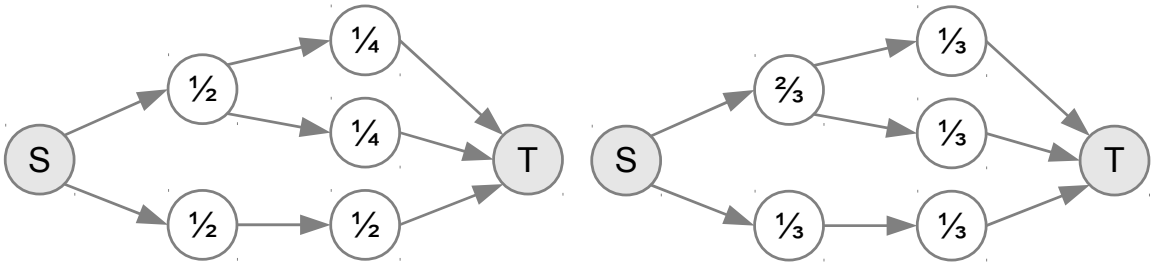
Like in the case of the (shortest-path) betweenness centrality, this procedure should be repeated separately for each component in unconnected graphs.

3.3.4 Load Centrality

Another centrality measure closely related to the betweenness centrality is load centrality. Load centrality was first introduced by Goh et al. (2001), who described it as equal to betweenness centrality. As Brandes (2008) points out, this is not strictly correct. He discuss it in the context of different variants of betweenness centrality and clarifies the differences between the two.

The difference lies in how multiple shortest paths are handled by the measure. As discussed above, the betweenness centrality is defined as the fraction of shortest paths between pairs of other nodes. This is also the case for load centrality, but the two measures differ in the way they handle the situation of multiple shortest paths between a pair of nodes.

With betweenness, the contribution to a node in the face of several shortest paths is simply the fraction of these paths the node is part of. A slightly different approach is taken in load centrality, which is defined from the view of a data packet sent between nodes following the shortest path. For each branch where it is possible for the packet to select among several shortest paths, the contribution is split evenly. Figure 3.5 illustrates the point in an example network.



(a) Load centrality (b) Betweenness centrality

Figure 3.5: Differences between load and betweenness centralities. Example network illustrates the difference in contribution from a single source-terminal pair with three shortest-paths. For load centrality (a), the contribution is divided evenly for each branch. The standard betweenness centrality (b) gives equal contribution to each of the paths, with more contribution for early nodes part of several paths. The example is adapted from Brandes (2008).

3.3.5 Closeness Centrality

Like betweenness, closeness centrality is path-based. It is concerned with the distance from node i to all other nodes in the network. A node with short distances to most of the network, it is thought, is in a position with good access to information. Nodes close to others are also in a good position to influence them. Messages originating in such a position should spread throughout the network in minimum time.

Newman (2001, p.4) formulates the motivation for closeness centrality, in the context of scientific collaboration networks, as follows:

> *Like betweenness it is a measure, in some sense, of the centrality of a vertex — authors with low values of this average will, it is assumed, be*

the first to learn new information, and information originating with them will reach others quicker than information originating with other sources.

In its most straight forward incarnation, the closeness centrality can be defined as

$$C_c(i) = \frac{1}{\sum_{j=1}^{N} d(i,j)} \tag{3.12}$$

where $d(i,j)$ denotes the number of geodesic paths linking nodes i and j.

Like with the other measures, it is important to normalize the values, so that they do not depend on the size of the graph. Normalizing by the number of nodes in the network gives the following formula.

$$C_c(i) = \frac{N-1}{\sum_{j=1}^{N} d(i,j)} \tag{3.13}$$

A problem with this formulation is, however, that it is only meaningful for connected graphs. In an unconnected graph, every point has at least one other point which cannot be reached from it, and thus

$$\sum_{j=1}^{N} d(i,j) = \infty \tag{3.14}$$

for all i. A way to handle this for unconnected graphs is to treat each component separately. This require an adaption of the normalization, in order to reflect node closeness in the overall graph from the locally calculated values.

$$C_c(i) = \frac{M-1}{N-1} \frac{1}{\sum_{j=1}^{M} d(i,j)} \tag{3.15}$$

where M is the number of nodes in the connected part of the graph containing node i, and N is the number of nodes in the entire graph.

A version of this measure can also be derived for graphs with weighed edges. As suggested by Newman (2001), Dijkstra's algorithm for shortest paths (Dijkstra, 1959) can be used to find $d(i,j)$ instead of the geodesic paths assumed by Freeman (1978).

3.3.6 Current Flow Closeness

Like current flow betweenness is a relaxation of the betweenness centrality to include not only shortest, but all paths between nodes, so is the current flow closeness centrality a relaxation of the standard closeness centrality.

Brandes and Fleischer (2005) generalize the closeness measure in much the same way as Newman (2001) generalized betweenness. Their derivation of current flow betweenness is, just like the derivation of current flow betweenness by Newman, based on a model of electrical networks. The electric network is defined as $Net = (G; c)$, where the graph $G = (E, V)$ is the usual tuple of sets of vertices and edges, and c is a list of positive edge weights. Edge weights in c indicate the strength, or conductance, of edge $e \in E$. Each edge is also given an arbitrary direction $\vec{e} \in \vec{E}$

in order to later describe the direction of current flowing through it. The actual orientation of $\vec{e}$ is of no importance for the end result, however.

To denote the supply of external current to the network a vector b indicates whether nodes are a source s or sink t of current.

$$b_{st}(v) = \begin{cases} 1 & v = s \\ -1 & v = t \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

Given a supply b_{st}, a current x_{st} is created through the network from s to t. A value of $x_{st}(\vec{e}) > 0$ indicates that the current follows the orientation of $\vec{e}$, a negative value means the current goes in the opposite direction. For each such current Ohm's law defines a potential difference, or voltage, given as $\hat{p}_{st}(\vec{e}) = x_{st}(\vec{e})/c(e)$. The absolute potentials for nodes $v \in V$ is given by vector p if $\hat{p} = p(v) - p(w)$ for all $(v, w) \in \vec{E}$. The absolute potential given st-supply b_{st} is denoted by p_{st}.

The current flow closeness is formally defined as

$$C_{cfc} = \frac{N-1}{\sum_{t \neq s} p_{st}(s) - p_{st}(t)} \tag{3.17}$$

where $p_{st}(i)$ denotes the potential difference, or voltage, of node i.

The term $p_{st}(s) - p_{st}(t)$ corresponds to the effective electrical resistance of the network when current is supplied at node s and drained from node t. This effective resistance can be interpreted as a measure of the distance between the nodes. The value of $N - 1$ in the numerator is a normalizing constant meant to remove the sensitivity to graph size.

Like the regular closeness centrality, described in the previous section, current flow closeness can be applied to both weighted and unweighted networks. To apply the measure to unweighted networks, the edge weight vector c can simply be set to 1 for all edges.

3.3.7 Eigenvector Centrality

Eigenvector centrality, as first presented by Bonacich (1972), is based on the idea that nodes are more central if they are connected to other nodes that are themselves central. Eigenvector centrality, thus, relies not only on the number of connections to other nodes, like degree centrality, but also on how well the neighbors are connected.

Bonacich defined the eigenvector centrality $V_E(v_i)$ of node v_i as a multiple of the sum of adjacent centralities.

$$\lambda V_E(v_i) = \sum_{j=1}^{N} a_{ij} C_E(v_j) \tag{3.18}$$

Expressed in matrix notation this becomes

$$\lambda \vec{x} = A\vec{x} \tag{3.19}$$

with A as the adjacency matrix, and $\vec{x} = (C_E(v_1), \ldots, C_E(v_n))$ a vector of centrality values. Equation (3.19) can be recognized as the eigenvector equation, which states

that x is an eigenvector of matrix A if there is a scalar λ such that the equation is fulfilled. Such a scalar is called an eigenvalue, and corresponds to the eigenvector. It is also required that the $\vec{x}$ is not the null-vector.

There are several algorithms for solving Equation (3.19) to find an eigenpair, an eigenvalue λ and the corresponding eigenvector x, for a matrix. One of these is the Power Iteration method. The λ found by this method is the one with the greatest absolute value. The eigenvector corresponding to this largest eigenvalue is called the principal eigenvector. Power Iteration is a simple method and has low storage requirements, which makes it suited for use with very large sparse matrices. A limitation of the method is, however, that it may converge slowly.

3.3.8 PageRank

PageRank is a version of the eigenvector centrality. It was first introduced by Brin and Page (1998), and is described in more detail by Page et al. (1998). The method is named after Larry Page, one of the founders of Google Inc. The algorithm has been made famous by Google's use of it to analyze the importance of pages on the World Wide Web.

The idea behind PageRank is that links from one page to another can be seen as votes among pages. A page links to another because it considers it a good page, in some respect, and pages with many incoming links should be considered authorities and be highly rated. The voting power of a link from a page is normalized by the total number of outgoing links, so that all pages with the same PageRank contributes with the same total amount of voting. Votes from pages that have higher PageRanks are seen as more important, and weighted higher than those with lower rank.

The intuitive justification of the algorithm is the so-called random surfer model. This model is described by Brin and Page (1998, p.110) as follows.

> *We assume there is a "random surfer" who is given a Web page at random and keeps clicking on links, never hitting "back" but eventually gets bored and starts on another random page. The probability that the random surfer visits a page is its PageRank.*

Viewed in this light, the PageRank is related to the random-walk model serving as justification of current flow betweenness, presented in Section 3.3.3. Where the random-walk model looks at the probability that a node will be encountered by random walks between all pairs of two nodes in the network, the random surfer model predicts probability that the node is visited by a random walks from random nodes.

Mathematically, the PageRank is described as

$$PR(v_i) = \frac{(1-d)}{N} + d \cdot \sum_{v_j \in \text{In}(v_i)} \frac{PR(v_j)}{|\text{Out}(v_j)|} \tag{3.20}$$

where $\text{In}(v_i)$ is the set of nodes with edges pointing to node v_i, and $\text{Out}(v_i)$ the nodes pointed to from v_i. The value of N is the number of nodes in the network, and d is a dampening parameter between 0 and 1, usually set to 0.85 (Brin and Page, 1998). Under the random surfer model, the value of d represents the probability that

a user clicks on one of the links when visiting a page. Thus, with probability $1 - d$, the random user gets bored and start with a random page. An illustrating example on a small network is given in Figure 3.6.

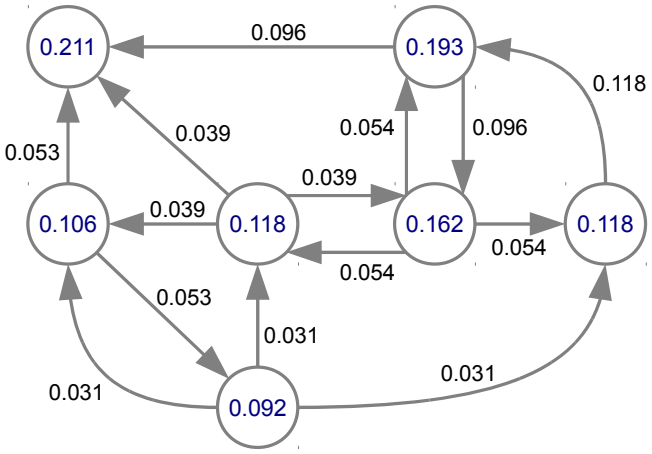


Figure 3.6: PageRank illustrated in a small graph. The values inside nodes indicate the centrality of the node. Values on the edges corresponds to the $\frac{PR(v_j)}{|Out(v_j)|}$ term of Equation (3.20). A value of 0.85 was used for d.

PageRank, as presented by Brin and Page (1998), was meant for use on the World Wide Web. In this context it is unusual for pages to link partially or multiple times to other pages, and thus the algorithm was designed for unweighted graphs. Mihalcea (2004) introduce the following formulation, extending the method to weighted graphs.

$$PR^W(v_i) = \frac{(1-d)}{N} + d \cdot \sum_{v_j \in \text{In}(v_i)} w_{ij} \frac{PR^W(v_j)}{\sum_{v_k \in \text{In}(v_j)} w_{kj}} \tag{3.21}$$

w_{ij} represents the weight of the edge from i to j.

The PageRank scores are calculated by starting from arbitrary values for each node in the graph, and iteratively compute new values using Equation (3.20) or (3.21) until convergence. The initial values do not affect the final values, only the number of iterations needed for convergence.

Page et al. (1998) also show that the PageRank can be calculated by finding the principal eigenvector of a modified adjacency matrix. That is, the eigenvector R, containing PageRank values, is the solution of

$$R = D + dA'R \tag{3.22}$$

where D is a vector where each value is $(1-d)/N$, and A' is the adjacency matrix adjusted so that all columns are normalized and sum to 1.

A final remark is that while the PageRank algorithm is designed for directed graphs, it can readily be applied to undirected graphs by replacing all edges between nodes by two edges: one in each direction.

3.3.9 Hyperlink-Induced Topic Search

Another iterative algorithm related to eigenvector centrality is the Hyperlink-Induced Topic Search (HITS) algorithm presented by Kleinberg (1999). He focus on the need

to identify the most "definitive" or "authoritative" pages on topics on the Web. He also notices that the link structure can be used to identify these pages. Based on his proposed link-based model, where some pages are authorities on topics and others link to many related authorities, he propose the HITS algorithm intended to exploit this structure.

For each node, the algorithm defines two scores, an authority score and a hub score. Authorities are pages with large number of incoming links. The authority of a page describes how much knowledge or information, it holds on a topic. Hubs are pages with many outgoing links. The hubness of a page is an indicator of how well it "knows" where to find information on a given topic. The best hubs point to the best authorities, and the best authorities are those that are linked to from the best hubs. Nodes can be both hubs and authorities at the same time. With $\text{In}(v_i)$ denoting the set of nodes with links to v_i, and $\text{Out}(v_i)$ the set of nodes v_i links to, the HITS scores are defined as

$$HITS_A(v_i) = \sum_{v_j \in \text{In}(v_i)} HITS_H(v_j) \tag{3.23}$$

$$HITS_H(v_i) = \sum_{v_j \in \text{Out}(v_i)} HITS_A(v_j) \tag{3.24}$$

An example of how HITS ranks nodes in a small graph is demonstrated in Figure 3.7. We see that the node in the upper left corner is the largest authority, and is linked to by several considerable hubs. It also has a hub score of zero, as it does not point to any other nodes. The leftmost of the two central nodes is the largest hub, linking to among others the largest authority node.

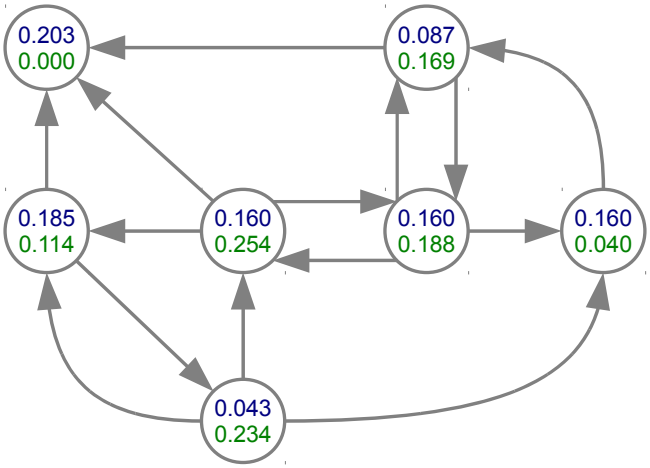


Figure 3.7: HITS illustrated in a small graph. The blue and green values inside nodes indicate their authority and hubness, respectively. Hubness and authority values are normalized so that they each sum to 1.0.

Like PageRank, the HITS algorithm is intended for use on the World Wide Web and does not take edge weights into account. A version for weighted graphs was also here proposed by Mihalcea (2004):

$$HITS_A^W(v_i) = \sum_{v_j \in \text{In}(v_i)} w_{ji} HITS_H^W(v_j) \tag{3.25}$$

$$HITS_H^W(v_i) = \sum_{v_j \in \text{Out}(v_i)} w_{ij} HITS_A^W(v_j) \tag{3.26}$$

Like PageRank, HITS is also designed for directed edges. It can, however, be applied to undirected graphs by converting them into directed ones by replacing all edges with directed versions pointing in both orientations.

Kleinberg (1999) describes how the authority scores and hub scores for the network is related to the eigenvector centrality. He shows that, with A as the adjacency matrix of the graph, the authority scores for the nodes can be found in the principal eigenvector of $A^T A$, and hub scores in the principal eigenvector of AA^T.

The main difference between PageRank and the HITS algorithm is that PageRank is based on a model in which authority is passed directly from authority to authority, without interposing the notion of hub pages.

3.4 Text as Graph

Many systems have structures that allow them to be modelled as graphs and possess properties from complex networks as described above. This also applies to the system of natural language. In this section, we will describe some of the research that has been done on the network properties of written language.

There are several ways in which networks can be constructed based on text. Solé et al. (2005) discuss and compare various ways in which to do this. They define three types of text networks: *co-occurrence networks*, *syntactic networks*, and *semantic networks*. Co-occurrence networks are networks in which words that appear close to each other in text are linked together. In semantic networks, edges represent semantic relationships between words, usually built from constituency structures from the text. The last type, semantic networks, captures semantic relations between concepts represented by words in the text.

Table 3.1 on the facing page, adapted from (Solé et al., 2005), lists some of the properties commonly found in the network types. In the following sections, we describe and give examples of research done on these network types. The network types do indeed display the expected properties, such as small-world and scale-free topologies.

3.4.1 Co-occurrence Networks

Ferrer i Cancho and Solé (2001) were the first to note that co-occurrence networks built from human language are both small-world networks and display scale-free degree distributions. They constructed a complex network from documents in the British National Corpus[1]. The network used words as nodes, and linked words together if they occurred next to each other, or with a single word separating them.

[1]The BNC is a 100 million word collection designed to represent a wide cross-section of current British English. It is available from `http://www.natcorp.ox.ac.uk/`.

Table 3.1: Typical values for some properties found in different types of language networks. Adapted from (Solé et al., 2005). All types display scale-free degree distributions, thus the γ-values represent power-law exponents.

	Co-occurrence networks	**Syntactic networks**	**Semantic networks**
Order, N	$10^3 - 10^6$	$10^3 - 10^4$	$10^4 - 10^5$
Average degree, $\langle k \rangle$	$4 - 8$	$5 - 10$	$2 - 4$
Characteristic path length, l	$3 - 4$	~ 3.5	$3 - 7$
Clustering, C/C_{rand}	$\sim 10^3$	$\sim 10^3$	$\sim 10^2$
Degree distribution, γ	$2.2 - 2.4$	~ 2.2	~ 3

They found that this kind of word co-occurrence network had a short characteristic path length $l = 2.63$, and a high clustering coefficient $C = 0.437$. Since the corresponding values in a random network would be $l_{\text{rand}} = 3.03$ and $C_{\text{rand}} = 1.55 \times 10^{-4}$, this is clearly a small-world network. The degree distribution of the network followed a two-regime power law, with degrees less than about 1000 decaying with an exponent $\gamma = 1.5$, while higher degrees had the higher $\gamma \simeq 2.7$.

Dorogovtsev and Mendes (2001) have looked more closely on the two-regime power law found by Ferrer i Cancho and Solé, and propose a theory of the evolution of language. They treat language as a self-organizing network of interacting words, and show that the two regimes in the distribution naturally emerge from the dynamics of this network. The network grows by preferential attachment as in the Barabási-Albert model, with the addition that at each increment of time new edges also emerge between words already present in the network. Based on this theory, they predict that the size of the kernel lexicon, the core part of language, does not change as the language evolve.

This prediction is also supported by Choudhury and Chatterjee (2010) in a recent study, where they expand upon the theory by Dorogovtsev and Mendes. Their conclusions are based on examinations of the two-regime power-laws for co-occurrence networks in several languages from three different language families. An interesting consequence of this is that it becomes only marginally harder for new speakers to learn a language as words are added to it, since they only have to learn the kernel lexicon.

3.4.2 Dependency Networks

Dependency networks are a type of *syntactic text network* in which edges represent syntactic dependencies between words within the same sentence. The dependency grammar formalism assumes that basic syntactic structure consist of lexical nodes, e.g. words, and binary dependency relations linking these together. The formalism thus defines a network structure where dependency relations connects pairs of words. The links can be defined with direction from the *head word* to its *modifier*, or vice versa. By considering the dependency network representing a sentence as a subgraph of a larger network structure, dependency networks can be formed for larger corpora.

Ferrer i Cancho et al. (2004) have studied dependency networks in several different languages. They constructed three syntactic dependency networks from large corpora in Czech, German and Romanian, and found that they display properties not unlike those that had been reported for other linguistic network types. The networks were small worlds, with clustering coefficients C of 0.1, 0.02 and 0.09, for Czech, German and Romanian, respectively. This is much higher than the corresponding C_{rand} of 4×10^{-4}, 6×10^{-6} and 9.2×10^{-4} of a random network. The characteristic path lengths l range from 3.4 to 3.8, which is roughly the same as their equivalent l_{rand}. They also found that the networks were scale-free, with $\gamma \sim 2.2$, when the edges were regarded as undirected.

In addition to these two properties, the found an interesting relationship between a word's frequency and its degree in the network. By comparing the average value of word frequency f against node degree k, they found an approximately linear relationship between the two. Taking into account Zipf's law, this must mean that function words (i.e., prepositions, articles, determiners, etc) must be the be the most connected in the network, a prediction they confirmed by observation.

Another interesting observation that has been done on dependency networks regards the spacial distribution of the semantic relations. The euclidean distance between two words in a sentence is defined as the number of words separating the two, plus one. $\langle d \rangle$ is defined as the average distance between pairs of words in a sentence that are connected by a dependency relation. Ferrer i Cancho (2004) found that $\langle d \rangle$ is small, and grows but very slowly with the sentence length. He estimate that as much as $50 - 67\%$ of dependency links are formed between words at distance 1, i.e. by successive words, and $16 - 25\%$ are formed at distance 2. This means that distances of 2 or less may contribute up to 92% of the syntactic relations.

3.4.3 Synonym Networks

A synonym network is an example of a *semantic network*. These are networks in which words are linked together only if they constitute a pair of synonyms. This kind of networks are also often called *thesaurus networks*.

Albert and Barabási (2002) report, among other things, an interesting study of such a network based on the Merriam-Webster Dictionary. In this network a giant component was detected, consisting of 22311 of the 23279 words that had synonyms. The average shortest path length in the network was only $l = 4.5$, and the clustering coefficient of $C = 0.7$ was very high compared to $C_{\text{rand}} = 6 \times 10^{-4}$ for a corresponding random network. The synonym network was thus, like those described above, a small-world network. Similarly, its degree distribution followed a power-law with $\gamma = 2.8$.

3.5 Summary

We started this chapter by introducing graphs and basic related concepts, before moving over to the more general field of networks. Networks are models of real systems, made out of relations between many entities, based on graphs. A network model typically contain a very large number of nodes, which makes it impractical

or impossible to look at the whole graph in detail. Instead, network properties such as clustering, degree distributions or the small world property are studied, and the network can be assigned a network model based on its properties. We have described random graphs, small world networks and scale-free networks as examples of the most important such models.

The concept of graph centrality was subsequently discussed, and many measures for this defined. Graph centrality is used to define the relative importance of nodes in networks.

Finally, we saw that natural language have many properties that enables us to view it from the perspective of networks, and that small-world topologies and scale-free degree distributions are common among textual networks.

Chapter 4

Graph-based Text Representations

In Section 3.4 we discussed the idea of representing text as networks. This chapter focus on the practical aspects of graph-based text representations, and their use in TCBR. Some related research is first presented in Section 4.1. Section 4.2 then presents our approach, which is a continuation of some of our earlier work (Valle, 2010). Next the evaluation methods and datasets used to assess the representations are presented in Section 4.3. A brief summary concludes the chapter.

4.1 Related Work

The major lines of our use of graphs to represent text do not represent a novel approach, but is based on and inspired by work done by many others. In particular, it is influenced by the TextRank (Mihalcea and Tarau, 2004) and LexRank (Erkan and Radev, 2004) systems, but also other systems and approaches have been influential.

TextRank and LexRank are described in the following section. Section 4.1.2 discusses some other related approaches, and in Section 4.1.3 the link with TCBR is explored. Some additional related research is mentioned in Section 4.1.4

4.1.1 TextRank and LexRank

Both TextRank and LexRank are systems that apply graph techniques to perform extractive text summarization, although the TextRank can be applied to other NLP tasks as well.

The two systems follow the same main steps. First they represent the text as a graph. Then, using graph centrality algorithms, they rank the different parts of the text. The parts of the text highest ranked is finally extracted as the summary. The TextRank process is described as follows by Mihalcea and Tarau (2004, page 406):

1. *Identify text units that best define the task at hand, and add them as vertices in the graph.*
2. *Identify relations that connect such text units, and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.*
3. *Iterate the graph-based ranking algorithm until convergence.*

4. *Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.*

As we see, TextRank leaves the choices of text units and relations open to be determined based on what best suites the task at hand. The task itself is also not specified, and the system can thus be applied to different problems, such as document summarization, keyword extraction or word-sense disambiguation (Mihalcea et al., 2004).

LexRank is created specifically for the purpose of document summarization, and defines the selection of text units and relations explicitly. Seen in this light, LexRank can be seen as an embodiment of the TextRank approach directed at this particular task. LexRank uses sentences as the basic text unit, and defines sentence similarity based on cosine of TF-IDF vectors as relations in the network.

Mihalcea and Tarau (2005) presents summarization and keyword extraction as example applications for TextRank. Sentences are used for summarization, while tokens are used as text units for keyword extraction. Similarity between sentences are defined in terms of content overlap, i.e. the number of tokens common to two sentences. Co-occurrences within textual contexts are used as relations between tokens.

As suggested by their names, both systems use PageRank as their ranking algorithm. LexRank experiments with the use of degree centrality, and TextRank has also been evaluated with HITS and the Positional Power Function (Mihalcea, 2004; Mihalcea and Tarau, 2005). However, no extensive investigation into the choice of graph centrality algorithms has been done.

Since TextRank is a general method, the specific graph representation must be selected depending on the problem. There has been no attempts at describing which graph representations are suited for what problems.

As pointed out by Mihalcea and Tarau (2005), the TextRank algorithm is language independent. They illustrate this by doing document summarization in both English and Portuguese. Similarly, Mengxiao et al. (2004) shows that keyword extraction can be done for Chinese. They have independently come up with a method similar to TextRank using co-occurrence networks.The main difference is that rather than PageRank they use a measure of how absence of nodes affect the characteristic path length of the network. This measure is in practice equivalent with the betweenness centrality measure (described in Section 3.3.2). Mengxiao et al. also show that networks created from Chinese documents, like English, have a small world structure, and discuss how this enable the keyword extraction process using the idea of node centrality.

4.1.2 Other Text Network Representations

Liu et al. (2008) have also studied text networks. Their proposed text network representation model basically does the same as the TextRank method, that is, use graph centrality on graph representations of text. They too use PageRank, and their model is applicable to different problem areas. They use words as the textual units in their networks, and discuss different ways to identify edges using co-occurrences, word dependencies and semantic relations.

Based on the PageRank scores of the words in the network, Liu et al. suggests to create a text representation vector. This vector can be used in problems such as document classifications, by defining document similarity as the cosine of representation vectors. Seen from the perspective of Information Retrieval, this is simply the case of using PageRank as a weighting scheme for the Vector Space Model.

Wang et al. (2005) presented a very similar approach, which they call *term graphs*. They also use graphs as an intermediate step, creating vectors of PageRank score values for each document. Their graphs are co-occurrence networks. The networks are built from the most frequently co-occurring terms, which are identified using association rule mining. Different from the model presented by Liu et al. are the similarity measures. Two similarity measures are presented: rank correlation and term distances. The rank correlation is measured by sorting the term-vectors for each document after relative rank, and then use a statistical correlation measure. To measure term distance, a term distance matrix T, with T_{ij} representing the smallest number of hops between terms i and j in the term graph, is created for each textual category. The similarity of a document D to each category is then calculated as

$$\frac{n}{\sum_{i,j \in D} (T_{ij})^{\alpha}}$$

where i and j represent pairs of terms in the document, and α is a parameter to adjust the effect of distance to the similarity score. In their experiments, Wang et al. use $\alpha = 2$. n is the total number of term-pairs in the document.

Another use of graphs is presented by Tomita et al. (2004a). Their main concern is the use of graph-based representations for knowledge discovery in Information Retrieval. They initially propose their graph model as a way to help users clarify their information need through interaction with *query graphs* (Tomita and Kikui, 2001). This idea is further developed to that of *subject graphs* (Tomita et al., 2004a,b), a text representation model intended to simplify the steps needed in knowledge discovery from large volumes of texts. The graphs are co-occurrence networks created from the text. Nodes are used to represent terms in the text, and co-occurrence frequencies of terms in context such as sentences, clauses or word windows are used to calculate edge weights. The model do not use node centrality to rank the terms in any way, but rather determine significance by term frequency (TF).

4.1.3 Graphs for TCBR

While the work described above mainly focuses on applications of graphs in IR, Cunningham et al. (2004) have investigated the use of graphs in TCBR. Their work expands upon a graph model presented by Schenker et al. (2003b,a). The model is based upon structural relationships between terms in documents, and graph matching algorithms are used to assess similarity between graphs. Edges in the graph represents adjacent words in the text, and are labelled with the section they occur in.

While Schenker et al. apply their model to tasks such as document clustering and classification, Cunningham et al. experiment with an extension of the algorithm in order to investigate its potential benefit to TCBR. The main difference in the extended graph model is that a set of domain-dependent terms, *signifiers*, influence the graph construction.

After evaluating graph models against a domain expert in a retrieval task, they conclude that graphs-based approaches to TCBR shows promise. One of the strengths of graphs is that they are ideally suited to capture domain-specific relationships that exist between features. Graph-based representations also promise possibilities of automated case generation from text, which would greatly reduce the required knowledge engineering effort.

They also point out a number of weaknesses and problems with their graph-based models. First and foremost, the graph distance measures fail to take into account the relative importance of features, which is key for good similarity measures in CBR. It is also a problem that the measures cannot address negations in the text.

4.1.4 Other Related Research

The intersection of graph theory and natural language processing is interesting, and a lot is being done in this area. The research presented above is only the part of the relevant work that has influenced our work most directly. Most of the presented approaches share the same basic premise: representing the text as a graph, and then ranking the nodes using one of the possible ranking or centrality algorithms. There are also many other ways to apply graphs to traditional text problems. For example, Dhillon (2001) uses biparite word-document graphs to do simultaneous clustering of words and documents. His method is based on spectral graph partitioning, which is a heuristic method for graph partitioning.

Another problem that has received attention is unsupervised word sense disambiguation (WSD). This is usually done by creating networks where both words from the text and their possible senses are represented by nodes. Edges are created using semantic relations between nodes found in a thesaurus. With such a network, the most appropriate senses can be found by ranking the nodes. Navigli and Lapata (2007) evaluate a wide range of graph connectivity measures for this purpose. Measures such as spreading activation, PageRank, HITS and P-Rank are used by Tsatsaronis et al. (2010) who also compare their results to other known techniques, including those discussed by Navigli and Lapata (2007).

4.2 Our Representations

Our approach is closely related to that of the LexRank and TextRank systems described in Section 4.1.1 above. We conceive the representation process of TextRank to be comprised of two main steps, as illustrated by Figure 4.1 on the next page. The first step is to represent the text as a network. In the second step, nodes of the network are evaluated using graph centrality measures, and their values are used to represent the document as a feature-vector.

Our approach is centered around the same two steps. Each step reflects one of the main decisions that must be made, namely how to construct a good graph representation from a textual document, and, given such a network, how to identify the important terms from the text. Although these questions have been answered by several of the systems described above, no thorough evaluation of different graph

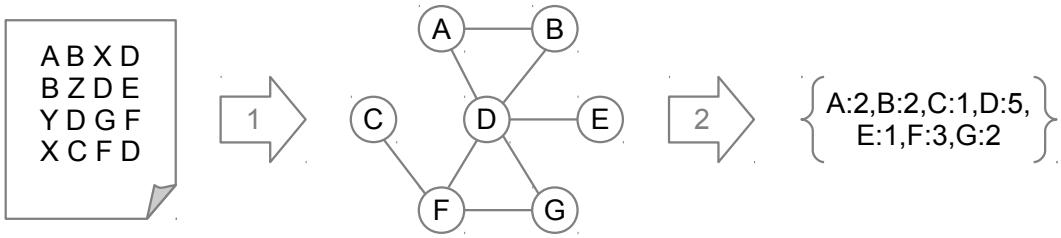


Figure 4.1: The basic steps of our representation process.

representations and centrality measures has been made. We seek to perform such an evaluation with specific focus on the task of measuring document similarity.

In the first step of the representation process, we go from a textual document to a text network representation. To do this, some preprocessing of the text is usually required, before a graph is constructed from the processed text. We shall, like some of the research presented previously in this chapter, use words as the unit of text represented by our nodes. What is needed next is to define what constitutes relations between words — the information to be retained in the edges. The choices of definitions for nodes and edges constitute the basis of a text network representation. In the following chapters we investigate two alternatives for such representations: co-occurrences networks and dependency networks.

In step two, a feature-value representation is constructed based on properties of the text network crated in the first step. This is done by evaluating the centrality of term-nodes in the text network, and building a feature-value vector with the vocabulary of terms as features and the corresponding term-centralities as values. PageRank have traditionally been a common choice for this task. We will evaluate PageRank, along with the other node centrality algorithms introduced in Section 3.3, to see which performs best.

A more detailed picture of the process is illustrated in Figure 4.2 on the following page.

Two tasks form the basis for the evaluations: *classification* and *case retrieval*. These are presented, along with the datasets used, in Section 4.3.

4.2.1 Step 1: Building Text Networks

The first thing to do is to create network representations of the documents. This corresponds to the first two steps in Figure 4.2 on the next page.

The networks require extraction of nodes and edges from the text. Many such representations are possible, with different choices of textual units for nodes, and relations between these as edges. We adopt individual terms as the basic unit of text for our nodes, and will focus on how to define edges from term-term relations in text. Some possibilities were described in Section 3.4. Three main types of text networks were identified: *co-occurrence networks*, *syntactic networks*, and *semantic networks*.

By far, the most common network type among the research presented in Section 4.1 above is the co-occurrence networks. These are networks in which edges are created between terms that occur close to each other. It is argued that co-occurrence networks retain a lot of information about the structure of the text. Co-occurrence

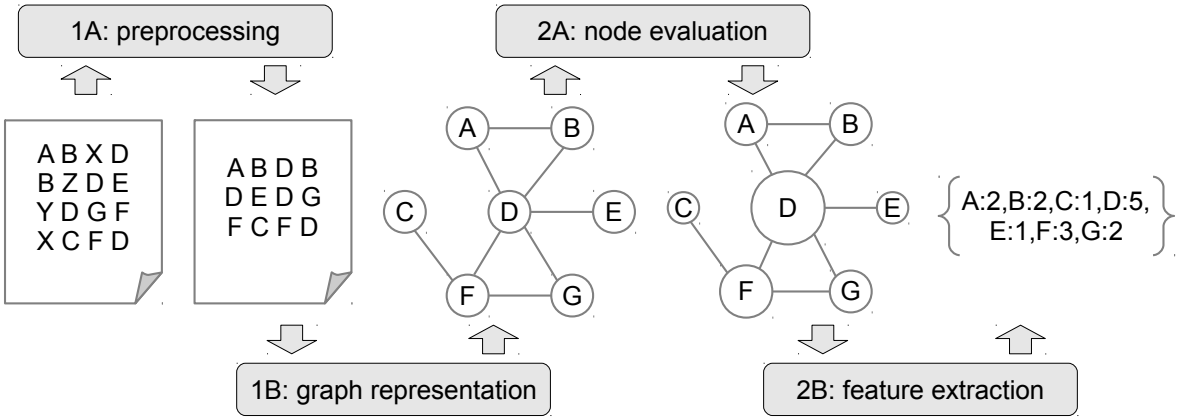


Figure 4.2: A more detailed outline of the representation process.

networks are also easy to implement and have given reasonably good results. We discuss this type of networks in Chapter 5.

As an example of the second type, syntactic networks, we described *dependency networks* in Section 3.4.2. In these networks edges are defined as special syntactic dependencies between words in sentences. We study this representation type in Chapter 6.

With each of the network representations, there are several aspects to investigate in addition to the significance of nodes and edges, such as whether edges should be directed or weighted, and how the text should be preprocessed. Through the following two chapters we evaluate these empirically in search of the best configurations for the representations. At the end of each chapter, we present the final version of the network representation.

4.2.2 Step 2: Creating Word Centrality Vectors

The evaluation of the network representations is not done directly, but is based on feature-value vectors constructed based on the networks. The conversion from network to vector representations is done in the last two steps of Figure 4.2. The items in these vectors capture the relative importance of each term in the documents. Thus, these document representation vectors can be seen as roughly equivalent to TF or TF-IDF vectors from the vector space model, presented in Section 2.2.4. The main difference is that here the estimates of term importance is not derived as a function of their frequency, but defined based on centrality as a nodes in the text networks. Once documents are represented as vectors in this way, similarity is measured by standard IR methods such as the cosine similarity measure.

An obvious way to decide the importance of a term in a given document is to calculate its node centrality in the document network. There are many centrality algorithms to choose from, and we presented the most important ones in Section 3.3. The most famous of these is perhaps PageRank, which is used in several of the systems we presented initially, including LexRank and TextRank. PageRank is based upon an idea of recommendations between nodes, and is undoubtedly a successful ranking algorithm for web pages. We have found no study, however, regarding whether it is particularly suited for text networks. For this reason, we evaluated

several centrality algorithms, including PageRank, as candidates for both network representation. Parts of this evaluation, with focus on co-occurrence networks, has been presented earlier (Valle, 2010). Here we build upon this work, and also present similar evaluations for dependency networks.

We will refer to the centrality of a term, as measured by a node centrality algorithm, as its *term centrality* (TC). This use of node centrality over a network created from a single document can be regarded as a local measure, equivalent to TF, in that it uses only information about the document itself.

Conversely, other measures take into consideration also information about the term within the entire corpus. The obvious example of this is the IDF part of TF-IDF. We will also examine such global measures, analogous to TF-IDF but based on network centrality rather than frequencies. The rationale behind TF-IDF is that terms that are frequent in one document, but which occur infrequently in the rest of the corpus, are considered as important. Likewise, we reason that terms that are much more central in one document than in the overall collection of documents should be considered more salient for this document.

We call this measure *term centrality-inverse corpus centrality* (TC-ICC), and define it as

$$\text{TC-ICC}_{t,d} = \frac{\text{TC}_{t,d}}{1 + \text{CC}_t} \tag{4.1}$$

where CC_t is the corpus centrality of term t, i.e. the centrality of t in a text network created from the entire corpus. Both TC and CC are normalized values in the range [0,1]. Thus, the 1 in the denominator is needed to avoid dividing by zero. The resulting TC-ICC values are also normalized to [0,1]. The highest scores are awarded to those terms that are both maximally central in the document ($T \sim 1$) and not at all central elsewhere in the corpus ($CC \sim 0$). All terms with low TC will get low TC-ICC values, while those with high values for both TC and CC are assigned scores at about 0.5.

4.3 Evaluation Methods

To test the various representation models, we need formal evaluation criteria able to determine their usefulness empirically. We have opted to base our evaluation on two separate tasks in order to get a broader assessment of the representations. One is a classification task, and the other a document retrieval task. Hence, we will be testing the suitability of the considered representations for measuring document similarity in two different ways. Similarity measurement is of key importance in the retrieval stage of CBR.

The measures are described throughout the following sections.

4.3.1 Classification Evaluation

Our first method of evaluation is document classification. Classification is the task of assigning a document to one or more categories, based on its contents. A k-Nearest Neighbours (k-NN) classifier will be used. The k-NN classifier is used because it bases the classification decision on similarity between documents. Thus, if a representation model represents similarly documents from the same category, it will

get a high classification performance. The similarity is measured using the cosine similarity measure (Section 2.2.4).

The evaluation process is outlined in Algorithm 4.1. The subroutines should be quite self explanatory: RETRIEVECASES retrieves the set of most similar cases from the training set; MAJORITYLABEL returns the most common label among the retrieved cases. This is the same evaluation method that was used in our previous study (Valle, 2010), and a more detailed discussion can be found there.

Algorithm 4.1 Classification evaluation

Input: set of *training* cases, set of *test* cases
Output: classification *accuracy*

```
1: correct ⇐ 0
2: for all query in test do
3:    retrieved ⇐ RETRIEVECASES(query, training)
4:    label ⇐ MAJORITYLABEL(retrieved)
5:    if correct label then
6:       correct ⇐ correct + 1
7:    end if
8: end for
9: accuracy ⇐ correct / SIZEOF(test)
10: return accuracy
```

Since this is a supervised evaluation method, it require that we have documents labelled with their category. For this purpose, we will use the TASA dataset, which is described in the next section.

The aim is, of course, to evaluate the quality of the measured case similarity. In some sense, we can view the document contents as problem descriptions, and the categories as their solutions. Given a query document from a particular category we want other documents from the same category to be retrieved. The performance of a representation model can be judged by the classification accuracy, i.e. the fraction of cases in the test set that are correctly classified.

To achieve good classification accuracies the documents found most similar need of course only be within the right category. With the TASA dataset, there is a 1 in 9 chance of getting the category correct by random selection. The evaluation also does not speak as to whether the best or worst cases in the category is retrieved. Thus, the classification evaluation ensures that the retrieved documents are within the right ballpark, so to speak, but not that the very best documents are found. This is where the second evaluation method, i.e. case retrieval, comes in. Case retrieval evaluation is presented shortly in Section 4.3.3.

4.3.2 The TASA900 Dataset

The TASA dataset is a corpus of documents containing text sampled from curriculum used in US high schools. It consist of 37 600 documents arranged into nine categories, totalling approximately 10 million tokens of text. The categories are: *Business*, *HomeEconomics*, *LanguageArts*, *Science*, *Unspecified*, *Health*, *IndustrialArts*, *Miscellaneous*, and *SocialStudies*. TASA stands for Touchstone Applied Science Associates,

whom we thank for providing us with the data. We use TASA for the classification evaluation method in the following two chapters, when evaluating various aspects of the representations.

TASA is a relatively diverse corpus with documents from a wide range of topics, both within and across categories. This means that it is a challenging dataset to classify, which is the reason we chose it for our evaluation. The classification challenge raises from the in-category variation. Within the category *SocialStudies*, for example, there are documents about topics as diverse as Japanese samurai warriors and class biases within public opinion polls. This, of course, makes it difficult to identify the category of a single document based on its contents. We believe that a challenging dataset is important in order to test the representations thoroughly.

Rather than using the complete corpus, we have created a subset to use in the evaluation, the *TASA900*. For this, we have used the first 100 documents from each category. The total number of words is 136227. Average length of the documents is 314, with a standard deviation of 32. The distribution of document lengths in the dataset is shown in the histogram in Figure 4.4(a), and the sentence length distribution in Figure 4.6(a).

The dataset has been split into training and testing sets containing 60 and 40 percent of the documents, respectively. The split has been done randomly, but in such a way that the categories are uniformly distributed in each set.

4.3.3 Case Retrieval Evaluation

This evaluation method is intended to be a finer evaluation of the retrieved document cases than the classification task. The point is to evaluate the quality of a retrieved case, regardless of its category.

Such an evaluation could be performed both in a supervised and an unsupervised way. Since we do not have available any dataset with this kind of information, and given the time and cost requirements connected with creating such a dataset, we have opted for an unsupervised approach. Our approach is heavily influenced by one of the tenets of CBR, namely that *similar problems have similar solutions*.

The case base consists of documents that have been split into two parts, *problem descriptions* and *solutions*. When a query case is presented, the case retrieval is performed solely on basis of the problem description part of the cases. The quality of the retrieval is subsequently determined based on the similarity between the cases' solutions. These two steps are illustrated in Figure 4.3.

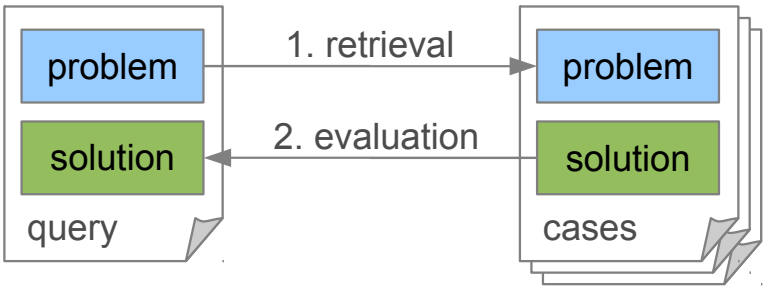


Figure 4.3: Illustration of the retrieval evaluation task.

Algorithm 4.2 outlines the process more formally. The RETRIEVECASES subroutine is the same as for the classification evaluation above, although this time only the problem description part of the case is included in the similarity judgement.

COMPARESOLUTIONS performs the evaluation of the retrieved case by comparing the solution part of the retrieved case to that of the query case. Evaluation of the case retrieval needs naturally be comparable between the various representations. For this reason, cosine similarity of TF-IDF vectors will be used. The TF-IDF vectors are created from preprocessed versions of the solution texts. The preprocessing consist of stop-word removal, case folding, stemming and removal of all terms with less than 3 characters.

The retrieval and subsequent solution evaluation is done in a leave-one-out fashion. That is, the case base consist of all the reports with problem descriptions and solutions, but when each is used as query case, it is excluded from the retrieved reports. The average match strength over all queries are used as the final performance measure.

Algorithm 4.2 Retrieval evaluation

Input: set of *training* cases, set of *test* cases
Output: solution match *strength*

```
1: similarities ⇐ {}
2: for all query in test do
3:    retrieved ⇐ RETRIEVECASES(query, training)
4:    sim ⇐ COMPARESOLUTIONS(retrieved, query)
5:    similarities ⇐ similarities ∪ {sim}
6: end for
7: strength ⇐ AVERAGE(similarities)
8: return strength
```

In theory, the evaluation measure have a possible range of 0.0 to 1.0, where the former would mean that no retrieved case solution has anything in common with that of their query, and the latter that all queries retrieve perfectly matching solutions. Only a limited part of this range is used much in practice, however. We see this in Figure 4.7 on page 47, which is a histogram of the similarity match between all pairs of solutions in the case base. This is because perfect scores are difficult to obtain since it would require the use of the exact same terms, and with the same frequencies, in the solutions. The wast majority of solutions are clearly very dissimilar. The tail of the distribution extends beyond that shown, almost all the way up to 1.0, but these are so infrequent as to be invisible in the figure. This means that the performance will be very sensitive to which reports that are retrieved, and it will be difficult for the representations to obtain high performance in this evaluation measure. Scores of ~ 0.20 are thus more realistic, and should be considered a reasonable performance.

4.3.4 The AIR Dataset

Our second dataset is based upon a set of incident reports, written by the Transportation Safety Board of Canada (TSB). The TSB is an independent agency created

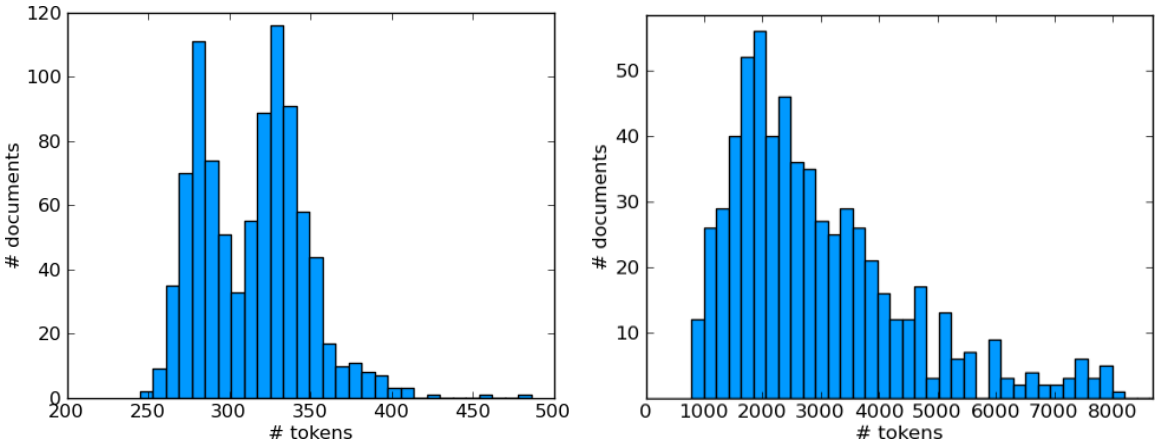
to advance transportation safety through the investigation of occurrences of incidents or accidents related to the marine, pipeline, rail and air modes of transportation. The reports are made available from the TSB website[1].

Our Aviation Incident Report (AIR) dataset consist of reports from the aviation category. The dataset cover a total of 628 reports, spanning the years 1990 through 2008. We use this dataset for the retrieval evaluation, and therefore split the reports into cases consisting of two parts: problem description and solution. This separation is done according to the structure of the reports, with some sections constituting each part.

We based the division into problem description and solution on the section titles of the reports. There were 54 distinct titles among the reports. We identified solution sections by their title, letting every section with a title containing the word *analysis*, *finding*, *causes*, *contributing factors*, *safety action*, or *conclusion* be part of the solution. Also subsections under a section with such a title were included in the solution. The rest of the sections in the reports were then treated as problem descriptions.

The documents in the AIR dataset are much longer than those in TASA. The TSB has in many cases studied the incident in depth, resulting in extensive reports. Average document length for the dataset is 2991 words, with a standard deviation of 1728. There are a total of 1878364 words in the dataset. Figure 4.4(b) show a histogram of the document length distribution. Left out of the histogram are a couple of outliers with lengths of 17159 and 17745.

Figure 4.5 on the following page shows the distribution of document lengths for the problem descriptions and solutions. We observe that while the problem descriptions and solutions have very similar length distributions, the latter are generally shorter.

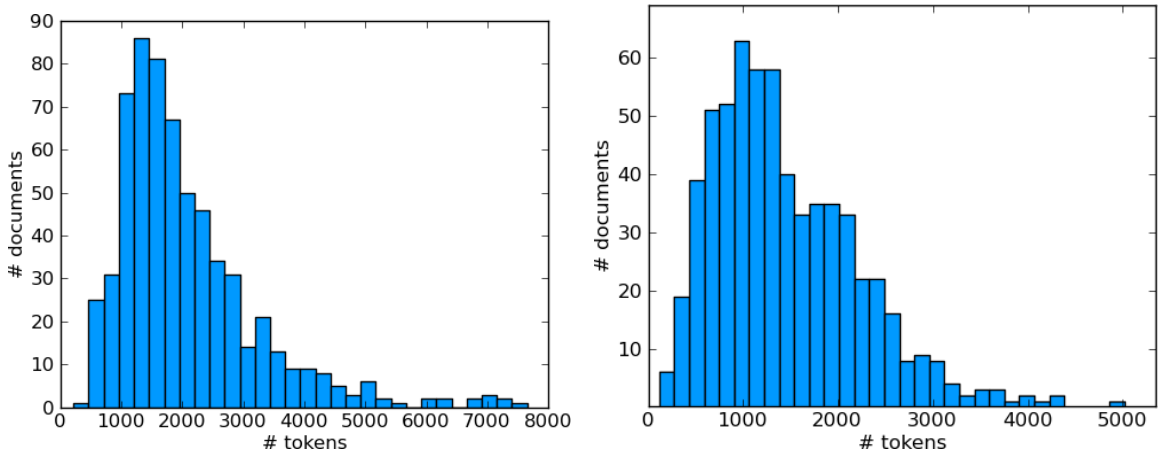


(a) TASA900 (b) AIR

Figure 4.4: Distribution document lengths in the TASA900 and AIR datasets.

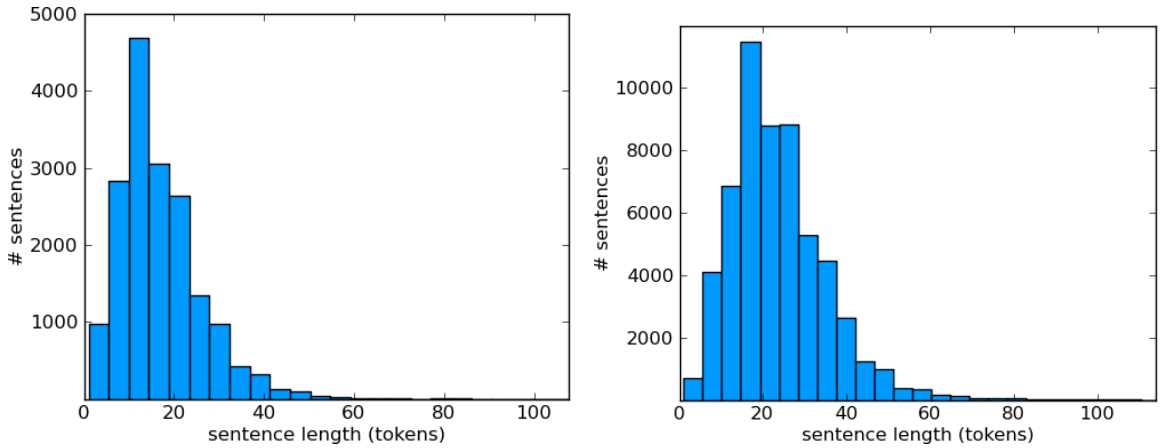
Not only the length of documents, but also the length of the sentences they are composed of, are important for the representations. Figure 4.6 plots the distributions of sentence lengths in the two datasets. We have plotted the length of sentences in the problem description part of the AIR dataset, since this is the part that is used for

[1] `http://www.tsb.gc.ca`



(a) Problem Descriptions (b) Solutions

Figure 4.5: Distribution of document lengths in the solution and problem description parts of the AIR dataset.



(a) TASA900 (b) AIR problem descriptions

Figure 4.6: Distribution of sentence lengths in the TASA900 and AIR datasets.

graph-based representations in the evaluation. It is clear that the average length of sentences is larger in AIR than in TASA900, but not significantly so.

4.3.5 The Implementation

The remaining sections of the chapter briefly outlines our implementation of the representations and their evaluation experiments. Our experimental setup can be divided into two main phases: *document preprocessing* and *graph construction and evaluation*. The purpose of the former is to make the documents ready to be processed by the graph construction modules, and the latter creates the graph-based representations and evaluates these. Each phase is described separately in in the following two sections. For a more detailed discussion of the experimental framework and its implementation, see Appendix B.

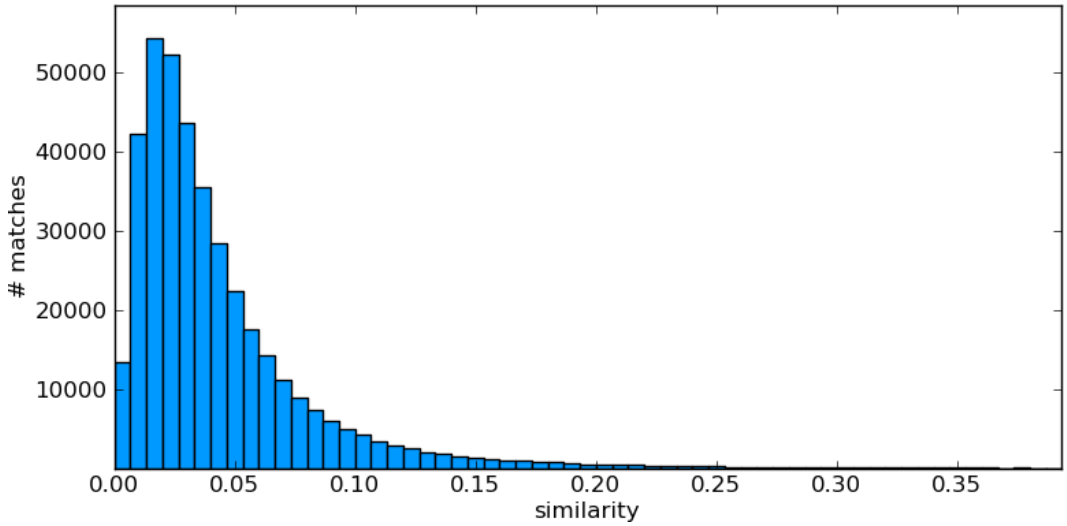


Figure 4.7: Distribution of *solution similarities* in the AIR dataset.

Document Preprocessing

The purpose of the first phase is to do as much preprocessing as possible of the documents in the dataset, before the construction of the networks. The preprocessing steps are the same, regardless of the details of the networks being evaluated, and this way it does not have to be done each time an experiment is run.

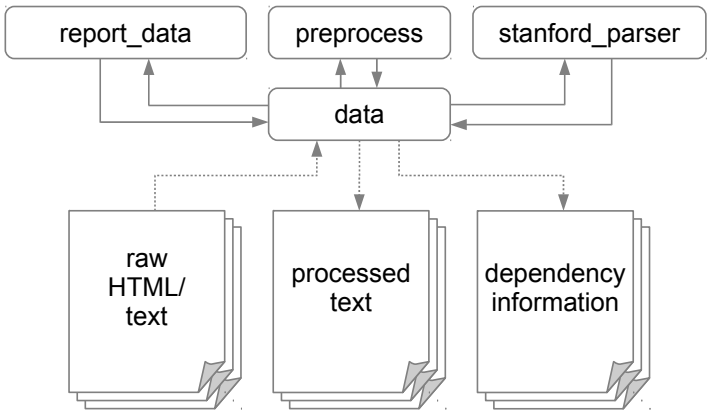


Figure 4.8: Document preprocessing.

The flow of data through the modules involved is illustrated in Figure 4.8. The `data` module is responsible for reading datasets from disk, and create preprocessed copies of them. To do this, it use several helper modules for various parts of the processing effort. The `report_data` module is used to read aviation incident reports in HTML format. Most natural language processing steps, e.g. stemming or case folding, are done by the `preprocess` module. To identify dependencies within sentences, for use in the dependency networks, the `stanford_parser` module is used.

The type of preprocessing needed differs for the network types, and thus different versions of the dataset are created. For co-occurrence networks, preprocessed text is stored. This is typically text where stop-words have been removed, and the remaining words stemmed and filtered in various ways. For the dependency net-

works, dependency information about each document is extracted and written in a serialized format to file for easy use later.

Graph Construction and Evaluation

Performance of the different representations are studied in the second phase. This is visualized in Figure 4.9. Data is first loaded from preprocessed versions of the dataset, and network representations are then created and evaluated

The `data` module is again used to read data from files. It utilizes the `preprocess` module for any processing that could not be done in the preprocessing step above. Documents are then passed on to the two representation modules, and turned into different representations by the `freq_representation` and `graph_representation` modules. The `graph` module is responsible for graph calculations such as the various centrality methods, and is used by `graph_representation`.

In the final stage, the various representations are passed on to the `evaluation` module. They are here tested experimentally. The two evaluation methods, classification and retrieval, are each implemented in a separate module.

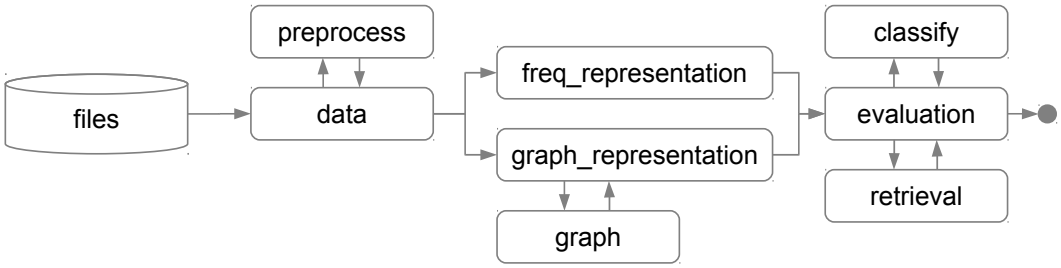


Figure 4.9: Graph construction and evaluation.

4.4 Summary

This chapter has presented the process of text representation based on text network models. Briefly described, the process consists of two stages. First, a network structure is constructed based on the text. We use terms from the text as nodes, and relations between these as edges. The following two chapters will explore different approaches to the identification of these relationships. Next, a term weighting measure is applied to the network to create a term-vector representation of the document. For this, we defined two weighting schemes: TC and TC-ICC. Both are based on the use of node centrality measures presented in the preceding chapter.

Our methods and framework for testing the graph-based representations were presented next. Two evaluation methods are employed, one a classification task and the other an unsupervised evaluation task based on document retrieval. These tasks will be used both to explore various aspects of the graph-representations in the following two chapters, as well as in the final evaluation experiments in Chapter 7.

Chapter 5

Co-occurrence Networks

This chapter explores various aspects of co-occurrence networks, based in part on our earlier studies and partly on new experiments. We look at various issues such as how the edges should be represented, how textual contexts should be defined, and whether higher order co-occurrences should be employed. The networks are evaluated using the measures introduced in Section 4.3. The goal of the chapter is to arrive at a co-occurrence representation well suited as a basis for the task of document similarity measurement and TCBR retrieval.

The chapter starts by presenting the basics and most important aspects of co-occurrence networks in Section 5.1. It is this section that is based on our earlier studies. Section 5.2 introduces new ideas, and evaluate whether they contribute to the representation. In Section 5.3 the TC-ICC measure is evaluated with the representation, in order to determine whether the use of information global to the corpus is useful when determining term importance. The co-occurrence representation is analyzed in Section 5.4. Graph properties such as connectedness, the small-world property and the degree distribution is studied, and the terms identified as important by the representation is examined. Finally, the main findings are summarized at the end of the chapter.

5.1 Basic Representation

This section describes the basic aspects of co-occurrence networks. We discuss how to construct the networks, and how to handle various aspects of the representation. The discussion is based on results from experiments performed as part of earlier work (Valle, 2010).

5.1.1 Construction

The co-occurrence network is one of the simplest and most intuitive ways to use graphs to represent text and is, not surprisingly, widely used.

A co-occurrence network is a graph where the nodes represent lexical units from the text, and are linked if the lexical units occur within a common context. We will use terms as our basic lexical unit, and thus nodes in the network represents unique terms from the documents. Contexts can be defined as the part of the text surrounding terms, e.g. paragraphs or sentences.

Since terms that appear close together in the text are linked, co-occurrence networks are able to retain some structure from the text. This is information that is completely discarded by the bag-of-words model, which underlies common frequency-based techniques such as TF and TF-IDF.

The algorithm to create a co-occurrence network using a sliding n-word window is presented in Algorithm 5.1. The context window of each word consist of the n subsequent words in the text (line 4). The UNIQUEWORDS subroutine called on line 1 lists the set of distinct terms from the text. The subroutine called on line 6, UPDATEEDGE, handles the update of the network for each co-occurrence.

The process is illustrated by an example in Figure 5.1. A small networks is constructed from the sentence "*A B C A D B E F*", where each letter represents a term. The graph starts out with a set of nodes, but with no edges. Then, for each term in the sentence, a context window of the $n = 2$ following terms are identified, and edges are created from the term to each of the terms in the context window. This corresponds to the for-loop in lines 5–7 of Algorithm 5.1.

Algorithm 5.1 Create co-occurrence network from text

Input: *text*: list of words
Input: n: size of the context window
Output: *graph*: co-occurrence network
1: $nodes \Leftarrow$ UNIQUEWORDS(*text*)
2: $edges \Leftarrow \emptyset$
3: **for all** $word_i$ in *text* **do**
4: $window \Leftarrow text[i+1, i+n]$
5: **for all** $word_j$ in *window* **do**
6: UPDATEEDGE(*edges*, $word_i$, $word_j$)
7: **end for**
8: **end for**
9: **return** $graph \Leftarrow \langle nodes, edges \rangle$

5.1.2 Preprocessing

Before the co-occurrence networks can be created, the text needs to be preprocessed. The level of preprocessing was determined through empirical experiments. The text is first case-folded, i.e. all upper case characters are made lower case. Tokenization is then done, turning the text string into a series of tokens. These tokens are filtered, in order to remove some potentially bad nodes. This is done by removing all stop-words, and all words containing numbers and less than three characters. The tokens are finally stemmed using the Porter (1980) stemming algorithm. In the case of sentences used as contexts, the sentence boundaries are determined prior to tokenization.

5.1.3 Centrality Measure

Through experiments with different centrality measures on classification, we found that the choice of graph centrality measure and the properties of the graph repre-

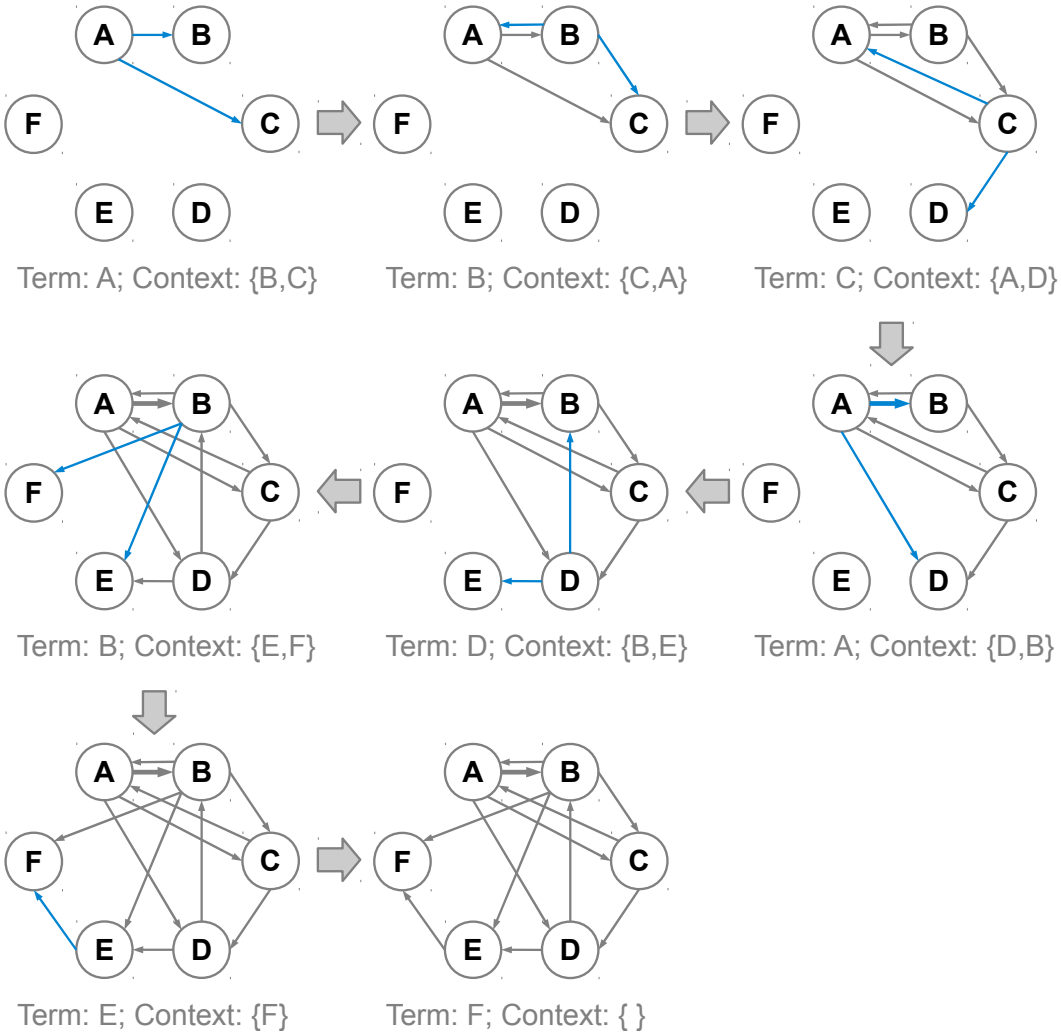


Figure 5.1: Example of co-occurrence network being constructed from the sentence "A B C A D B E F", using context window-size $n = 2$.

sentation such as edge weights and directedness, were highly interdependent.

We found both PageRank and the group of degree-based centrality measures to perform consistently good, while the rest of the measures performed variably and often poorly. The group of degree centrality measures consists of weighed and unweighted versions of the degree centrality, including in-degree and out-degree in the case of directed networks.

Based on these results, we will consider only PageRank and degree centrality as candidates for the co-occurrence network representation. Because we have found the choice of context to greatly influence the performance of the centrality measures, the decision about which of the two to use in the final representation is delayed until Section 5.2.1, where we will evaluate different types of textual contexts.

5.1.4 Edge Direction

When creating an edge between two terms as nodes in a co-occurrence network, there are basically three choices regarding the edge direction. The forward direction is to follow the flow of the text, i.e. $word_i \rightarrow word_j$ if $word_i$ appears before $word_j$.

Conversely, backward directed edges go against the flow of the text. The third option is to leave the edge undirected.

For the best performance, it is important to select a directedness of the graph edges that suits the centrality measure used. For degree-based centrality measures, undirected edges are best. For betweenness-based centralities, it is better to use directed than undirected edges. The direction of the edges, however, seems to matter little. The results obtained for closeness-based centrality measures were not conclusive, and no edge type was found to perform significantly better than the others. Neither were the results for centrality measures based on eigenvector calculations easy to interpret. Forward and backward directedness sometimes outperform undirected undirected edges, but this seems to depend on factors such as context size, centrality measure and dataset. Undirected edges constantly perform reasonably well, and seem to be the best and safest choice for this group. Considering that we will be using degree centrality or PageRank, it seems that undirected edges will be the best choice.

The choice of undirected edges is also supported by Mihalcea and Tarau (2004), who did similar work with co-occurrence networks in their TextRank model (more on this in Section 4.1.1). They report the following from their experiments (Mihalcea and Tarau, 2004, page 408).

> *Regardless of the direction chosen for the arcs, results obtained with directed graphs are worse than results obtained with undirected graphs, which suggests that despite a natural flow in running text, there is no natural "direction" that can be established between co-occurring words.*

Although this conclusion is not supported by all our experiments, our results agree that undirected edges are best when PageRank is used, which, as its name suggests, the TextRank model do.

5.1.5 Edge weights

We have also evaluated whether the use of edge weights, that is, information about the frequency of co-occurrences, is useful to include in the representation. We found that those centrality measures able to take into account weight information for edges in most cases outperformed their unweighted equivalents. A notable exception from this rule was the load centrality measure, which constantly performed best without weights. For degree centrality the weighted version performed better than its unweighted counterpart only some of the time.

Given the use of PageRank, our representation will benefit from the use of edge weights. It is not unreasonable to assume that this added information should be beneficial, since it separates random and infrequent co-occurrences from those that appear often.

5.2 Improvements

Throughout the preceding section we presented the basics of the co-occurrence network representation. This section presents a couple of new ideas, and experiments

to evaluate whether they prove useful in improving the basic networks. We will first consider sentences as an alternative to n-word windows as textual contexts. Next, in Section 5.2.2, we explore a potential new source of relations for the networks. The idea is to use indirect, or higher-order, co-occurrences. That is, we investigate whether it is beneficial to connect terms that do not occur together directly, but have indirect relations through their common neighbors.

5.2.1 Textual Contexts

Two usual choices for contexts are sliding n-word windows and sentences. In our previous experiments we used windows as contexts. We found that smaller context windows generally performed better than large ones. Although depending highly on the centrality measure used, we found that good values for n usually were found in the range from 2 to 5. Based on this observation we expected the use of sentences, which usually are significantly longer than this, might perform worse than windows.

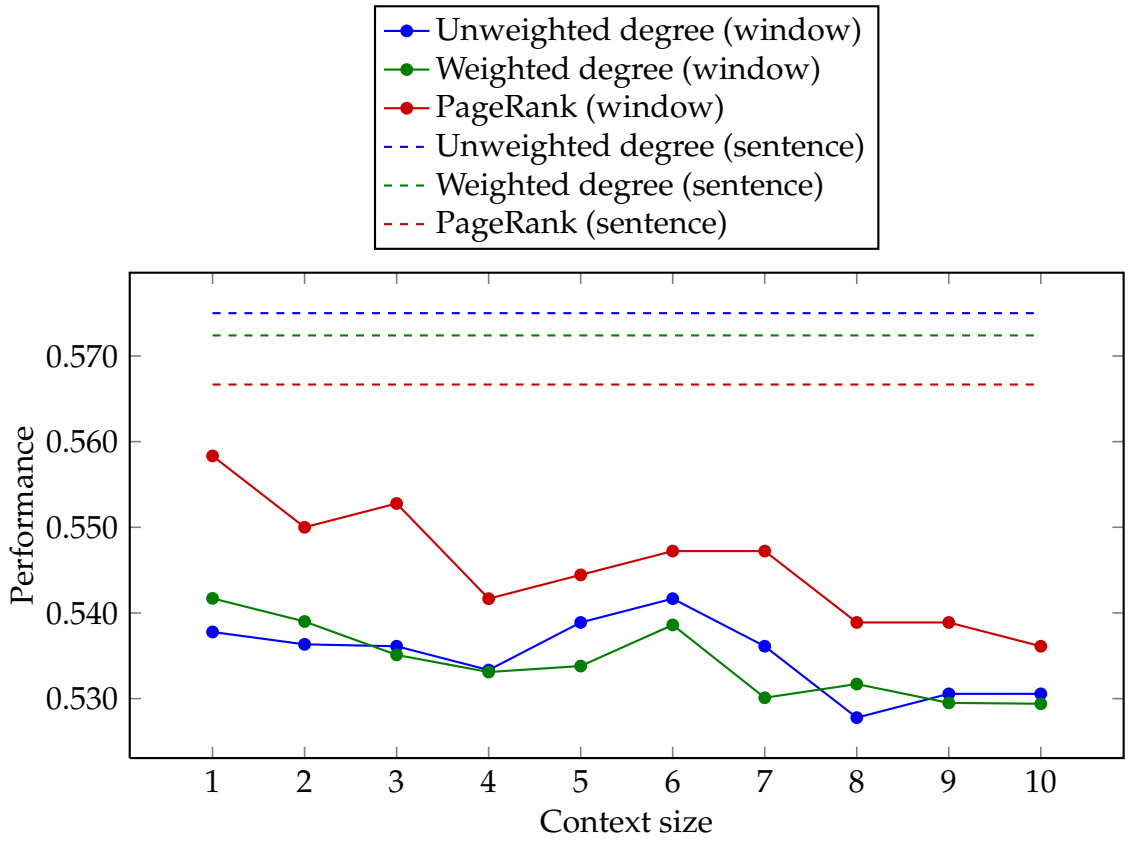
The use of larger contexts also increases the density of the network, as the number of edges needed to represent each sentence grows in the worst case exponentially with the size of the context window. In some corpora, the average sentence length can be quite high. The use of smaller sliding n-word context windows are thus preferable also from a performance point of view.

To test our assumptions, we designed an experiment to test how the different choices of contexts affect the performance of the networks in the evaluation tasks. Co-occurrence networks were created using contexts as windows with $n = 1 \ldots 10$, and as sentences. PageRank and both weighted and unweighted degree centrality measures were used.

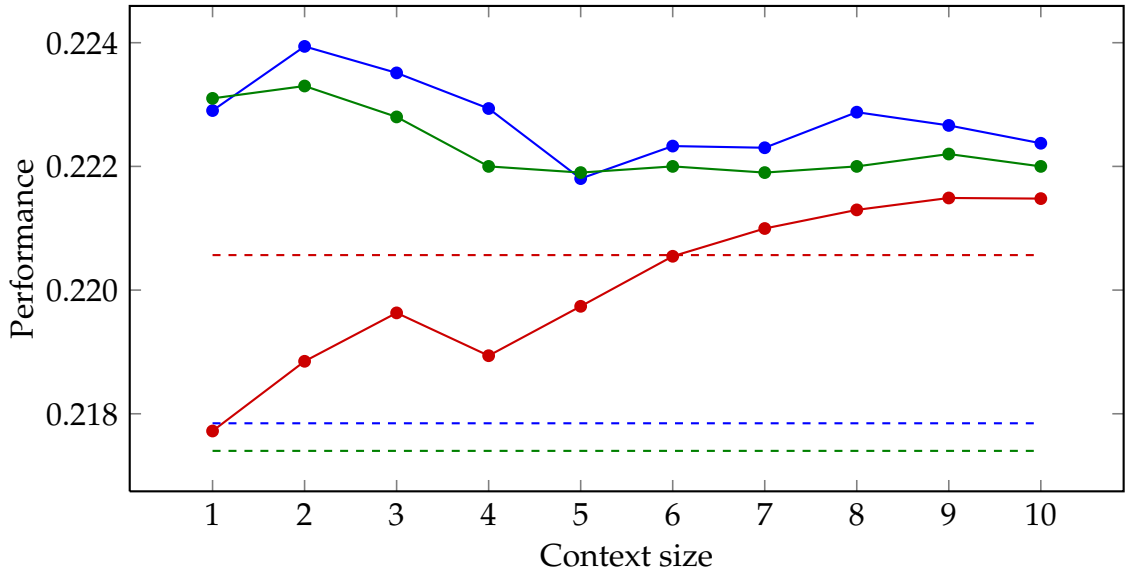
The results are shown in Figure 5.2, and lead to several observations — some quite unexpected:

1. Sentence contexts are best suited in the classification task and window contexts are best in retrieval. The best performance in retrieval is achieved with window size $n = 2$.
2. Degree centrality performs better than PageRank in both tasks. PageRank performs better than degree in classification when windows are used for contexts, but is otherwise inferior to degree centrality. Unweighted degree is consistently better than weighted degree.
3. While the trend seems to be decreased performance with higher values for n, the opposite is the case for PageRank in the retrieval task. PageRank clearly show increased performance with the larger sizes.
4. Although their performances differ, PageRank and degree centrality seems to follow similar curves for context windows.

The most surprising observation is that sentence-based contexts perform far better than window contexts in classification. As we reason above, larger contexts introduce meaningless relations, which is clearly demonstrated by the larger window sizes in the plots. That sentences still perform better, must mean that many of the meaningless relations must occur across sentence boundaries. The reduced performance of the sentence contexts in retrieval might be caused by the fact that



(a) Classification



(b) Retrieval

Figure 5.2: Evaluation of context types.

the sentences generally tend to be longer in the dataset used here (see Figure 4.6 on page 46).

From the second observation, we conclude that unweighted degree centrality is the better measure for co-occurrence networks, as long as the context type is chosen accordingly.

	Classification			**Retrieval**		
Context size, n	20	40	80	20	40	80
PageRank	0.539	0.539	0.536	0.221	0.222	0.221
Unweighted degree	0.528	0.531	0.531	0.223	0.223	0.222
Weighted degree	0.524	0.530	0.531	0.222	0.222	0.222

Table 5.1: Evaluation of large context sizes n.

The increase in performance with PageRank for larger n in retrieval is interesting. In order to understand whether these trends continue for even larger contexts, we tested sizes $n = 20, 40, 80$ for both degree centrality and PageRank. The results are listed in Table 5.1. We see that the results vary very little, which means that the performance has converged and changes in the context sizes beyond $n = 10$ have little or no effect. This would seem to indicate that there is a limit to the penalty for having too large contexts, and conversely, the benefit when using PageRank for retrieval.

The last of our observations is perhaps the least surprising one. Although PageRank and degree are different measures, they essentially try to measure the same thing, the importance of each node in the network, and they do this by considering the neighbors of the nodes. As described in Section 3.3.8, PageRank can in some sense be seen as an expansion of the degree centrality which take into account the importance of the neighbours, not only their number. Given this, it is to be expected that they will have somewhat correlated results.

We conclude that unweighted degree centrality should be used with the co-occurrence networks, and that the best choice of context type depends on the intended task. When classification is performed, co-occurrence networks based on sentence contexts perform better, while 2-word context windows are a better choice for retrieval.

5.2.2 Higher Order Co-occurrences

The co-occurrence networks presented so far in this chapter can be denoted as first order co-occurrence networks. By this, we mean that the relations captured represent first order co-occurrences, i.e. co-occurrences directly between words within the same context. Chakraborti et al. (2007) describe an algorithm for mining higher order co-occurrence relations between words. This section investigates whether this approach can be used to enhance our co-occurrence network representation. The approach is based on the theory that it is beneficial to use indirect, in addition to direct, associations between words. By capturing higher order co-occurrences, certain relations that would otherwise elude us can be identified.

This type of relations are illustrated by an example in Figure 5.3. Even though *A* co-occurs only with *B* in the contexts on the left hand-side, we can extract higher order relations to *C* and *D*, of orders 2 and 3, respectively. This is because *B* also co-occurs with *C*, which in turn occur together with *D*.

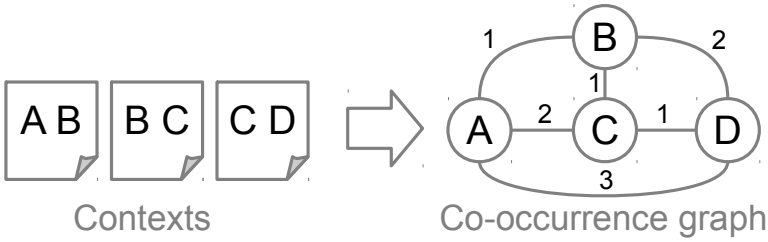


Figure 5.3: A simple example of a higher order co-occurrence graph. Edge weights indicate the order of co-occurrence between the terms.

Consider, for example, synonyms. In many cases, two words with the same meaning will not be used within the same documents. The words will, however, be used in the same way and in similar local contexts within each document, i.e. together with the same other terms. By identifying words that occur in similar contexts we can identify such words, and hopefully other similar term relations. Concrete examples of this, reported by Chakraborti et al. (2007), are near-synonyms such as *road* and *street*, and *internet* and *web*.

The approach presented by Chakraborti et al., is intended for co-occurrences in document collections, hence their contexts are entire documents. But, as they point out (Chakraborti et al., 2007, p.64), "*the context can be localized to arbitrary length windows or sentences to restrict the number and scope of mined associations*", which is exactly what we be doing.

Related Work

Several people have looked at higher order co-occurrences, both theoretically and their practical use.

An example of the latter is χ-Sim, a co-similarity based clustering algorithm presented by Bisson and Hussain (2008), where word similarities are based on higher order co-occurrences. χ-Sim is based on the idea of simultaneously generating similarity matrices between rows and between columns, documents and terms, respectively, iteratively basing one on the other. For each iteration, co-occurrences of higher orders are found, with the nth iteration corresponding to detection of co-occurrences of order n. Hussain and Bisson (2010) also use higher order co-occurrences to incorporate class knowledge into the learned similarity matrices created by χ-Sim. They do this by forcing second order co-occurrences between words within the same class by adding dummy words.

Higher order co-occurrences have also been studied in the field of cognitive neuroscience. Livesay and Burgess (1998) looked at higher order relations in light of Mediated Priming (MP). MP is the effect in which spreading of activation happens in semantic memory networks by means of related concepts. To use the example by Livesay and Burgess, in the prime-target pair *lion-stripes*, priming occurs because *lion* is closely related to *tiger*, which of course is related to *stripes*. Thus, by being

prompted by *lion*, the concept of *stripes* is primed in our memory. Their results indicate that MP works by indirect relations between concepts, and compare this to context vector representations.

Lemaire and Denhière (2006) have also looked at the mental activation of one term when another is presented, the strength of which they define as semantic similarity. Although it is proven that co-occurrences and semantic similarity are highly correlated, their experiments indicate that frequencies of co-occurrences tend to overestimate the semantic similarity. They also investigate the role of higher order co-occurrences in relation to similarity. They found that such relations tend to increase similarity.

Calculation of Higher Order Co-occurrences

A *first order* co-occurrence is said to exist between word A and word B if both occur within the same context. These are the kind of co-occurrences on which we have based our representation in the previous sections. If words B and C also co-occur within a context, there is a *second order* co-occurrence between A and C. Further, a *third order* co-occurrence exist between A and a new word D, if word D then co-occurs with C.

We capture the higher order relations in term-term matrices. The calculation is done as follows.

1. Create T_0, matrix of first-order co-occurrences as before.
2. Create an updated first-order co-occurrence matrix T from T_0 by
   (a) converting all non-zero values to 1, and
   (b) setting the diagonal values to 0.
3. Create the second-order co-occurrence matrix as $T_2 = T^2$.
4. Create initial third-order co-occurrence matrix as $T_3 = T^3$.
5. Calculate n as the vector of column sums of T.
6. Create discount matrix D as $D_{i,j} = n_i + n_j - 1$ for all $i \neq j$.
7. Create revised third-order matrix as $T_3' = T_3 - D \times T$, where $\times$ is pointwise multiplication.

Given a third order co-occurrence between terms A and D through intermediate terms B and C, we must ensure that neither B nor C are the same as A or D. Step 2b ensures that A is unlike B, and C unlike D. In addition, it must be ensured that C is not the same as A, and B is not the same as D. This is what is being done through steps 5–7. By subtracting $D \times T$ from T_3, we enumerate and eliminate the invalid paths of types *A-B-A-D* and *A-D-C-D* from the final representation.

Next, we present a simple example to illustrate the procedure.

Example

The following example illustrates the process of identifying higher-order co-occurrences by calculate the described matrix operations.

We start by creating a (first order) co-occurrence matrix of $|terms| \times |terms|$ elements from the document, T_0. Our example document in this case has four distinct

terms. This matrix is then converted into the first order matrix, T, according to step 2 above.

$$T_0 \overset{\text{Step 1}}{=} \begin{pmatrix} 0 & 1 & 5 & 3 \\ 0 & 2 & 3 & 1 \\ 2 & 1 & 0 & 0 \\ 2 & 1 & 0 & 2 \end{pmatrix} \xrightarrow[\text{non-zero to 1}]{\text{Step 2a}} \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \xrightarrow[\text{diagonal to 0}]{\text{Step 2b}} \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} = T$$

Based on the first order matrix, the second order matrix is then calculated as $T_2 = T^2$. T_3, the initial third order matrix is similarly calculated as T^3.

$$T_2 = T^2 \overset{\text{Step 3}}{=} \begin{pmatrix} 2 & 2 & 1 & 1 \\ 2 & 2 & 0 & 0 \\ 0 & 1 & 2 & 2 \\ 0 & 1 & 2 & 2 \end{pmatrix} \quad \text{and} \quad T_3 = T^3 \overset{\text{Step 4}}{=} \begin{pmatrix} 2 & 4 & 4 & 4 \\ 0 & 2 & 4 & 4 \\ 4 & 4 & 1 & 1 \\ 4 & 4 & 1 & 1 \end{pmatrix}$$

The columns of T are then summed in n, and the discount matrix D calculated as $D_{i,j} = n_i + n_j - 1$.

$$n \overset{\text{Step 5}}{=} \begin{pmatrix} 2 & 3 & 2 & 2 \end{pmatrix} \quad \text{and} \quad D \overset{\text{Step 6}}{=} \begin{pmatrix} 0 & 4 & 3 & 3 \\ 4 & 0 & 4 & 4 \\ 3 & 4 & 0 & 3 \\ 3 & 4 & 3 & 0 \end{pmatrix}$$

Finally, the third order matrix, T_3, is adjusted according to the discount matrix, resulting in the revised third order matrix T_3'.

$$T_3' = T_3 - D \times T \overset{\text{Step 7}}{=} \begin{pmatrix} 2 & 0 & 1 & 1 \\ 0 & 2 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

The results of the process are the matrices T, T_2, and T_3', which holds first, second and third order co-occurrences between the different terms, respectively. Based on one or more of these, higher order co-occurrence networks can be created.

Relation to Latent Semantic Indexing

It is natural to compare the higher order co-occurrence relations just described to Latent Semantic Indexing (LSI) (Deerwester et al., 1990). LSI is a text-mining algorithm using a mathematical technique called Singular Value Decomposition (SVD) to create vector space representations of documents. The technique is claimed to bring out latent semantic relations in document collections, similar to our higher order co-occurrences.

LSI works by identifying patterns of relationships between the terms/concepts contained in documents. It is based on the idea that words used in the same contexts tend to have similar meanings. This is done by reducing the matrix of document-term associations to a lower number of dimensions. In the resulting matrix, space is arranged to reflect the major patterns in the data. As a result, terms may end up close even though they were not in the original data.

LSI is centered around the term-document matrix, just like higher order co-occurrences as presented by Chakraborti et al. (2007). Like we have done with co-occurrences, LSI may similalarly be adopted to use smaller contexts instead of documents.

Kontostathis and Pottenger (2003) have shown that the values produced by SVD correlate highly with second order term co-occurrence, and present a proof that information regarding higher order co-occurrences are used by the SVD algorithm. Terra and Clarke (2003) studied the use of word similarity measures based on methods for estimating word co-occurrence frequencies. Their results showed improvement over similar methods based on LSI.

One of the main differences is that a higher order co-occurrence explicitly captures the higher order associations, while LSI does so implicitly. It is possible to assess the similarity between two terms using LSI by comparing their context vectors. This gives us, however, little information about why they have a particular level of relatedness. Higher order co-occurrences, on the other hand, provide more information. Depending on how the co-occurrences are calculated, we can obtain information about the level of their co-occurrence and their common contexts.

From Matrix to Graph

Given the matrices of higher order co-occurrences as calculated above, the next question is how to use them in the network representation. The two problems that need to be addressed are:

1. Which of the higher orders should be used?
2. How should different orders of relations be represented in the network?

The first of these questions will be answered in the following section, where the different combinations of orders is evaluated empirically. By *combination of orders*, we mean the set of higher orders that are represented in the network. As an example, $\{1 + 3\}$ indicate the combination of the first and third order relations. Thus, no second order relations would be included in theis network.

For the second question, there are several aspects to be considered. First, whether to use multigraphs or regular graphs. That is, should each order be represented by a separate edge between nodes, or should the presence of co-occurrences on multiple levels be recorded on a single edge between nodes. Second, the relative importance of each of the orders must be decided. The importance can be reflected in the edge weights, or by making sure the applied centrality algorithm treats different types of edges differently.

We have decided to combine the different order relations into single edges between nodes (i.e., not to use multigraphs), and to represent the relative importance of each order by their contribution to the overall edge weight. This representation is chosen because it allows us to use the same centrality algorithms as before. This is also the approach used by Chakraborti et al. (2007). The weight for each edge is determined by a linear combination of occurrences of each of the first three orders:

$$w_{u,v} = \alpha * \text{first\_order}_{u,v} + \beta * \text{second\_order}_{u,v} + \gamma * \text{third\_order}_{u,v} \qquad (5.1)$$

We will start by evaluating all possible combinations of orders, with each order being equally important ($\alpha, \beta, \gamma = 1.0$). Then we compare these results to the case where $\alpha = 1.0$, $\beta = 1.53$, and $\gamma = 1.51$. This is the average of the combinations found to perform best by Chakraborti et al.. It would be interesting to study different combinations of weights in more detail, but this is not done here, as such a study is beyond the scope of this thesis.

Evaluation

Other than the use of higher order relations as just described, the representation is exactly the same as in the previous sections: edges are undirected, and we use sentences and 2-word windows as contexts for classification and retrieval, respectively. The representations are tested in the usual manner, as described in Section 4.3.

The representations using $\alpha, \beta, \gamma = 1.0$, are tested first. Table 5.2(a) lists the performance of classification and retrieval using all combination of orders.

From these results the use of higher order co-occurrences does not seem to be beneficial. Compared to the results for classification and retrieval achieved without the higher order approach, none of the results obtained here indicates any improvement. For both classification and retrieval, use of orders higher than the first result in decreased performance in all cases. Combinations including the third-order relations have especially poor performance.

This outcome is both somewhat disappointing and surprising. Chakraborti et al. (2007) reported that use of higher orders significantly improved their results, and even outperformed LSI. It may be expected that the combinations excluding the first order relations would show reduced performance, but not even the combination of the first two orders outperformed standard co-occurrence. This might possibly be explained by the fact that the weighted combination of strengths for the different orders of co-occurrence may not have been the optimal one. We attempt to remedy this by using weights based on those found by Chakraborti et al. (2007) in a new run of the experiment.

Table 5.2: Results of higher order co-occurrence experiments.

(a) $\alpha, \beta, \gamma = 1.0$

Orders	Classification	Retrieval
$\{1\}$	**0.5750**	**0.2239**
$\{2\}$	0.5500	0.2220
$\{3\}$	0.5389	0.2156
$\{1, 2\}$	0.5500	0.2220
$\{1, 3\}$	0.5417	0.2156
$\{2, 3\}$	0.5361	0.2156
$\{1, 2, 3\}$	0.5361	0.2156

(b) $\alpha = 1.0, \beta = 1.53, \gamma = 1.51$

Orders	Classification	Retrieval
$\{1\}$	**0.5750**	**0.2239**
$\{2\}$	0.5500	0.2220
$\{3\}$	0.5389	0.2156
$\{1, 2\}$	0.5528	0.2217
$\{1, 3\}$	0.5444	0.2157
$\{2, 3\}$	0.5361	0.2164
$\{1, 2, 3\}$	0.5361	0.2162

Table 5.2(b) lists the results obtained using the weight combination $\alpha = 1.0$, $\beta = 1.53$, and $\gamma = 1.51$. As we see, some of the results have improved, but the changes are all minor and the higher order combinations still perform worse than the first.

It is possible that the weights used, although reportedly good in the experiments by Chakraborti et al., are unsuited for our datasets. We have not done a thorough evaluation of the possible weight combinations to test this, so it is possible that other combinations might lead to better results. Based on what we have seen so far, however, the prospects for higher-order co-occurrences do not look very good.

No combination of higher order relations performed better than the use of first order relations only. We conclude, therefore, that higher order relations do not benefit our representation, and will not be used as part of the co-occurrence network representation.

5.3 TC-ICC Weighting

Up to this point the TC measure, introduced in Section 4.2.2, has been used when converting the networks to centrality vectors for evaluation. TC is a simple measure, using information only from the document network itself. The more elaborate measure presented in the same section, TC-ICC, includes information about the term's overall centrality in the rest of the corpus.

Table 5.3 lists results for the classification and retrieval evaluation tasks when TC is replaced with TC-ICC. The network representation itself remains unchanged, only the method for extracting term importance values from the networks is changed. Because TC-ICC is a different approach to capturing term importance, we evaluate the results using all available centrality measures. This is done to determine whether the same centrality measures perform well with TC-ICC as with TC. For comparison, the best performance using TC, obtained with degree centrality, was 0.5750 for classification and 0.2239 for retrieval.

Values for current-flow betweenness were not obtained in the experiment. This is because this centrality measure is far more complex than the others in terms of memory requirements. The experiments were run on a computer with 12 GiB memory, which proved insufficient for current-flow betweenness on the corpus networks. Based on the poor performance of this measure using TC, we decided not to pursue the matter further.

Table 5.3: Performance of TC-ICC with co-occurrence networks.

Representation	Classification	Retrieval
Degree	**0.5333**	**0.2559**
Closeness	0.5167	0.1842
Current-flow Closeness	0.5111	0.1793
Betweenness	0.4333	0.1912
Current-flow Betweenness	n/a	n/a
Load	0.4222	0.1928
Eigenvector	0.5056	0.2068
PageRank	**0.5333**	0.2123
HITS-authorities	0.5083	0.2042
HITS-hubs	0.5083	0.2042

We see from the table that the measures that performed well with TC are also the best ones when TC-ICC is used. The degree centrality performs best in both tasks. PageRank performs equal to degree centrality in classification, and is the second best measure in retrieval. Thus, although we based the choice of centrality measure on evaluations done with TC, the results clearly apply to TC-ICC as well.

Compared to the performance of TC with degree centrality, TC-ICC performs poorly on classification, but far better than TC on retrieval. This seems to indicate that the use of global information in the representations does harm in the classification task, but can be useful for retrieval.

5.4 Network Properties

This section lists some properties of the co-occurrence networks described and evaluated in the preceding sections. Unlike the co-occurrence networks described in Section 3.4.1, we are not interested in the properties of networks representing the whole corpus, but rather how the networks we create for individual documents behave. The goal is to determine whether they behave according to the same basic rules as they do for larger networks.

Table 5.4 on the next page list some of the central network properties for networks created based on the TASA900 and AIR datasets. We report the average value for each property, as well as their standard deviation. Note that the networks are created differently for each dataset, as sentences are used contexts on TASA900, while two-word windows serve as contexts for AIR.

To compare the values for l and C, we constructed Erdős-Rényi random graphs with similar properties to those build from TASA900 and AIR. That is, the random networks have the same number of nodes N and mean degree $\langle k \rangle$ as the co-occurrence networks, but their edges are distributed randomly between pairs of nodes. The values for these are also included in the table.

5.4.1 Connectedness

We observe some characteristic differences between the TASA900 and AIR networks. While all individual networks based on the AIR dataset were connected, this applies to only 59% of the TASA900 networks. The average number of connected components found for TASA900 is 1.98, and varies greatly between the networks. This is a natural consequence of the different context types used on the two datasets. The networks based on AIR use sliding windows as contexts, which means that every word is at least linked to the word following it in the document. Thus, a path can be traced through the network corresponding to the sequence of words in the document. No such path is guaranteed with the sentence contexts used on TASA900, since each sentence is considered separately.

5.4.2 Small World Property

Since the networks span only single documents, they are small compared to those described in Section 3.4.1, with only about 96 and 405 nodes on average for each

Table 5.4: Some properties of the co-occurrence networks. The rightmost columns show values for random networks generated with the same order and size as those built from TASA900 and AIR.

	TASA900		**AIR**		**Random**	
	Mean	St.dev.	Mean	St.dev.	TASA900	AIR
N	96.054	20.420	405.610	156.305	96	405
$\langle k \rangle$	24.342	8.637	15.197	2.260	23.937	15.219
l	2.259	0.296	3.149	0.198	2.071	3.175
C	0.866	0.052	0.457	0.020	0.236	0.037

dataset. The average characteristic path lengths l and clustering coefficients C are, however, very similar to those reported for the larger networks. The very high average degree, compared to the number of nodes, in the TASA900 networks is caused by the larger contexts used to create these networks.

The random network comparable to the TASA900 networks has clustering coefficients $C_{\text{rand}} = 0.236$ and characteristic path length $l_{\text{rand}} = 2.072$. For the random network corresponding to the AIR networks, the values are $C_{\text{rand}} = 0.037$ and $l_{\text{rand}} = 3.175$. As we see, $C > C_{\text{rand}}$ and $l \sim l_{\text{rand}}$ in both cases. Although the difference between C and C_{rand} is not as pronounced as for the larger networks, these are still clearly small-world networks.

5.4.3 Degree Distribution

Contrary to the results from the research presented in Section 3.4.1, our networks seems to not be scale-free. We created co-occurrence networks covering both of the datasets, but the degree distributions of neither fitted power-law models — two-regime or otherwise. For the document networks, the results were similar, but varied considerably. Some of these networks seemed to be scale-free, while others clearly had distributions that did not follow power-laws. Also, for some of the document networks, the number of nodes were too small for us to determine with confidence whether they were scale-free or not. We refer to Appendix A for the details about the degree distributions, and our methods for testing for power-laws.

Although the networks turned out not to be scale-free, they share some of the properties of such networks, like the presence of hubs. This fact, more than the overall distribution governing the degrees, should be important for our representations. Any long-tailed degree distribution, power-law or not, implies that there will be some nodes with a very high degree. These hubs will also, depending on the used centrality algorithm, be very central in the network.

As to why our networks turned out not to be scale-free, we do not have an adequate explanation. This is a question that demands further investigation. The reason could either be that something about our representation or our datasets causes the networks to lose this property, or that the scale-free property is not as universal as previously assumed. Our implementation of the co-occurrence networks is identical to those described as scale-free in Section 3.4.1. Since the scale-free property should, in theory, not depend on the dataset used, and because we apply our method to two

different datasets, we feel confident that the results are not restricted to these data.

Ferrer i Cancho and Solé (2001), who originally identified the two-regime power-laws in co-occurrence networks, did not explain explicitly the method by which they tested the power-law model. They illustrated the fit on a log-log plot, but did not provide any statistical goodness-of-fit measure testing how well the data actually fit the model. Clauset et al. (2009) describe such a measure, and explain why linearity in log-log plots are insufficient to detect power-laws, and that such conclusions often are premature. We used the methods, and implementation, by Clauset et al. to test for power-laws for our representations.

5.5 Summary

We have presented a method for constructing co-occurrence networks by creating edges between words that appear close to each other in the same contexts within the text. This is motivated by that words appearing close together might be related, and that such networks retain information about the structure of the text, in contrast to the bag-of-words model which completely discards term order.

Based on previous experiments, undirected edges were chosen. The representations was found to generally perform better with weighted edges, representing the frequency of co-occurrences between pairs of terms, for the majority of centrality measures. The best centrality measure, unweighted degree, does, however, not utilize these weights.

Further experiments evaluated combinations of different contexts and centrality measures. Degree centrality was chosen as the centrality method of choice for both evaluations. Sentences were chosen as contexts for classification, and 2-word windows for retrieval.

Experiments performed on networks using higher-order co-occurrences revealed that the use of such relations did not increase the performance of the representations. Only direct, or first-order, co-occurrences are therefore employed by our co-occurrence networks.

Table 5.5 summarizes the central aspects of the co-occurrence representation.

Table 5.5: Summary of the co-occurrence network representation.

Property	**Description**
Nodes	Terms used in the text
Edges	Co-occurrences of terms within the same contexts
Contexts (classification)	Sentences
Contexts (retrieval)	2-word windows
Edge weights	Yes, number of co-occurrences
Edge directions	Undirected
Centrality	Unweighted degree centrality
Higher-orders	No, only first-order co-occurrences used
Text preprocessing	Stemming, token filtering, case folding
Stop-word removal	Yes

Chapter 6

Dependency Networks

The relations captured in the co-occurrence networks presented in the preceding chapter are simply *words that appear together* in the text. For any given pair of terms, it is hard to say exactly what the relation between them is. We saw, from the experiments described in Section 5.2.1, that co-occurrences crossing sentence boundaries might be irrelevant or even harmful to the representation in some situations. It is also easy to imagine that the importance of different term-relations within sentences differ, and that while some represent important characteristics of the document others are more or less arbitrary.

Motivated by this, this chapter presents the idea of *dependency networks*. These networks are based on explicit relations between terms, as defined by a *dependency grammar*. We use a dependency parser to identify the dependencies, which belong to one of several dependency types. To our knowledge, this is the first time graph centrality is applied to text networks built from word dependencies in order to measure document similarity.

The parser enables us to identify the links between those terms that have meaningful relationships, and say something about the character of this relation. Dependencies are identified between terms in the same sentence, which makes this roughly equivalent to the use of sentence contexts in co-occurrence networks.

The chapter is organized as follows. The first section discusses parsing of language, describes what dependency grammars are, and explain how dependency parsing is used to identify dependencies in sentences. The Stanford Parser, which is the parser used in this work, is described and illustrated with a practical example.

Dependency networks are investigated next. We start with a basic representation in Section 6.2 and gradually build on this through Section 6.3. The network properties of dependency networks are investigated in Section 6.5, before a summary of the dependency network representation concludes the chapter.

6.1 Language Parsing

Parsing is, in both computer science and linguistics, the process of syntactic analysis of text. The process analyzes a text consisting of a sequence of tokens (e.g. words), in order to determine its grammatical structure. This structure is defined by a grammar, i.e. a description of the relevant language.

Parsing works in much the same way for natural language as it does for computer programming languages, but with a difference in the characteristics of the grammar. Programming languages are artificial and easier to define formally than natural languages. They tend to be specified in terms of *context-free grammars*, since efficient parsers can be written for these.

A context-free grammar is a grammar that generates a formal language where clauses can be nested inside other clauses. The nesting can be done arbitrarily deep, but overlap between grammatical structures is not allowed. The grammars are called context-free because they can be expressed in terms of rules, $V \rightarrow w$, where all rules have a non-terminal on the left-hand side. The symbol on the right-hand side can always replace the non-terminal, regardless of the context in which it occurs.

Natural language is harder to define formally, since context often has influence on how a sentence should be parsed. The choice of syntax is affected by both linguistic and computational concerns. A good grammar needs to be able to express most or all of the possible sentences of a language, and must also support parsers that can be run with reasonable computational resources.

There is a rich variety of both linguistic theories and parsers for natural language. Some systems use *lexical functional grammars* (LFGs), but the parsing these is in general known to be NP-complete. LFGs view language as being made from several dimensions, each represented by a distinct structure defined by rules and concepts. Examples of such structures are grammatical functions, syntactic constituents, phonology, and morphological structures. The *head-driven phrase structure grammar* is another example of a popular formalism, but this is complex, and thus also computationally demanding. Simpler formalisms are often preferred for this reason. These types of grammars are called *phrase structure grammars*, as they divide sentences into phrases which are again divided into smaller components.

Another popular strategy is *dependency grammar parsing*, which is described fuller in the following section. Dependency grammars are distinct from phrase structure grammars by their lack of phrase nodes. They define structure simply as binary relations between words.

In contrast to parsers for programming languages, modern parsers for natural language are usually at least partially statistical. While formal grammars are based on syntax and production rules rules, stochastic grammars are trained from corpora of training data which has already been annotated. This allows the grammars to calculate information about the frequencies of various constructs in specific contexts. Such grammars typically use information about the tokens, such as their part-of-speech (POS).

A *probabilistic context-free grammar* (PCFG) is a context-free grammar where each production rule is augmented by a probability. The results of a parsing is the product of probabilities of the production rules used in the derivation. PCFGs can be seen as weighted context-free grammars, where the weights reflect the relative frequencies of each rule. In a small example grammar, the rule

$$0.62\ \langle \text{VP} \rangle \rightarrow \langle \text{V} \rangle\ \langle \text{NP} \rangle$$
$$0.38\ \langle \text{VP} \rangle \rightarrow \langle \text{V} \rangle\ \langle \text{NP} \rangle\ \langle \text{NP} \rangle$$

would indicate that a verb phrase consist of a verb followed by a noun phrase 62

percent of the time, while the verb in the the remaining 38 percent is followed by two noun phrases.

Statistical parsers search through the space of all candidate parses and parse the probability of each candidate. When this is done, the most probable parse of the sentence is chosen. A popular method for performing this search is the Expectation-Maximization (EM) algorithm.

6.1.1 Dependency Parsing

There are a lot of variations between the different theories of dependency grammar. We will not go into details about their differences here, only describe their basic common ground. This section concerns the theoretical background, while Section 6.1.2 describes a certain dependency model, the Stanford Dependencies, in more detail and with practical examples.

Although the roots of dependency grammars can be traced back many hundred years (Nivre, 2005), the starting point of modern theoretical dependency grammars is usually contributed to work by Tesnière (1959, referenced by Nivre, 2005). The dependency grammar is a class of grammars that share a certain basic assumption about the syntactic structure. In particular the assumption that syntactic structure consist of lexical elements linked by binary asymmetrical relations called dependencies. Because of this, the dependency grammars lacks phrasal nodes, which separates them from representations based on constituency. Dependency grammars also do not require a specific word order, and are thus suited for languages with free word order, such as Turkish.

Dependencies are defined as asymmetrical relations between two words, a *head* and its *dependent*. *Governor* or *regent* are alternate terms for *head*, and *modifier* is sometimes used instead of *dependent*.

Criteria for defining the relations, and for distinguishing the head from the dependent, are central concepts in dependency grammar theory. Many such criteria have been suggested and used. Some common criteria for identifying relations between head H and dependent D in a construction C are listed by Nivre (2005):

1. *H determines the syntactic category of C and can often replace D.*
2. *H determines the semantic category of C; D gives semantic specification.*
3. *H is obligatory; D may be optional.*
4. *H selects D and determines whether D is obligatory or optional.*
5. *The form of D dependes on H (agreement or government).*
6. *The linear position of D is specified with reference to H.*

The dependencies can be divided into three types: *morphological*, *syntactic* and *semantic*. Syntactic dependency can be distinguished in *endocentric* and *exocentric* constructions. An endocentric construction is one where the head can replace the whole without disrupting the syntactic structure. Exocentric constructions thus fail criterion 1, but may satisfy the rest.

A distinction is also made between *head-complement* and *head-modifier* relations in many dependency theories. Head-complement relations are exocentric, while head-modifier are endocentric.

While the head-complement and head-modifier structures usually have a fairly straight-forward analysis, there are also a group of constructions that have an unclear status. This includes constructions that involve grammatical function words such as articles or auxiliary verbs, and structures involving prepositional phrases. There is no general consensus for whether these should be regarded as dependency relations at all, and if so, what should be the head and what should be the dependent.

Another unresolved question is whether the notion of dependency is assumed to be not just *necessary* but also *sufficient* for analysis of the syntactic structure of natural language. This assumption was not made by Tesnière (1959), who included two complementary concepts in his model.

Although they are not mutually exclusive, two main types of strategies can be found for dependency-based systems for syntactic parsing: the *grammar-driven* and *data-driven* approaches.

Grammar-driven dependency parsing can, according to Nivre (2005), be further divided into two main trends. The first is closely related to context-free grammars, and can therefore use techniques from context-free parsing. This method is, however, limited in the types of dependency structures it can identify. The second is a formalization of dependency grammar in terms of constraints. Parsing is here viewed as a constraint satisfaction problem. There is also a third and simpler notion of dependency parsing, which is based on deterministic parsing strategies. This notion is motivated by the way humans do sentence processing and a desire to make more efficient syntax parsing. Parsing comes in different versions, but a common one is simple left-to-right parsing. It works by accepting words one and one from the beginning of the sentence, and then trying to link each word as head or dependent to every previous word.

Within the second approach, data-driven parsing, the earliest attempts were also grammar-driven in that they relied on an underlying grammar, and used corpus data only to train a probabilistic model for disambiguation. In essence, this was a PCFG model, where the context-free grammar defined dependency relations. This model performed poorly, and the approach became more successful once the need on formal grammars were removed. Data-driven dependency parsers take a machine learning approach to the grammar construction problem, and learn how to make good and bad parsing decisions solely from corpora of labelled data, without any intervention of underlying grammars.

Dependency information is useful in many tasks. Padó and Lapata (2007) describe a framework for constructing semantic space models based on text annotated with, among other things, dependency information. They show how this framework can be used for various cognitive and NLP tasks such as semantic priming, synonymy detection and word sense disambiguation.

As another example, Gao et al. (2004) describe the use of dependency relations in Information Retrieval. They create a dependency structure, an acyclic planar graph linking related terms, much like the dependency networks described later in this chapter. They define a language model based on the dependency structure, which defines the probability that a query q could be observed as a sample from a given document d. The most probable document is retrieved for the query.

Nivre (2005) describes the theory of dependency grammar in a lot more details,

and outlines the current state of the art in dependency parsing.

6.1.2 The Stanford Dependency Parser

To extract dependencies for our dependency network representation, we have opted to use the Stanford Lexical Parser[1], a statistical language parser created by The Stanford Natural Language Processing Group. We use version 1.6.5 of the parser in our work.

The Stanford typed Dependencies (SD) representation is designed specifically to be usable for people without linguistic background. It provides a simple description of grammatical relationships that are easily understood, and facilitate easy extraction of textual relations for our graph representations. As stated by De Marneffe and Manning (2008b, p.2),

> *All information is represented as binary relations. This maps straightforwardly on to common representations of potential users, including [...] graph representations (with labelled edges and nodes).*

The representation defines a set of 52 different relations for English. There are also versions of the parser for Chinese, Arabic and German, but these are not considered here. The relation types are listed with both full and abbreviated names in Table 6.1 on page 72. More detailed descriptions of each dependency, with examples, are provided in the manual (De Marneffe and Manning, 2008a).

In order to make the parser suitable for non-linguists, SD adheres to the following design principles for its relations (De Marneffe and Manning, 2008b, p.2):

1. *Everything is represented uniformly as some binary relation between two sentence words.*
2. *Relations should be semantically contentful and useful to applications.*
3. *Where possible, relations should use notions of traditional grammar for easier comprehension by users.*
4. *Underspecified relations should be available to deal with the complexities of real text.*
5. *Where possible, relations should be between content words, not indirectly mediated via function words.*
6. *The representation should be spartan rather than overwhelming with linguistic details.*

The relations are defined as a hierarchy with dependent (*dep*), the most generic grammatical relation, as the root. This relation is used when a more precise relation in the hierarchy does not exist or cannot be found by the system. The hierarchy is shown in Figure 6.1 on page 71, where the different groups of grammatical relation types can be seen together. Nodes in bold font are not actual dependencies, but rather represent categories of dependency types. The *pcomp* and *prepc* relations are shown in italics, because they are not explicitly defined by De Marneffe and

[1] Available from `http://nlp.stanford.edu/software/lex-parser.shtml`

Manning (2008a), but have been placed by us where they would seem to naturally belong.

SD was initially influenced by the theory of lexical functional grammar, especially in the set of grammatical relations and the naming of these. This intellectual debt is also reflected in the dependency hierarchy. Relations internal to noun phrases (NP) are an inherent part of much corpus text, and very useful in real-world applications. SD therefore define many such relations. Examples include *appos* (appositive modifier), *nn* (noun compound), *num* (numeric modifier), *number* (element of compound number) and *abbrev* (abbreviation). Such relations are good at capturing meaningful relationships between objects, concepts and properties.

SD comes in five variants, or modes, determining which relations are used and whether additional relations are inferred. The modes range from surface-oriented to more semantically interpreted representations. The former constructs a basic tree from tokens and dependencies, while the latter collapse preposition dependencies and include relations that may lead to cyclic dependency graphs. We briefly list the five variants below.

Basic: This is the simplest and most basic form of parsing which uses the dependency relations in Table 6.1 on page 72 to form a tree structure. Loops are not allowed, and all words in a given sentence, except one, is the dependent of only one other word. The last word is the head of the sentence and forms the root of the tree. Some of the dependencies are excluded in order to avoid cycles.

Collapsed: In the collapsed representation, some dependencies are collapsed into new ones. This is done for dependencies involving prepositions, conjuncts and others that form part of indirect relations between content words. Such indirect relations are discarded, and new relations directly linking content words are introduced. There are also some multi-word phrases that function like prepositions in English, and these are also collapsed.

An example of a collapsed dependency would be the creation of the new dependency `prep_in(airports, Canada)` from the old `prep(airports, in)` and `pobj(in, Canada)`.

This representation considers all the relation types, including those who may break the tree structure turning the dependency structure into a directed graph.

Collapsed with propagation: This variant is an extension of the collapsed representation, and thus use all dependency relations and collapses dependencies as described above. In addition, it does propagation of conjunct dependencies. This means that relations that exist on one part in a conjunct relation will be propagated also onto the other part.

For an example of this, consider the sentence *The pilot was certified and qualified*. A parsing of this sentence, with collapsed dependencies, would include `nsubjpass(certified, pilot)` and `conj_and(certified, qualified)`.

By propagating the *nsubjpass* relation onto the second part of the conjunction, we introduce the `nsubjpass(qualified, pilot)` dependency.

Collapsed tree: The collapsed trees collapse dependencies, but do not use any relations that may break the tree structure. This also means that propagation of conjuncts cannot be done.

Non-collapsed: In the non-collapsed representation every dependency relation is used, but no collapsing or propagation is performed.

Of these five options, we have chosen the *collapsed dependencies with propagation of conjunct dependencies* in our representation. This is because we believe the collapse of indirect relations to be a removal of unimportant information, and a good way to reduce noise. The propagation of dependencies is also good, since it includes new and meaningful relations into the representation.

The Stanford Parser also produce part-of-speech (POS) tags and a parse tree in addition to the dependency structure. This is demonstrated through the following example.

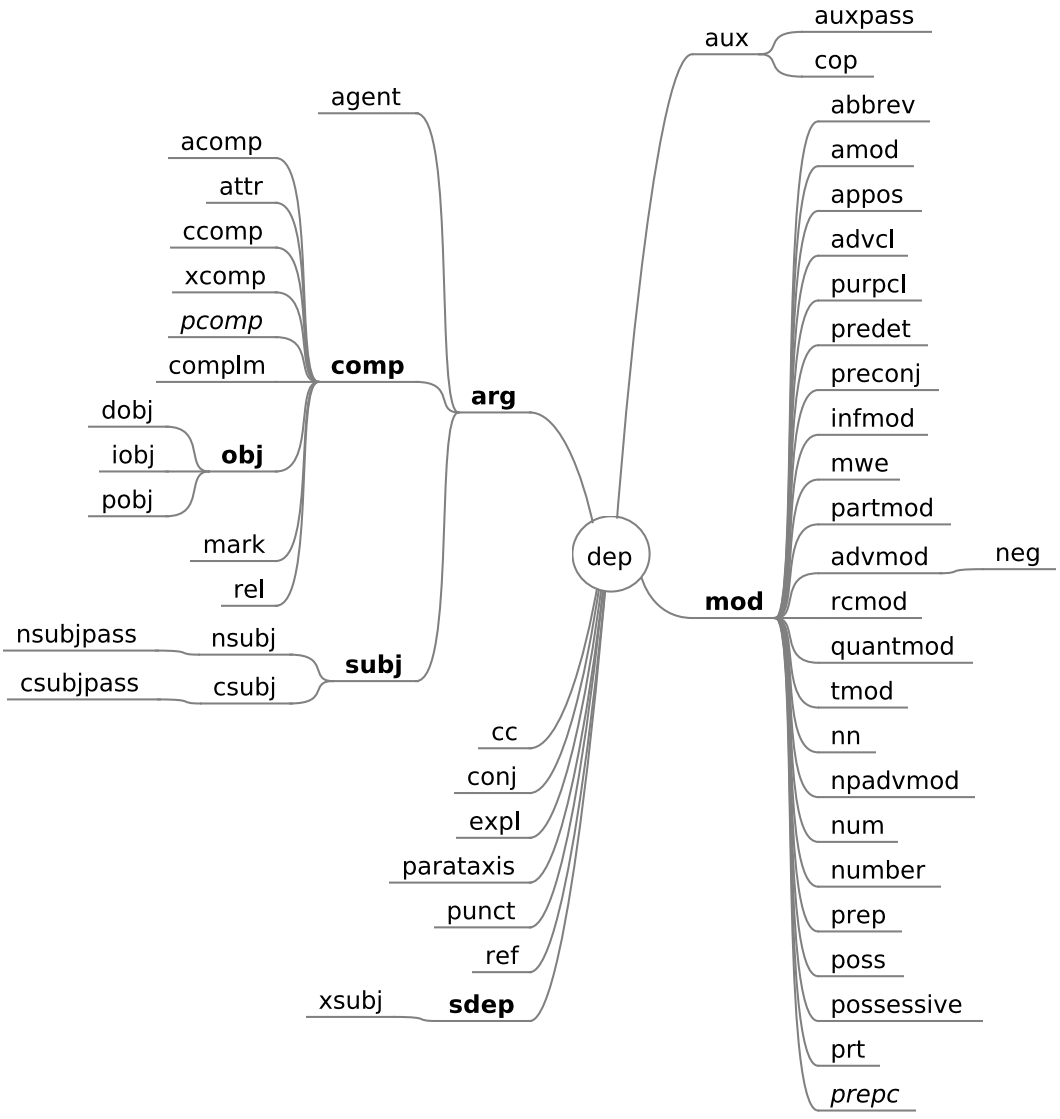


Figure 6.1: Hierarchy of dependency relations defined by the Stanford typed dependency representation.

Table 6.1: Stanford Dependency relations, sorted alphabetically.

Short	Full name
abbrev	abbreviation modifier
acomp	adjectival complement
advcl	adverbial clause modifier
advmod	adverbial modifier
agent	agent
amod	adjectival modifier
appos	appositional modifier
attr	attributive
aux	auxiliary
auxpass	passive auxiliary
cc	coordination
ccomp	clausal complement with internal subject
complm	complementizer
conj	conjunct
cop	copula
csubj	clausal subject
csubjpass	passive clausal subject
dep	dependent
det	determiner
dobj	direct object
expl	expletive (expletive "there")
infmod	infinitival modifier
iobj	indirect object
mark	marker (word introducing an advcl)
mwe	multi-word expression modifier
neg	negation modifier
nn	noun compound modifier
npadvmod	noun phrase adverbial modifier
nsubj	nominal subject
nsubjpass	passive nominal subject
number	element of compound number
num	numeric modifier
parataxis	parataxis
partmod	participial modifier
pcomp	prepositional complement
pobj	object of preposition
possessive	possessive modifier ('s)
poss	possession modifier
preconj	preconjunct
predet	predeterminer
prepc	prepositional clausal modifier
prep	prepositional modifier
prt	phrasal verb particle
punct	punctuation
purpcl	purpose clause modifier
quantmod	quantifier modifier
rcmod	relative clause modifier
ref	referent
rel	relative (word introducing a rcmod)
tmod	temporal modifier
xcomp	clausal complement with external subject
xsubj	controlling subject

A Practical Example

In order to demonstrate the parser, we illustrate the output of an example sentence found in one of the reports of the AIR dataset:

> *Immediately after the second touchdown, the pilot decided to perform a go-around.*

When parsed, the raw output from the parser given the above sentence is the following.

```
advmod(decided-9, Immediately-1), det(touchdown-5,
the-3), amod(touchdown-5, second-4), prep_after(decided-9,
touchdown-5), det(pilot-8, the-7), nsubj(decided-9, pilot-8),
xsubj(perform-11, pilot-8), aux(perform-11, to-10),
xcomp(decided-9, perform-11), det(go-around-13, a-12),
dobj(perform-11, go-around-13)
```

The head word is listed first in each dependency, and the dependent last. The number behind each term indicates its index in the parsed sentence, and is useful in cases where two occurrences of the same token needs to be distinguished from each other. Note that since we use the *collapsed dependencies with propagation*, the dependencies form a directed graph rather than a tree. This is evident by the appearance of `pilot-8` as dependent in both `nsubj(decided-9, pilot-8)` and `xsubj(perform-11, pilot-8)`, and is illustrated in Figure 6.2. In the graph representation we ignore the term-index of the tokens, so that the *the* node represents both `the-3` and `the-7` above.

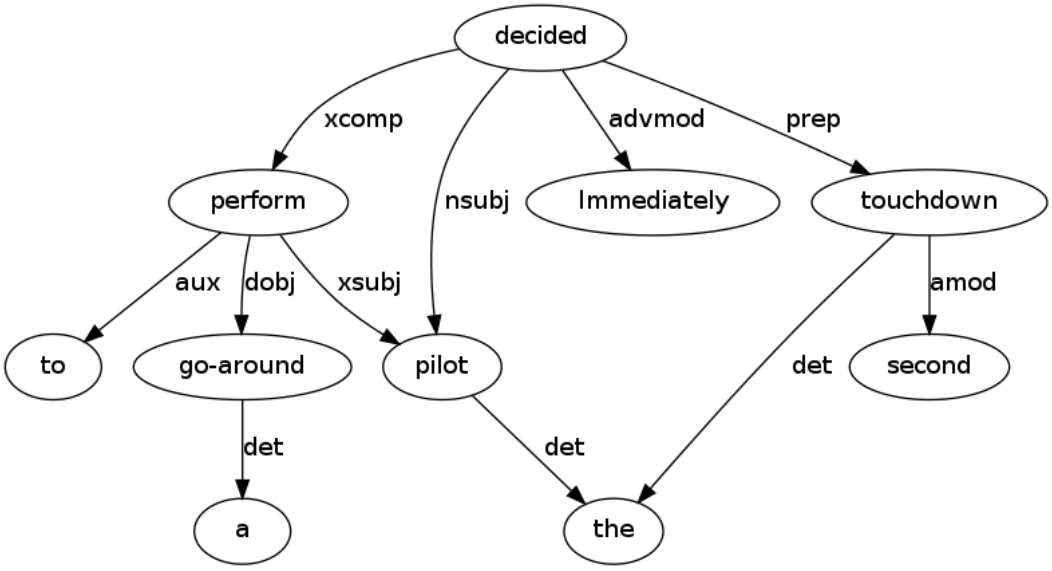


Figure 6.2: Example of a dependency graph using Stanford Dependencies.

The parser also provides a parse tree of the sentence structure, and POS-tags for each word. The parse tree is depicted in Figure 6.3 on the following page and, the sentence tagged with POS-tags looks as follows:

> *Immediately/RB after/IN the/DT second/JJ touchdown/NN , the/DT pilot/NN decided/VBD to/TO perform/VB a/DT go-around/JJ .*

The example sentence consist of 12 words, which gives us 66 co-occurrences. In contrast, only 11 dependencies are extracted — co-occurrences that carry meaningful information. Some of the relations mainly hold information about the syntactic structure, such as `det(touchdown-5, the-3)`, linking *touchdown* to its determiner. Other relations carry more semantic meaning. A good example of this is the `prep_after(decided-9, touchdown-5)` dependency, which represent the fact that something was decided after a touchdown. Which *touchdown* is indicated by the

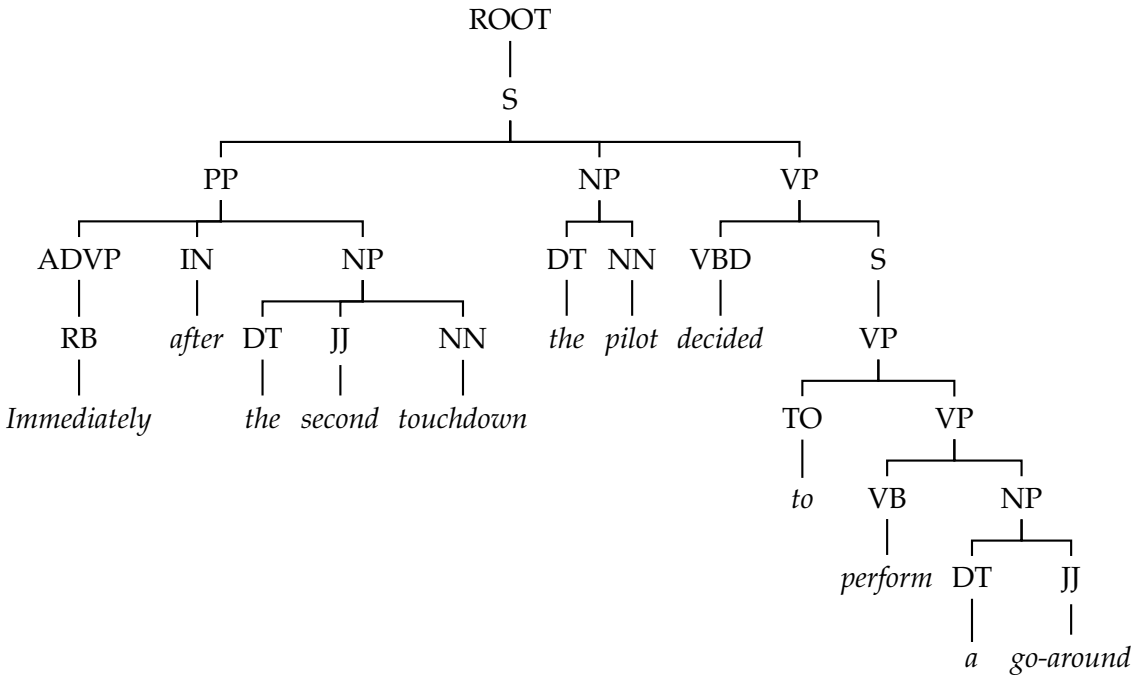


Figure 6.3: Example parse tree created by the Stanford Parser.

adjectival modifier `amod(touchdown-5, second-4)`, and details about the decision is included in several other dependency relations.

Practical Applications

The Stanford Parser has been used for many different applications, probably due to its user friendliness to people without any deep linguistic background. We present here one such example that both nicely illustrates the usefulness of dependencies, and relates to our intended use of the parser.

Ever since the completion of the Human Genome Project, there has been a rapid increase in publications on genetics. Most of this information exists only as free text, i.e. the publications, and is not contained in any structured databases. Fundel et al. (2007) have developed a system, RelEx, that extracts relations from free text. They use this system to identify relations between genes in articles published in the biomedical domain. Relations are extracted by applying a combination of dependency parse trees and rules to the text. They identify dependency relations containing terms recognized as names of genes, and apply the rules to determine whether a dependency is actually a meaningful genetic interaction. The central role of dependency parsing is to identify text passages starting and ending with gene names that are good candidates for the rule based extraction.

While RelEx only extracts and stores sets of interactions, Ozgür et al. (2008) take this one step further. By means of similar methods to those used by Fundel et al. (2007), they also extract gene interactions. The major difference is that instead of using rules, they apply the gene-dependency relations to a Support Vector Machine (SVM) classifier to determine whether the dependency describes an interaction between the gene pair or not. From the extracted gene interaction sets, they build interaction networks, with genes as nodes and their interactions represented by the

edges.

The motivation for their work is to apply network analysis to predict gene-disease interactions. In order to do this, they build disease specific gene-networks by specifying sets of genes that are known to be related to the disease and use these as training data for the SVM classifier. The gene interaction extraction method is more extensively described by Erkan et al. (2007). Once the disease specific network is created, they apply centrality metrics to the genes in the network. Their hypothesis is that the more central genes should be likely to be related to the disease. They evaluate degree, eigenvector, betweenness and closeness centrality, all of which show promising results.

This use of graph centrality in networks is similar to our approach to identify important terms within documents, as described in Section 4.2. Similarly, the gene-networks resemble the text-network representation which we will discuss shortly, in Section 6.2, the primary difference being that while Ozgür et al. (2008) extract only genes, we include the majority of the terms in the documents.

A Note on Parsing Long Sentences

The Stanford Parser uses considerable amounts of memory, and the use is roughly quadratic in the length of the sentences. Because of this, parsing some of the longer sentences in the documents cannot be done, because the Java Virtual Machine runs out of memory, even with increased amounts of heap space available. This affects sentences of more than around 120 words, depending on the specific sentence structure. Since such sentences are very infrequent, we simply skip these while parsing the reports.

6.2 Basic Representation

Our first approach towards creating dependency networks uses a fairly straightforward representation based upon dependencies extracted by the Stanford Parser. We use all the 52 dependency types as relations for the network. Unlike Ozgür et al. (2008) described above, we do not filter out any particular terms, but include all terms from the text as nodes. Initially, we ignore the order of head and dependent, making the network undirected. The following sections describe the construction of the networks, and an evaluation of this first approach.

6.2.1 Construction

The procedure for creating the basic dependency networks is fairly simple. Each document is parsed by the Stanford Parser. A set of dependencies is then extracted for each sentence in the document. For each dependency, two things are done.

First the terms, head and dependent, are added to the network as nodes if they have not been already. Before the terms are added, some processing is done on the terms. The processing includes case-folding and stemming, and is done in order to make sure different forms and inflections of the same term in the text is treated as a single node in the network. Secondly, the dependency is added to the network as an undirected edge between the two terms.

The procedure is described in Algorithm 6.1.

Algorithm 6.1 Create dependency network from text

Input: *text*: sequence of sentences
Output: *graph*: dependency network

```
1: nodes ⇐ ∅
2: edges ⇐ ∅
3: for all sentence in text do
4:    dependencies ⇐ STANFORDPARSER(sentence)
5:    for all head, dependent in dependencies do
6:       nodes ← nodes ∪ {head}
7:       nodes ← nodes ∪ {dependent}
8:       edges ← edges ∪ {(head, dependent)}
9:    end for
10: end for
11: return graph ⇐ ⟨nodes, edges⟩
```

6.2.2 Evaluation

In order to determine how well the basic dependency networks perform, we have performed some experiments. Since this is a new type of network, we do not make any assumptions about which centrality measure will perform best. The networks are therefore evaluated using all centrality measures. We also test whether the representation gains from using the additional information of the frequency of each dependency as edge weights. The evaluations are therefore repeated for both weighted and unweighted dependency networks. Stop-word removal is not done in the construction of these basic networks, thus all terms from the text are included as nodes.

6.2.3 Results

Figures 6.4 and 6.5 display the results for classification and retrieval, respectively, as bar charts.

It seems the choice of centrality measure has a lot more to say for performance than the use of edge weights does. There are considerable differences between the classification performance when different centrality measures are employed. Whether the use of edge weights is beneficial seems to depend on the centrality measure used. For some measures, weighted networks are better, for others they are not. The differences between weighted and unweighted are, however, for most measures considerably less than the differences between measures.

An interesting observation is that all the eigenvector-based measures perform best with unweighted edges. This trend is found for all the measures in this group in both evaluation tasks, and the difference is most distinct for eigenvector centrality and the HITS centralities. The preference for unweighted networks among these measures seems to indicate that, using the metaphor of recommendations among nodes, it does not matter how often a term *recommend* another, only that it does

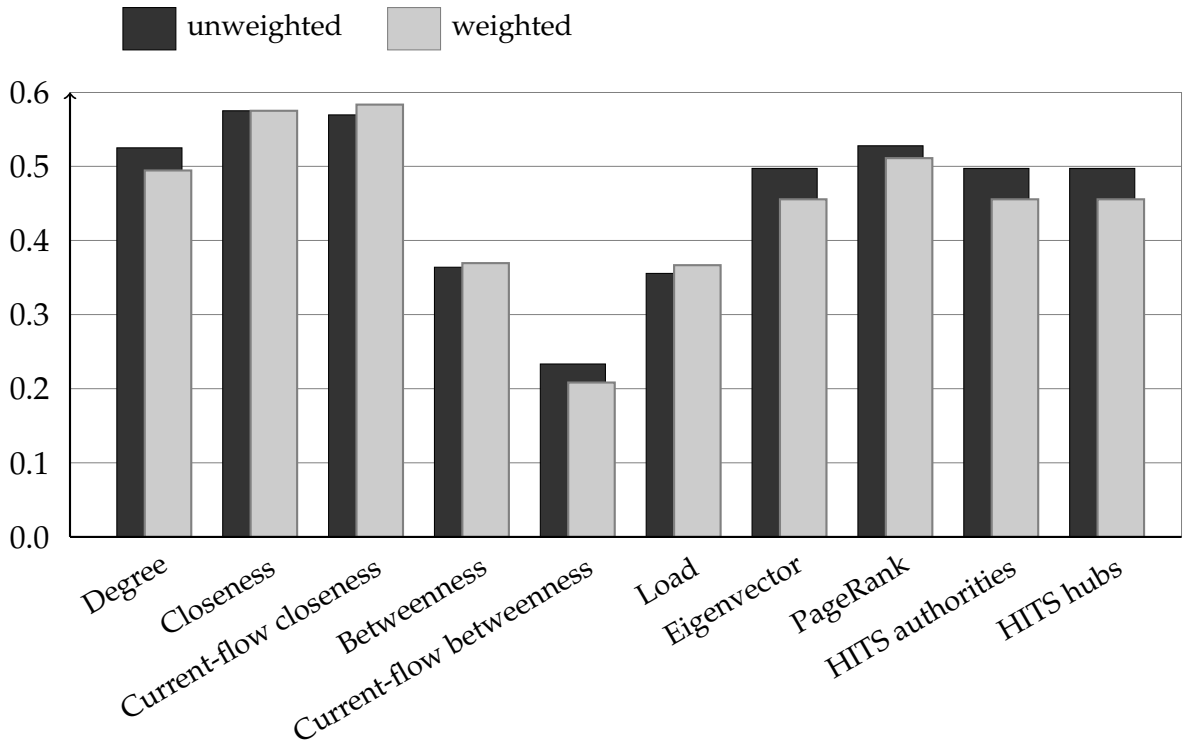


Figure 6.4: Classification results using weighted and unweighted networks with different centrality measures.

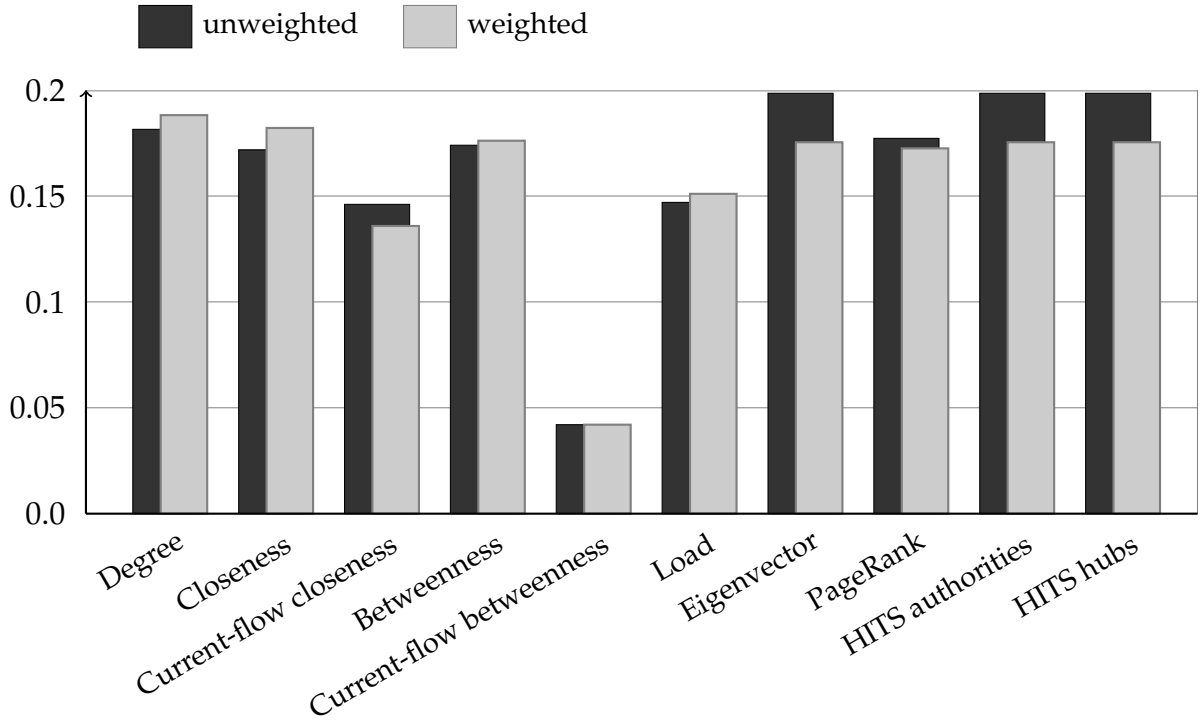


Figure 6.5: Retrieval results using weighted and unweighted networks with different centrality measures.

so. Recommendation in the context of dependency networks means that a term indicates another as important if they are connected by a dependency.

Some consistency between the performance of the various centrality measures can be seen in the two evaluations. The most notable common trait is the remarkably low performance of current-flow betweenness in both evaluation tasks. It is also interesting to see that closeness, both current-flow and regular, outperforms the corresponding betweenness measures in both evaluations. This is the opposite of what was the case for co-occurrence networks, where closeness generally constituted the weakest centrality measure group. This difference is interesting because it reveals that there might be considerable differences in the characteristics of the network types.

In order to find clues as to these differences, we look at the properties of the basic dependency networks network. Table 6.8 on page 86 lists some properties of the dependency networks created from our two datasets. We compare these results to the corresponding values for co-occurrence networks in Table 5.4 on page 63. The most notable differences are found in the connectivity of the networks. These are much as expected, considering how the networks are constructed. We see that the dependency networks have about the same orders, i.e. number of nodes, as the co-occurrence networks, but that they have a lot less edges. This has in turn led to a much lower average degree for the nodes. The characteristic path length has also increased, and the clustering coefficient is significantly lower. The dependency networks are most easily compared to the co-occurrence networks using sentences as contexts, since sentences is the textual unit from which dependencies are extracted. Whereas a co-occurrence network includes edges between all pairs of words occurring within the sentence, the dependency network relate only those words for which a formal dependency relation exists. Thus, the set of edges in a dependency network will necessarily be a subset of the edges in a co-occurrence network for the same document. This explains the reduction in edges within the dependency networks.

It is hard to say exactly what has affected the performance of the closeness compared to the betweenness centrality, but some hypotheses present themselves from the differences just described. One possibility is that the reduction in edges, and the resulting lengthening of paths, has made it easier to distinguish important terms based on how well connected they are to the rest of the network. If dependencies really are a subset of the most meaningful co-occurrences within sentences, then closeness by these edges may be a better measure of term importance. Another factor that might have had an impact is the drastic reduction in clustering coefficient. In a network, such as a co-occurrence network, with a high clustering coefficient, there will be some terms that belong to two or more cliques. Such words will score high in betweenness centrality. Since these words are less common and pronounced in dependency networks, due to the lower degree of clustering, this might be part of the reason for the low performance of betweenness.

6.2.4 Conclusion

The basic dependency network representations provide insights into the usefulness of this kind of representation. We have, based on the initial experiments, looked

at the performance of various centrality measures and compared weighted and unweighted edges. The next section will investigate various aspects of the representation in order to improve it.

The best centrality measures for the classification task are clearly closeness and current-flow closeness, regardless of the use of weights. Given that closeness and current-flow closeness performs about equally well, and that closeness is a far less computationally expensive measure, we select closeness for use with the dependency networks in classification. For retrieval, the HITS measures and eigenvector are the best performers when used on unweighted networks. Of the three, eigenvector centrality is the simplest measure and will for this reason be used with the representation for this task.

In classification the closeness centrality gain nothing from the addition of weights to the edges, and the use of weights seem to actually harm performance in retrieval. Hence, the final dependency network representation will be unweighted.

6.3 Improvements

Through the following sections, we will describe and evaluate a number of different ways in which the basic dependency network representation might be improved. We will adopt to the final network representation those features that improve its performance.

6.3.1 Removing Stop-Words

In the evaluation of the basic dependency networks above, stop-word removal was not done as a part of the network construction phase. Stop-words are common function words, such as *the*, *is*, and *at*. Stop-word removal was done as part of the preprocessing for the co-occurrence networks, where it lead to improvements in performance. Removing these words improved quality of the resulting representation because the words carried little meaning. We are interested in finding out whether stop-word removal will have a similar effect for dependency networks. Stop-word removal will in this case have an effect on the set of dependencies used in the networks, because some of the dependency types, such as *determiner* typically target words from this group.

To determine the usefulness of stop-word removal, we evaluated the representation with and without removing stop-words. The results are listed in Table 6.2.

Table 6.2: Evaluation of stop-word removal for dependency networks.

	Classification	**Retrieval**
Including stop-words	**0.5750**	**0.1985**
Excluding removed	0.5667	0.1852

We see that for dependency networks, the removal of stop-words actually harms the performance for both evaluation tasks. Our dependency network representation should therefore clearly not use stop-word removal as part of the construction process.

It is clear that stop-words are more valuable in dependency networks than in co-occurrence networks, or even in frequency-based representations such as TF-IDF vectors. The reason for this is not obvious, but a possible factor might be that stop-words act as hubs in the network. While they do not hold much semantic information, they link with a very high number of other words. That is, they act as a form of *function nodes*, equivalent to their role as function words in the text.

For example, the *the* node in the example in Figure 6.2 on page 73 link *pilot* with *touchdown*. This is just a single sentence, and the degree of such words in networks for complete documents will naturally be much higher. Thus, we hypothesize that by keeping the network better connected, the stop-words contribute to the overall performance.

This hypothesis can easily be tested by looking at the common hub words. Table 6.3 holds a lists of the 10 words most frequent among the top 10 hubs in both datasets with and without stop-word removal.

Table 6.3: Lists of common hub words with and without stop-words included in the representations.

With stop-words		**Without stop-words**	
TASA900	**AIR**	**TASA900**	**AIR**
all	four	becaus	four
not	**all**	onli	fatal
them	fatal	veri	signific
onli	voic	one	voic
then	signific	often	factual
becaus	veri	good	veri
they	partial	still	partial
each	**further**	also	profici
veri	privat	everi	tail
should	profici	might	follow

The lists on the right contain the most common hubs where stop-words have been removed, while the lists on the left show hubs when the stop-words are retained. We see that there are stop-words in both the lists on the left, although there are many more in TASA900 than in the AIR dataset. This means that at least some of the stop-words occupy the suspected highly interconnected position contemplated above.

The findings are consistent with those of Ferrer i Cancho et al. (2004), discussed in Section 3.4.2, who found a linear relationship between term-frequency and degree in dependency networks. Since function words are among the most frequent terms, this explains the strong presence of stop-words among the hubs.

For TASA900 the majority of the listed hubs are stop-words. The only exceptions are *onli*, *becaus* and *veri*, which are stemmed forms of *only*, *because* and *very*. These words should arguably also have been included in the list of stop-words.

Comparing the lists for AIR reveals only two stop-words: *all* and *further*. This shows that stop-words hold positions as hubs also here, but to a far lesser extent.

A likely reason for this is that while the TASA corpus deals with a wide range of topics, the AIR dataset focus entirely on the aviation incident domain. Since TASA is topically diverse, the stop-words, which themselves are topically neutral, have a wider range of terms to interact with. Many terms do not generally interact across topics, and thus stop-words may be needed to connect the different topics in the network.

6.3.2 Edge Directions

Adding directionality to the edges is another possibility for enhancement of the dependency networks. There are basically two alternatives to the undirected edges we have been using so far: *head-to-dependent* and *dependent-to-head*.

The use of directed edges obviously hold additional information that might prove useful in the representation. For example, given the `nsubj(decided, pilot)` relation in the above example, it is clear that it was the pilot that decided something, and not the other way around. However, with directed edges we also lose some information. With an edge directed from dependent to head, *pilot* $\rightarrow$ *decided*, there would be only an implicit relation from *decided* back to *head*. For our centrality measures such implicit relations are not taken into account, and the information is thus lost in the final representation.

To test the practical usefulness of edge direction, we have evaluated the representation with both edge directions, and compare the results to the undirected version. Table 6.4 holds the results of this evaluation.

Table 6.4: Evaluation of edge direction for dependency networks.

	Classification	**Retrieval**
Head-to-dependent	0.3638	0.1226
Dependent-to-head	0.3638	0.1512
Undirected	**0.5750**	**0.1985**

It is evident that the use of directed edges is harmful to the performance. For classification we see an identical and substantial drop in accuracy for the two directions, and for retrieval there is also a considerable performance loss, especially for the head-to-dependent direction. These results suggests that the direction information cannot be utilized by the centrality measures, and should therefore not be used in the final representation.

6.3.3 Dependency Types

The dependency networks explored so far utilize all available dependency types from the Stanford Parser. This is a diverse set of 52 relations. Some, like *determiner*, hold information related closely to the syntactic structure of a sentence. Others hold information more concerned with relations or properties of semantic objects. An example of this last type is the *amod* (adjectival modifier) dependency. It describes a modification of the meaning of a noun phrase. Some examples from the AIR dataset

are `amod(rain, light)`, `amod(factors, physiological)` and `amod(difficulties, navigational)`.

We have, in order to determine the contribution of each of the dependency types, performed a simple experiment. By evaluating the networks 52 times, each time leaving out one type of dependencies, we measure how much performance is gained or lost by excluding that type. The aim of the experiment is to find the best possible set of dependency types to use in the network, enhancing the performance in the process.

The results of the experiment are listed in Table 6.5. The dependency types are listed in order of decreasing contribution to the evaluation performance. For each dependency type, the performance achieved without the dependency included in the networks is listed. The difference between this performance and that of the the networks using the complete set of dependencies is listed as well. A negative change means that the removal of that dependency harms overall performance, and that the it contributes positively to the total representation. We see that for both evaluation measures, there are some types which removal harm the performance. Other measures either have no influence over the outcome, with differences of 0.0, or benefit the performance by being removed.

It is tempting to simply remove all dependency types that do not show a direct contribution towards a higher performance. There are, however, two problems with this.

First, there is no clear distinction between the good and the bad dependency types. For example, the most important type for classification, *advmod*, has a negative effect in retrieval. Many such examples exist, showing that the results of this evaluation serve only to give an indication of what worked well under the current conditions, i.e. our datasets and tasks. The differences between the two tasks may be caused by the different natures in the evaluation methods, or be characteristic to the two datasets used in the evaluation. Hopefully, the common ground between the results will be sufficient to tell us something more general.

Second, it is important to note that this evaluation does not reveal anything about the interplay between the different dependency types. The dependency networks are complex structures, and it is reasonable to assume that the performance of the whole is not simply the sum of their parts. We evaluated the contribution, or lack thereof, from dependencies separately. From this, it is impossible to say with confidence what will happen when several types are removed at once. It might, for example, be that the removal of every dependency type showing a difference of 0.0 could harm the performance, even though none of the involved types cause harm or benefit individually. This simply because removal of many edges from the representation would make the network much more sparse.

In order to understand this better, and to determine the best set of dependency types to use in the representation, another evaluation has been done. We evaluated representations constructed with different subsets of dependency types. We test four strategies defining which edges should be removed form the representation:

Defensive: The most cautious strategy calls for the removal of only those dependency types that proved directly harmful in both the evaluations. This concerns only a small set of the dependencies: *agent*, *advcl* and *parataxis*.

Table 6.5: Evaluation of the contribution from each dependency type.

(a) Classification

Type	Performance	Diff
advmod	0.5306	-0.0444
prep	0.5361	-0.0389
ccomp	0.5389	-0.0361
nn	0.5417	-0.0333
det	0.5444	-0.0306
dobj	0.5444	-0.0306
cop	0.5500	-0.0250
amod	0.5500	-0.0250
nsubj	0.5528	-0.0222
xsubj	0.5556	-0.0194
complm	0.5583	-0.0167
prt	0.5611	-0.0139
conj	0.5611	-0.0139
dep	0.5639	-0.0111
preconj	0.5639	-0.0111
rcmod	0.5639	-0.0111
num	0.5639	-0.0111
appos	0.5639	-0.0111
iobj	0.5667	-0.0083
purpcl	0.5667	-0.0083
mwe	0.5667	-0.0083
tmod	0.5667	-0.0083
pcomp	0.5667	-0.0083
prepc	0.5667	-0.0083
auxpass	0.5694	-0.0056
nsubjpass	0.5694	-0.0056
expl	0.5694	-0.0056
infmod	0.5694	-0.0056
partmod	0.5694	-0.0056
quantmod	0.5694	-0.0056
acomp	0.5722	-0.0028
xcomp	0.5722	-0.0028
rel	0.5722	-0.0028
predet	0.5722	-0.0028
neg	0.5722	-0.0028
npadvmod	0.5722	-0.0028
poss	0.5722	-0.0028
number	0.5750	0.0000
attr	0.5750	0.0000
aux	0.5750	0.0000
possessive	0.5750	0.0000
abbrev	0.5750	0.0000
punct	0.5750	0.0000
ref	0.5750	0.0000
pobj	0.5750	0.0000
mark	0.5750	0.0000
csubj	0.5750	0.0000
csubjpass	0.5750	0.0000
agent	0.5778	0.0028
parataxis	0.5778	0.0028
cc	0.5778	0.0028
advcl	0.5861	0.0111

(b) Retrieval

Type	Performance	Diff
det	0.1902	-0.0083
conj	0.1941	-0.0044
prep	0.1948	-0.0037
amod	0.1961	-0.0024
num	0.1965	-0.0020
appos	0.1969	-0.0016
partmod	0.1969	-0.0016
nn	0.1970	-0.0015
auxpass	0.1972	-0.0013
cop	0.1975	-0.0010
xsubj	0.1977	-0.0008
infmod	0.1978	-0.0007
mark	0.1982	-0.0003
acomp	0.1983	-0.0002
complm	0.1983	-0.0002
rel	0.1984	-0.0001
quantmod	0.1984	-0.0001
iobj	0.1985	0.0000
attr	0.1985	0.0000
csubj	0.1985	0.0000
csubjpass	0.1985	0.0000
cc	0.1985	0.0000
mwe	0.1985	0.0000
npadvmod	0.1985	0.0000
number	0.1985	0.0000
possessive	0.1985	0.0000
prt	0.1985	0.0000
punct	0.1985	0.0000
ref	0.1985	0.0000
pcomp	0.1985	0.0000
expl	0.1985	0.0000
purpcl	0.1986	0.0001
parataxis	0.1986	0.0001
tmod	0.1986	0.0001
poss	0.1987	0.0002
rcmod	0.1988	0.0003
agent	0.1989	0.0004
pobj	0.1989	0.0004
abbrev	0.1989	0.0004
dep	0.1990	0.0005
predet	0.1990	0.0005
prepc	0.1992	0.0007
xcomp	0.1993	0.0008
preconj	0.1995	0.0010
advmod	0.1995	0.0010
aux	0.2002	0.0017
nsubjpass	0.2005	0.0020
dobj	0.2007	0.0022
advcl	0.2008	0.0023
neg	0.2008	0.0023
nsubj	0.2011	0.0026
ccomp	0.2012	0.0027

Aggressive: This strategy takes a more dramatic approach by removing all dependency types that are shown to be harmful in either of the evaluations. The types removed include the three from the defensive strategy, and an additional 19 more: *dep*, *aux*, *ccomp*, *xcomp*, *dobj*, *pobj*, *nsubj*, *nsubjpass*, *cc*, *abbrev*, *purpcl*, *predet*, *preconj*, *advmod*, *neg*, *rcmod*, *tmod*, *poss*, and *prepc*.

Compromise 1: The compromise between the above strategies calls for the removal of a dependency type if it proved harmful in one of the evaluations, unless it is seen to make a significant contribution in the other. As thresholds for significant contribution, we use 0.01 for classification and 0.001 for retrieval. The following dependencies are affected in addition to those from the defensive strategy: *aux*, *xcomp*, *pobj*, *nsubjpass*, *cc*, *abbrev*, *purpcl*, *predet*, *neg*, *tmod*, *poss*, and *prepc*.

Compromise 2: This strategy uses the same criterion as the first compromise, but additionally removes any dependency with contribution of zero for both evaluation tasks. This calls for the removal of *attr*, *csubj*, *csubjpass*, *number*, *possessive*, *punct*, and *ref* in addition to those mentioned for Compromise 1 above.

The performance of the different strategies is listed in Table 6.6. It is clear that changes to the set of dependency types impact the performance of the representation. The *defensive* strategy is best in both evaluations, providing an improvement over the baseline using all dependency types.

In retrieval, all removal strategies perform better than baseline. We see, however, a drastic decline in performance for the more aggressive strategies for classification. This suggests that TASA900 or the classification task depends on a wider selection of dependency types than retrieval on the AIR dataset does. It is in any case clear that some of the dependencies identified by the Stanford Parser is harmful to the representation, namely those removed by the defensive strategy: *agent*, *advcl* and *parataxis*. We consequently remove these from our final dependency network representation.

Table 6.6: Evaluation of dependency type removal strategies.

Strategy	Classification	Retrieval
Defensive	**0.5889**	**0.2020**
Aggressive	0.5056	0.2000
Compromise 1	0.5639	0.2009
Compromise 2	0.5583	0.2004
No removal	*0.5750*	*0.1985*

6.4 TC-ICC Weighting

Section 4.2.2 introduced two measures of term importance, TC and TC-ICC. These are used to build vector representations for documents, based on their text networks. In this chapter, TC has been used for the dependency network evaluations

up to this point. Here we evaluate the performance of TC-ICC with dependency networks.

Table 6.7 lists the results for the evaluation tasks. For comparison, the best results obtained using TC with the same measures were 0.5889 for classification and 0.2020 for retrieval. Like for the co-occurrence networks described in Section 5.3, the current-flow betweenness centrality measure proved too memory demanding also here. Since this makes the measure impractical for use with TC-ICC, and because of its extremely poor performance with TC (as seen in Figures 6.4 and 6.5 on page 77), we decided not to pursue the issue.

Table 6.7: Performance of TC-ICC with dependency networks.

Representation	Classification	Retrieval
Degree	0.5028	0.1914
Closeness	**0.5056**	0.1732
Current-flow Closeness	0.5028	0.1753
Betweenness	0.3806	0.1480
Current-flow Betweenness	n/a	n/a
Load	0.3778	0.1479
Eigenvector	0.4667	**0.2048**
PageRank	0.4833	0.1772
HITS-authorities	0.4611	0.2025
HITS-hubs	0.4611	0.2025

It is clear that TC-ICC performs far worse than TC for the classification task. The best measure here, closeness centrality, achieves classification accuracy of only 0.5056, which is far from the best performance reached using TC. The second best measure is the degree centrality, which performs at almost on the same level as closeness. All other measures perform considerably worse than this. We thus conclude that the introduction of information bout the overall corpus centrality is harmful to the performance for classification.

The TC-ICC measure performs better on the retrieval task. The best measure when using TC, the eigenvector centrality, is again best. Its score is 0.2048, which is slightly higher than the corresponding value for TC. Also the HITS measures perform on the level with the best TC result. PageRank, however, which is measure in many ways similar to these, performs rather poorly. It would seem that the use of corpus level information does not decrease retrieval performance, but neither is the performance greatly increased.

Not surprisingly, we see a high correlation between the performances using TC and TC-ICC for the various measures. Those centrality measures that achieve high scores with TC, also performs best when TC-ICC is used. This is the case in both classification and retrieval. This indicates that although our choice of centrality measure was based on evaluations done with TC, the results apply to TC-ICC as well.

To summarize, we see the same main results here as with the use of TC-ICC on co-occurrence networks presented in Section 5.3. TC-ICC seems to perform worse than TC in classification, but on the same level or better for retrieval. Also, the results indicate that the choice of centrality measure does not depend on whether

TC or TC-ICC is used.

6.5 Network Properties

This section describes some properties of the dependency networks. The discussion should be seen in light of Section 3.4.2, where previous research on this type of networks was presented. Unlike the networks described there we do not create a single network over the entire corpus. Rather, we investigate the properties of the networks representing each document in our datasets.

Table 6.8 lists central network properties for networks created from TASA900 and AIR. For each value, we list the mean for all document networks in the datasett, as well as the standard deviation.

We have also constructed random networks, according to the Erdős-Rényi model, as a basis for comparison with the dependency networks. These networks have the same number of nodes as the average over each of the two datasets, and approximately the same number of edges. Details about the random networks are also included in Table 6.8.

Table 6.8: Some properties of the dependency networks.

	TASA900		**AIR**		**Random**	
	Mean	St.dev.	Mean	St.dev.	TASA900	St.dev.
N	170.076	23.816	533.826	172.171	170	533
$\langle k \rangle$	20.305	1.098	27.634	3.624	20.294	27.448
l	2.194	0.046	2.153	0.029	2.463	2.678
C	0.569	0.024	0.572	0.012	0.114	0.050

6.5.1 Comparison with Co-occurrence

It is reasonable to first compare the properties of the dependency networks to those found for the co-occurrence networks in Section 5.4. We see that the dependency networks are larger for both datasets. This is because stop-words are removed prior to constructing the co-occurrence networks, while they are included in the dependency networks. Further, the mean degree $\langle k \rangle$ of nodes are lower both for TASA900 and AIR. The same is true for the clustering coefficients, C. This probably indicate that while the number of relations extracted with the dependency parser is lower than that of full sentence contexts, it is higher than that of two-word context windows. The increase in average degree $\langle k \rangle$ for AIR is likely because the stop-words, as we saw in Section 6.3.1, are hub words and thus have a major contribution to the connectedness of the networks.

6.5.2 Small World Property

By comparing the values of C and l to the corresponding values for the random networks, C_{rand} and l_{rand}, we see a similar pattern to that found for the co-occurrence

networks. The fact that $C > C_{\text{rand}}$ and $l \sim l_{\text{rand}}$ shows that both datasets form small-world networks.

We note that the value of C_{rand} is much higher for TASA900 than for AIR. The TASA900 networks have a much higher $\langle k \rangle$ relative to their number of nodes N. This is caused by the fact that while the average number of nodes in AIR networks is more than 3.1 times that of TASA900, the average $\langle k \rangle$ is only 1.35 times higher. We believe this to be caused by the role of the hubs in the network, which typically are stop-words that are part of both dataset. Since these words accounts for the majority of connections in the network, the higher number of nodes in AIR does not entail a proportionate increase in $\langle k \rangle$, which leads to the difference in expected connectedness.

6.5.3 Scale-Free Property

Like the co-occurrence networks, the dependency networks are not scale-free. This result is in stark contrast to results from previous research, as described in Section 3.4.2. The degree distributions do share some properties consistent with power-law distributions, such as the presence of hubs in the network, but do not otherwise follow this model. The same remarks made regarding this in Section 5.4.3 applies here as well.

Ferrer i Cancho et al. (2004) do not describe the exact method used to determine their power-laws, but use the linear log-log plot of the distribution as supporting evidence. Our methods used to rule out the power-law hypothesis, described by Clauset et al. (2009), are more rigid. Linearity in the plot is a necessary but not sufficient requirement to identify a distribution governed by a power-law. A statistical goodness-of-fit measure is also needed to determine whether a power-laws fitted to the data actually is a plausible model. It would be interesting to see results of the methods described by Clauset et al. on the networks created by Ferrer i Cancho et al. We refer to Appendix A for the details about power-laws and how we searched for them in the network data.

Table 6.9: Summary of the dependency network representation.

Property	Description
Nodes	Terms used in the text
Edges	Syntactic dependencies between terms
Contexts	Sentences
Edge weights	No
Edge directions	Undirected
Centrality (classification)	Closeness centrality
Centrality (retrieval)	Eigenvector centrality
Text processing	Stemming, token filtering, case folding
Stop-word removal	No
Dependency types	All except *agent*, *advcl* and *parataxis*

6.6 Summary

As an initial approach to a dependency network representation, a basic dependency network was described and evaluated. This representation uses undirected and unweighted edges, performs no stop-word removal, and includes all available dependency types.

Building on this representation, both stop-word removal and edge directions were tested. In both cases, it turned out that changing the basic dependency network representation did harm rather than good, and both measures were dismissed. Different subsets of dependency types were then evaluated, and as a result of this some dependencies were removed. The main aspects of the final dependency network representation is summarized in Table 6.9 on the preceding page.

Chapter 7

Experiments

In this chapter we compare the graph-based representations to each other and to the baseline methods that use TF and TF-IDF based weigthing. The intention is to find out whether the new representations are able to outperform the commonly used BoW representations, and to determine which of them facilitate the best measurement of text similarity.

We start the chapter by exploring the terms identified as most central or important by the different representations. This is done in the first section, by means of a case study of one of the aviation incident reports form the AIR dataset. Section 7.2 next describes the experiments conducted to evaluate the performance of the representations. The evaluation tasks are the same as those used in the previous chapters, although new datasets are introduced for each task. The results are subsequently described and discussed in Section 7.3. Finally, Section 7.4 ends the chapter with brief summary of the experiments.

7.1 Study of Central Terms

In order to better understand how the representations work, we are interested to find out which terms in the document collection are determined to be important. To do a thorough analysis of this would be a comprehensive task. Instead, we have opted to do a tentative evaluation, based on a case study of one of the incident reports from the AIR dataset. This will give us some ideas about the nature of the central terms, but is naturally too limited for us to say anything conclusive. In the absence of a domain expert, the evaluation of the terms extracted by the different representations is done manually by us, based on our understanding of the report.

The report selected for the study has identification number `A05A0059`, and is available online from the TSA[1]. We use the entire report in our study. Its summary is presented below.

> *A de Havilland DHC-8-100 (Dash 8) aircraft (registration C-GZKH, serial number 117) operated by Provincial Airlines Limited was a passenger revenue flight from St. John's to Deer Lake, Newfoundland and Labrador, with 36 passengers and 3 crew on board. During the climb-out*

[1] `http://www.tsb.gc.ca/eng/rapports-reports/aviation/2005/a05a0059/a05a0059.asp`

> *from St. John's, the indicated airspeed gradually decreased to the point that the aircraft entered an aerodynamic stall. The aircraft descended rapidly, out of control, losing 4200 feet before recovery was effected approximately 40 seconds later. The incident occurred during daylight hours in instrument meteorological conditions. There were no injuries and the aircraft was not damaged.*

The report describes the factual circumstances of the incident, such as weather, events, aircraft, and crew. It also presents TSA's analysis of the situation, and their interpretations of the causes and contributing factors. Risks related to the occurrence and safety actions taken as a consequence of the investigation is also described. The report concludes that the cause of the loss of control was human error. The crew failed to detect signals of the stall of the aircraft until it was too late, and failed to activate the pneumatic de-icing equipment when they should have. This might be attributed to a limited training in stall recognition and recovery. No technical malfunctions were found that could lead to the occurrence.

By constructing text networks of the report, we identified the 20 highest weighted terms using TC and TC-ICC. The most important terms according to TF and TF-IDF were also found. Table 7.1 on the next page lists the central terms, in order of decreasing importance, for the various representations.

The local measures, term frequency (TF) and term centrality (TC) from the two graph types, are listed first. The measures including global information, i.e. TC-ICC and TF-IDF, are listed thereafter. The terms outlined in bold font for the TC and TC-ICC representations are those that do not also appear in the corresponding lists of terms for TF or TF-IDF, respectively.

Stemming was done as part of the text preprocessing in all representations. The stemmer used, the PorterStemmer (Porter, 1980), is far from infallible. This explains why some of the words look strange or incomplete.

Starting with the local measures, we see that there is a high degree of overlap between terms selected in different representations. Looking at the relevance of the terms with respect to the report, no representation is therefore remarkably better than any of the others. The terms identified as relevant by TF are, perhaps not surprisingly, mostly relevant. Terms exclusive to the graph-based representations vary somewhat in relevance, but are also mainly relevant to the contents of the report. A problem with many terms found by both TF and TC is that they although relevant also tend to be general. Examples include *aircraft* or *flight*. Terms directly relevant to the cause of the incident are, however, also identified, such as *de-ice* and *airspeed*.

Of the graph-based measures, TC on co-occurrence networks is the one most similar to TF, introducing only 4 different terms. This is likely because degree-centrality, which is used with this network representation, is similar to the frequency measure. The difference is that while TF rates number of occurrences, unweighted degree measures how many different terms a term occurs close to. Thus, frequent terms with varied contexts score higher here.

TC on dependency networks also have many of the same terms as TF. We observe two differences here: Firstly, as a result of not removing stop-words, *the* appears as the most central term. Secondly, there are more verbs among the dependency terms. This can be explained by the fact that verbs often are syntactically central

Table 7.1: Central terms from the different representations. Terms highlighted in bold font are those included by graph-based but not by the corresponding frequency-based representation.

Representation	Most central terms
TF	aircraft, flight, ice, stall, control, crew, data, mode, select, pilot, oper, airspe, train, de-ic, climb, pneumat, captain, recoveri, safeti, use
Co-occurrence TC	stall, ice, mode, aircraft, flight, select, crew, pilot, oper, **procedur**, pneumat, control, **boot**, climb, airspe, train, de-ic, **error**, data, **system**
Dependency TC	**the**, select, stall, aircraft, flight, mode, control, airspe, **system**, oper, climb, use, ice, crew, **inform**, **monitor**, **activ**, **warn**, **sop**, **indic**
TF-IDF	edg, unev, dispel, signific, ice, abil, follow, captain, c-gzkh, accur, transduc, inlet, articl, rudder, program, becam, larg, digit, risk, none
Co-occurrence TC-ICC	**stall**, ice, **mode**, **select**, **pneumat**, **airspe**, **de-ic**, **error**, **climb**, **data**, captain, **recoveri**, **engag**, **monitor**, **sop**, **boot**, **ia**, **afc**, **train**, **turbul**
Dependency TC-ICC	**the**, **select**, **stall**, **aircraft**, **mode**, **flight**, **airspe**, **control**, **system**, **climb**, **oper**, **use**, ice, **crew**, **inform**, **monitor**, **activ**, **warn**, **sop**, **detect**

in sentences, and hence often form the root of the extracted dependency tree. An example of this is seen in the sample sentence in Figure 6.2 on page 73.

Moving on to the global measures, the small overlap between TC-ICC and TF-IDF is prominent. Only two terms from those listed by TF-IDF appear for TC-ICC as well. TF-IDF very clearly shows a preference for terms with low frequencies among the other reports. TC-ICC, on the other hand, actually includes more terms in common with TF or TC, than with TF-IDF. Again we also see *the* as the most central term, and the same bias towards identifying verbs as important terms for the dependency representation. This seems to indicate that the ICC portion of the TC-ICC measure is not given enough weight to really make a difference for the most central terms. A contributing factor here might be that in TC-ICC a low value for TC can in effect cancel out a high value for ICC, whereas in TF-IDF a high value for IDF in some cases lead to the term being considered important despite not actually having a high TF value. A good example is the term *c-gzkh* which is one of the terms identified as important by TF-IDF. *c-gzkh* is actually the registration number of the airplane and is only mentioned in this particular report, giving it the highest possible IDF score. The term is mentioned only once in the report, as part of the summary, which makes the TF value the lowest possible. TF-IDF still identifies it as important, because it is so rare. The term is not scored highly by the TC-ICC measures, however. This is because although the ICC score is high, the TC value (as the numerator in Equation (4.1) on page 41) is almost zero. Conversely, a term with a low overall ICC value, such as *stall*, might be considered important by TC-ICC

because it has a very high TC value.

To summarize the results of the case study, we find that the TC measures favours many of the same terms as TF. TC-ICC differs from TF-IDF in that very rare or non-central terms in the corpus are not rated highly unless they also have a high TC value. Of the two graph-based representations, the co-occurrence networks are most similar to term-frequency. The dependency network representations, for both TC and TC-ICC, seem to favour verbs as the most central terms.

7.2 Comparison Experiments

In Section 4.3 we presented two evaluation methods based on document similarity, which have been used throughout Chapters 5 and 6 to measure the performance of our representations. The tasks, *document classification* and *case retrieval*, are reused as experiments here.

We introduce new datasets for the experiments performed in this chapter. This is done because TASA900 and AIR were used while developing the representations. By using fresh datasets, we ensure that the representations do not perform better because they are fitted especially towards a specific set of data, and we can thus better make unbiased assessments of the representations.

7.2.1 Baselines

Two common representations based on word frequencies will be used as baseline in the experiments. The two, *term frequency* (TF) and *term frequency-inverse document frequency* (TF-IDF), were introduced in Section 2.2.4, and we refer to this description for the details. The main difference between these measures and our graph-based representations is that TF and TF-IDF are pure bag-of-words models. This means that they completely disregards the order of words in the documents.

Although relatively simple measures, TF and TF-IDF generally perform reasonably well, and are widely used in both the field of IR and in TCBR. We use them as baselines here for this reason, and because we want to discover how well our representations perform compared to bag-of-word representations based solely on frequency.

7.2.2 Experiment 1: Classification

The first experiment evaluates the representations based on document classification. The *k*-NN classifier performs text classification using document similarity. Thus, representations better able to facilitate document similarity measurement will perform better in the experiment. We refer to Section 4.3.1 for the full description of the evaluation measure. New here, compared to the initial evaluations, is the use of the Reuters dataset.

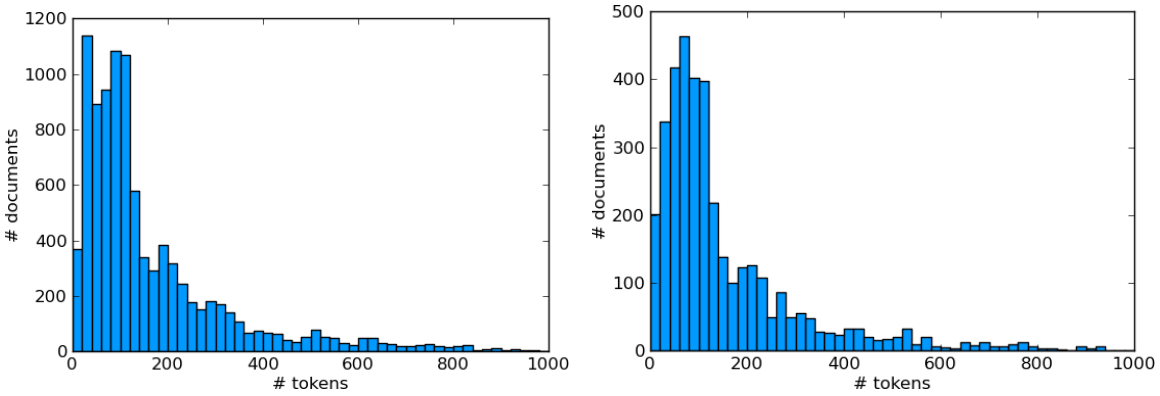
Reuters Dataset

Reuters-21578 is a collection of news articles that appeared on the Reuters newswire in 1987. It is widely used for text classification tasks, and has been manually la-

belled by Reuters personnel. The set consists of 21578 documents, some of which are unlabelled and some labelled with one or more of 672 different categories. The categories are divided into five different classes: *exchanges*, *orgs*, *people*, *places*, and *topics*.

The documents are separated into training and testing sets according to the ModApté split, which is described in the README-file accompanying the distribution of the dataset. ModApté uses categories from the *topics* class, consists of documents from 90 different categories. These are the categories which include at least 1 training document and 1 test document. There are a total of 9598 training documents and 3744 testing documents in the collection.

The dataset is available online[2] in its original form, as a set of files in the SGML format. Our copy was downloaded from a version maintained[3] by Alessandro Moschitti at the University of Trento, who have done a great job of preprocessing the data. His versions are provided in a format structured as a hierarchy of files and directories, where each category corresponds to a separate directory. We chose to use this versions of the datasets because this format is well adapted to our implementation, and therefore easier to use.



(a) Training (b) Test

Figure 7.1: Distribution of document lengths in the Reuters dataset.

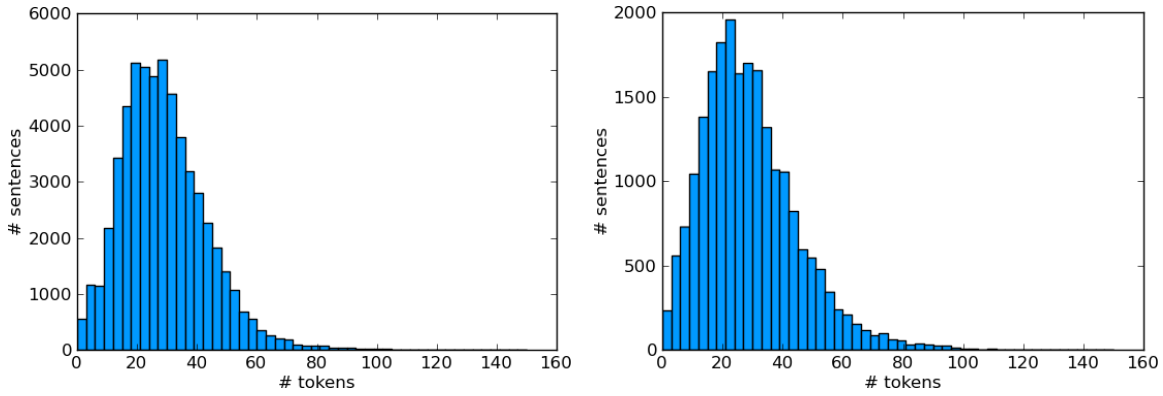
Figure 7.1 shows the distributions of document lengths in the training and testing portions of the dataset. Parts of the tails have been cut to make the histograms more readable. The highest document lengths are 1409 and 1804 tokens for *training* and *test*, respectively, but such long documents are rare. In both cases, 98.5% of documents have lengths shorter than 1000 tokens.

In Figure 7.2, similar distributions are shown for the sentence lengths. These distributions also have a long tail, similar to the document lengths. For *training*, only 45 out of 56893 sentences are longer than 150 tokens, with the longest being 260 tokens long. The *test* data consists of 21924 sentences, the longest of which 55 is longer than 150 tokens, and the longest is 428.

Compared to TASA900 (presented in Section 4.3.2) the distribution of document lengths is very different. TASA900 had a lower bound at about 250 words, and most documents consisted of between this and approximately 400 words. Reuters,

[2] `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

[3] `http://disi.unitn.it/moschitti/corpora.htm`



(a) Training (b) Test

Figure 7.2: Distribution of sentence lengths in the Reuters dataset.

in comparison, consist of many shorter documents, with the majority having less than 200 words. This is a natural considering the nature of the Reuters collection, namely brief news bulletins. A consequence of the shorter document lengths is that the text networks necessarily are smaller as well. The sentence length distributions are similar for TASA900 and Reuters, although the curve is shifted a bit towards longer sentences for Reuters. The distributions are very similar for the *training* and *test* parts of the dataset. Thus, text networks based on Reuters will tend to be smaller and more connected than those based on TASA900.

7.2.3 Experiment 2: Retrieval

The retrieval task was introduced in Section 4.3.3, and used to evaluate aspects of the representations throughout Chapters 5 and 6. This experiment is based on the same task but use a new dataset — a collection of *marine incident reports*.

To reiterate its key aspects, the case retrieval evaluation is an unsupervised evaluation task based on the CBR retrieval phase. The documents are split into two-part cases, one part a problem description and the other the solution. Retrieval is done based solely on the problem description parts, and performance is determined based on the similarity between the solutions of the query to the solutions of the retrieved cases.

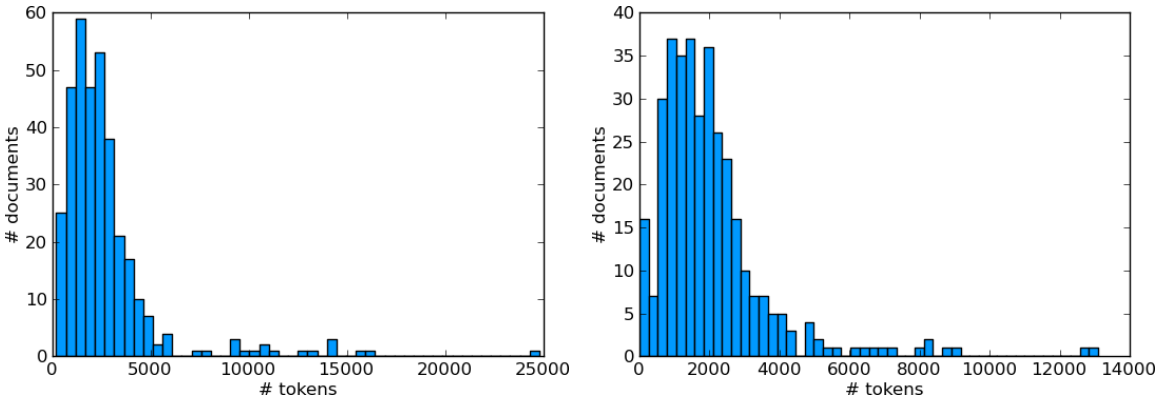
The Marine Incident Reports

The dataset used in the second experiment is similar to the AIR dataset presented in Section 4.3.4. AIR is based on a set of incident reports written by the Transportation Safety Board of Canada (TSB) from the domain of aviation. The TSB have also released a number of incident reports from the domain of marine transportation. These Marine Incident Reports (MIR) constitute our dataset for the retrieval evaluation in this chapter. Like the reports from the AIR dataset, the marine related reports are also available online[4].

[4] `http://www.tsb.gc.ca/eng/rapports-reports/marine/`

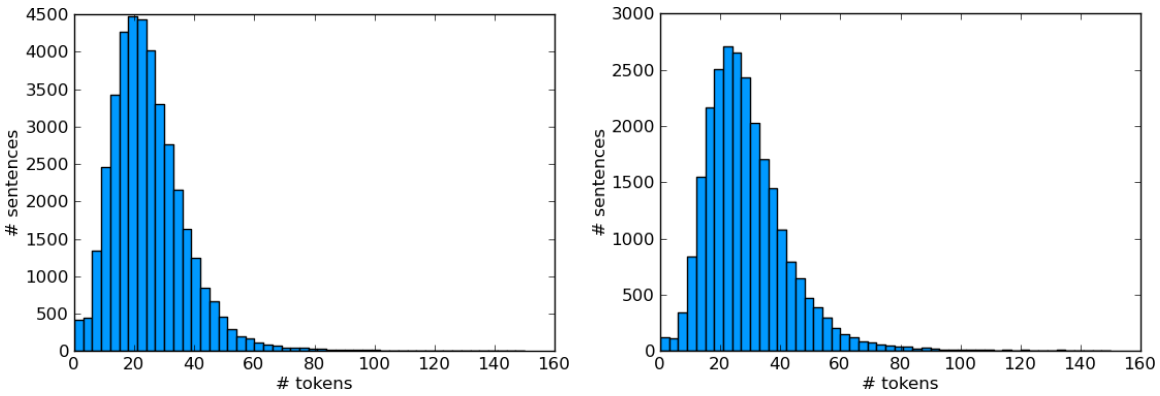
The MIR dataset is smaller than AIR, composed of only 337 documents. We used the same process as described in Section 4.3.4 to divide the reports into cases composed of problem description and solution parts.

Figure 7.3 presents the distributions of document lengths in the problem description and the solution parts of the dataset. Corresponding distributions for sentence lengths are shown in Figure 7.4.



(a) Problem descriptions (b) Solutions

Figure 7.3: Distribution of document lengths in the MIR dataset.



(a) Problem descriptions (b) Solutions

Figure 7.4: Distribution of sentence lengths in the MIR dataset.

We observed that the distributions are very similar between the problem descriptions and the solutions. In both cases, the majority of documents contain less than 5000 words, with some extreme exceptions constituting the long tails. The largest problem description has 24782 words, and the corresponding solution 13099 words. Similar long tails are seen in the sentence length distributions, although these have been cut short in the figure in order to improve readability. Only 18 of 39706 problem description sentences are longer than 150 terms, of which the longest is 460. For the solutions, 17 out of 25328 sentences are longer than 150 terms, and the longest is 264.

Compared to AIR, we see that the statistics for MIR are very similar. The main difference is that the reports in the MIR dataset typically are a bit longer than those

in AIR, although the change is not very large. Consequently, we can expect text networks based on the two datasets to be similar in size and connectivity.

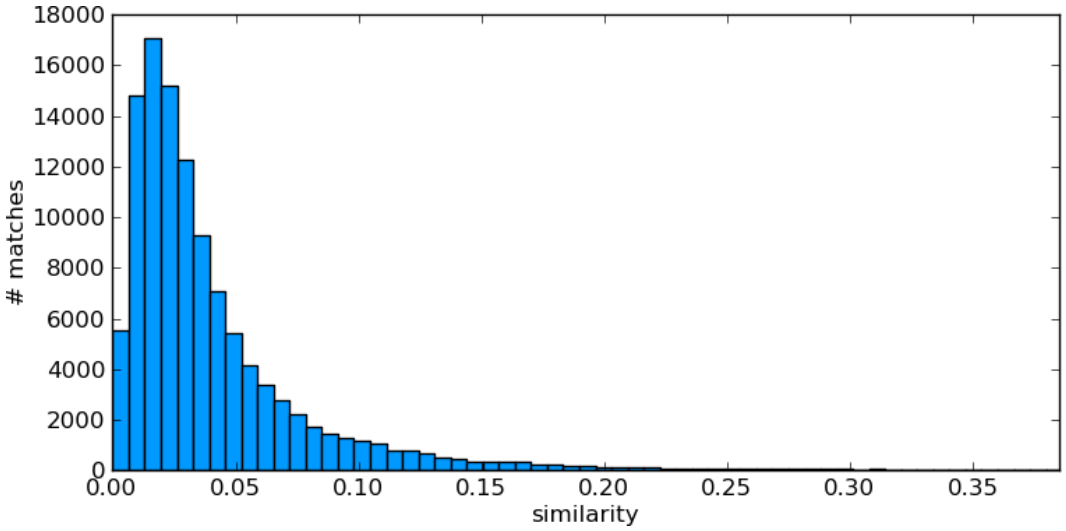


Figure 7.5: Distribution of *solution similarities* in the MIR dataset.

Also the distribution of solution similarities is similar to that of AIR. Figure 7.5 displays the distribution of similarities between all pairs of case solutions in MIR. These similarities are defined in terms of cosine between TF-IDF document vectors, as described in Section 4.3.3. The corresponding distribution for AIR is found in Figure 4.7 on page 47. By comparing the two figures, we see that the distributions are almost identical. If anything, the MIR distribution is skewed marginally towards lower similarity values. Based on this, we can expect retrieval results similar to those performed on AIR.

Of the MIR solution pairs, 99% have a similarity value below 0.23. Thus, to get a retrieval performance above this value, the retrieved cases must on average be among the one percent best solutions.

7.3 Results

The results from the experiments are listed in Table 7.2 on the facing page, and illustrated as a bar chart in Figure 7.6. For comparison, Table 7.3 on the next page also summarize the performance of various representations in their initial evaluations in Chapters 5 and 6. These results are illustrated visually in Figure 7.7.

The first thing to note is that in both comparison experiments, as well as for the preliminary evaluations, graph-based representations outperform the frequency-based representations. For classification, there are in both cases a marked difference of a few percent. In the retrieval experiment, the difference in performance is also notable. In the classification experiments, both for the comparison experiment and for the preliminary evaluations, there is a clear tendency that local measures perform better than the global. In every case, the local measures (TF and TC) perform better than the corresponding global ones (TF-IDF and TC-ICC, respectively). The difference is most pronounced for the dependency network representation on

Table 7.2: Results from the comparison experiments.

	Representation	Classification (Reuters)	Retrieval (MIR)
local	TF	0.6693	0.2243
	Co-occurrence (TC)	**0.6880**	0.2205
	Dependency (TC)	0.6827	0.2094
global	TF-IDF	0.6375	0.2392
	Co-occurrence (TC-ICC)	0.6875	0.2472
	Dependency (TC-ICC)	0.6763	**0.2555**

Table 7.3: Results from the preliminary evaluations throughout Chapters 5 and 6.

	Representation	Classification (TASA900)	Retrieval (AIR)
local	TF	0.5678	0.2240
	Co-occurrence (TC)	0.5750	0.2239
	Dependency (TC)	**0.5889**	0.2020
global	TF-IDF	0.5455	0.2459
	Co-occurrence (TC-ICC)	0.5333	**0.2559**
	Dependency (TC-ICC)	0.5056	0.2048

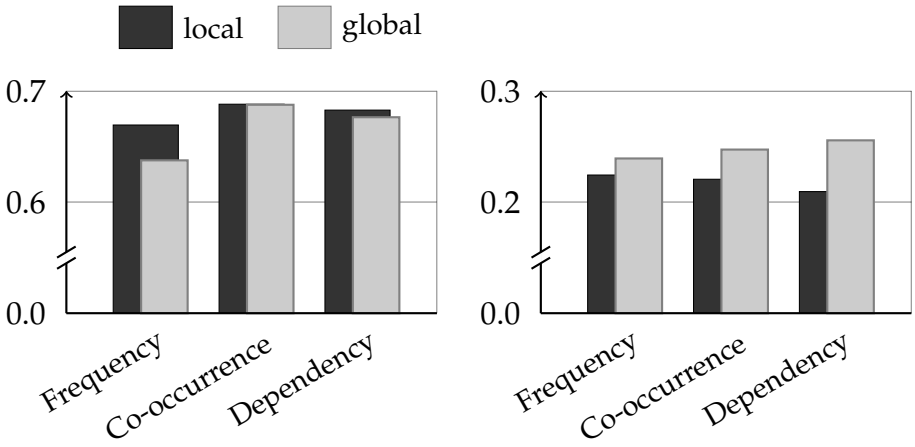
TASA900 and the frequency representation on Reuters. For the retrieval experiments, the situation is reversed. Here the global measures perform better than their counterparts among the local measures. This is seen clearly in the bar charts of Figures 7.6(b) and 7.7(b).

Although the graph-based representations are able to outperform TF and TF-IDF, it is not clear which graph type forms the better representation. This is because different experiments show both co-occurrence and dependency networks as the top performer for each of the evaluation tasks.

In short, the graph-based representations perform better than the representations based on frequencies in both experiments. A difference between the evaluation types is that local representations perform best in classification, and global measures do better for retrieval. The results unfortunately give no indication as to which of the graph-based representations is better.

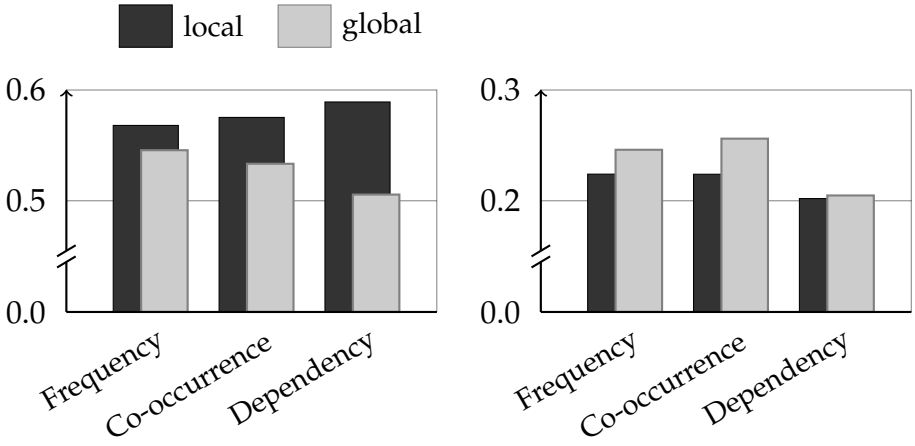
7.4 Summary

We studied the different representations and compared them to each other through a case study of central terms and two comparative experiments. The case study found that TC rated many of the same terms as TF highly. It was also clear that the dependency representation had a bias towards verbs. Among the global representations it was found that TC-ICC did not share the pronounced preference towards rare terms possessed by TF-IDF. The comparison experiments supported most of



(a) Classification (Reuters) (b) Retrieval (MIR)

Figure 7.6: Bar chart visualizing the results from the final comparison experiments listed in Table 7.2.



(a) Classification (TASA900) (b) Retrieval (AIR)

Figure 7.7: Bar chart visualizing the results from the preliminary evaluations listed in Table 7.3.

the results from the initial evaluations. The graph-based representations performed better than TF and TF-IDF in both evaluations.

Chapter 8

Discussion and Conclusions

In the preceding chapter we studied the graph-based representations, and saw that they were able to outperform the frequency-based representations in our experiments. We made several observations regarding the types of terms the different representations considered important, and differences in performance between local and global term weighting measures.

This chapter takes a deeper look at the representations in light of the experiment results, and tries to explain some of the findings. We focus on what type of information the networks are able to capture, and what this means for the final vector representations. The chapter is ended by a summary of the main conclusions of the thesis.

8.1 Discussion

In Chapter 1, we presented three hypotheses forming the basis of this thesis. We now revisit them in light of the work done and experiments performed.

We first hypothesized that the use of graph centrality would form a better basis for term weighting than term frequency. The results presented in the preceding chapter supports this hypothesis, as our graph-based weighting measures performed markedly better in both the classification and retrieval experiments.

Our second hypothesis was that term weighting would be improved by incorporating centrality in networks spanning the entire corpus. The results are here divided between the evaluation tasks. The results from the case retrieval experiments support the hypothesis, but the classification experiment does not. In fact, the use of global information seems to harm classification performance, an effect demonstrated with both TF-IDF and the TC-ICC weighting measures. This is discussed further in Section 8.1.3.

The third hypothesis stated that dependency networks should perform better than co-occurrence networks. While dependency networks did perform better in some of the experiments, co-occurrence networks worked better in others. The hypothesis is thereby neither supported nor rejected by these results. More experiments are needed in order to determine the significance of using grammatically defined term dependencies as opposed to raw co-occurrences, and under which conditions each of these is best suited.

8.1.1 The Experiment Results

Although the results show improvements over the frequency-based weighting measures, they are based on only two experiments. How much can we generalize from this and what confidence can we have in the performance of the representations?

The experiments involved two different evaluation methods performed on separate datasets, and were done in addition to the initial evaluations performed as part of the development of the representations throughout Chapters 5 and 6. The fact that the results of the new experiments were consistent with those of the preliminary evaluations strengthens our confidence in the results.

We concede that our version of the retrieval task, as described in Section 4.3.3, is not an established method of evaluation. It is based on the retrieval phase of TCBR, but with a novel unsupervised performance measure due to the lack of explicit case similarity data. We believe that the task able to capture the relevance of retrieved cases, but there is clearly room for improvements. Standard VSM similarity is currently used, but better method for assessing the relevance of retrieved cases would provide more confidence in the results.

The two evaluation tasks are complemented by each other. Classification tests whether the representations are able to identify documents concerning the same topics, i.e. documents from the same categories. It cannot tell, however, how good a match is beyond being correct or not. This is what is measured by the retrieval evaluation. The retrieval task is unaware of category information, but tries to measure the quality of retrieved case directly.

To conclude, although we cannot make strong generalized predictions based on these results, they seem promising for graph-based term weighting approaches. Experimentation with additional datasets and evaluation measures are required to further evaluate the representations and their usefulness. Ideally, retrieval evaluation performed with manually determined solution similarities would form the most proper evaluation.

8.1.2 Interpreting the Representations

What information is actually captured by the text networks, and how is this information retained in the term vectors by the centrality-based weighting?

Local interactions between terms form the essence of what is captured by the networks. Both co-occurrence and dependency networks capture relations between terms in small textual contexts. In co-occurrence networks all local interactions are included, while dependency networks include syntactically predefined relations. Neither network type retain term order directly, however, since both are undirected. Document structure, at the level of term proximity and dependency, is nonetheless retained as term connectedness, and this is the main distinction between the graph-based representations and the frequency-based BoW models.

While the structure is obviously not stored explicitly in the final term vectors, the weight of each term is determined based on information about the structure through the centrality algorithms. The information is used differently depending on which centrality measures is used to calculate the weights.

The unweighted degree centrality, which is used with the co-occurrence networks on both evaluation tasks, is one of the simpler measures and arguably the one

that retains the least structural information in the term weights. All that is stored here is the number of distinct context terms each term occurs together with.

The measures used with the dependency networks are more interesting, as they retain more structural information. Eigenvector centrality is here used for the retrieval task, and closeness centrality for classification.

Although the eigenvector centrality, like the unweighted degree, is concerned with the direct neighbors, the extracted information is very different. Eigenvector centrality does not care about the number of distinct neighbors of the weighted terms, but rather the strength of the connections and the importance of the neighbors. Thus, the connectivity of important terms influence the weighting in a way not possible in a BoW model.

The closeness centrality is able to retain other aspects of the structure. The connectivity is here adhered to so that terms with short network paths to others are given priority. This way, the terms best able to connect others in the text are considered important.

While being the best performing measure for dependency networks, closeness centrality measures performed worst for co-occurrence networks. A possible reason for this is that co-occurrence networks include many relations of doubtful semantic and syntactic validity. These "shortcuts" may interfere with the more meaningful relationships. In dependency networks, on the other hand, where all edges are predefined syntactic relations, short paths better corresponds to syntactic distances in the text, thus enabling the closeness centrality to work properly.

We note that there is a correlation between the rigidity of the term relations captured in the networks, and the complexity of the centrality measures that seem to perform best. Co-occurrence networks are based on very loosely defined relations, and here the simplest of the centrality measures were found to perform best. For the more strictly defined dependency networks, conversely, more complex centrality measures performed better.

Most of the centrality measures presented in Section 3.3 are based on notions of spreading of information in social networks. Analogies of this can also be made for the text networks. Edges in social networks represents relationships between social agents, i.e. channels of information and influence between people. Nodes in our text networks are terms — far simpler entities. The relations between terms can still be seen as channels of communication, since the meaning of terms are influenced by those surrounding it. Sentences constitute the smallest containers of information in text, and convey a piece of information or fact through the interactions of its terms.

Undirected networks worked best for both network types, and the direction of this "communication" is thus not considered important by either. This indicates that the cooperation of the terms in our metaphor represents a mutual, rather than a unidirectional, flow of information.

The captured term relations are, like social networks, to a large extent syntactic rather than semantic in nature. While our networks might capture the relations, possibly even the type of relations, they do not capture the semantics of the communication. The identification of semantic information in the text for use in the network representations is thus one of the most obvious possibilities for improvements.

8.1.3 Global versus Local Representations

One of the most interesting results from the experiments presented in the previous chapter is that, contrary to our hypothesis, the local measures (TC and TF) performed better than the corresponding global ones (TC-ICC and TF-IDF, respectively) for classification. While the differences between the local and global versions the measures varied, the local version was better in every case.

The TF-IDF measure, originally used in IR, has been widely adopted in text classification research. IDF weights rare terms higher than common ones, which has proved useful in information retrieval. It is not clear that this is the best approach for the text classification task, however. In fact, several researchers have suggested that the use of IDF is indeed inappropriate for text classification, and that supervised term weighting methods should be used instead (Batal and Hauskrecht, 2009; Soucy and Mineau, 2005). Our results show that this preference for rare terms in TF-IDF is indeed harmful to classification performance.

The same trend is to some extent also present for the TC-ICC measures. Analogous to how IDF favors rare terms, ICC weights those terms higher that have low overall centrality in the document collection. Although the difference in performance for TC-ICC, compared to TC, differs between the experiments done on the Reuters and the TASA900 datasets, we found that TC consistently performed better. This leads us to conclude that the inclusion of global information is harmful also for the graph-based measures.

We do not know exactly why the preference for local measures is much more distinct on TASA900 than on Reuters. One explanation might be that since TASA900 is a topically much more diverse corpus, there may be more rare terms for the representations to identify, thus making the effect more pronounced. In Section 7.1 we found that the weighting of terms in TC-ICC differed somewhat from that of TF-IDF. While TF-IDF identify as important many extremely rare terms, TC-ICC will not weight such terms highly unless they also are central in the relevant document. This should intuitively lead TC-ICC to be less influenced by the rare terms than TF-IDF is. As we see, this is the case for the Reuters dataset, but not for TASA900.

For the retrieval task, as previously mentioned, the global representations consistently performed better. This is not too surprising, considering that this task is much closer to standard information retrieval, for which TF-IDF was originally designed. Another factor that must also be considered here is that the retrieval evaluation might be intrinsically biased towards use of global information in weighting. To compare the retrieved solutions, a measure of text similarity was needed. Lacking any well-established similarity assessment for case retrieval, TF-IDF with cosine similarity was used for this purpose. The evaluation might therefore be biased towards representations resembling TF-IDF. TF-IDF was nevertheless outperformed by the TC-ICC representations.

8.1.4 Remarks Regarding Computational Costs

It should come as no great surprise that the use of graphs is computationally far more costly than the frequency-based measures. Both the construction of text networks and the subsequent calculation of their node centralities contribute to this

cost. Of the two stages, the node centrality calculation is by far the most expensive one.

The most important factor in terms of computational costs is the choice of centrality measure. While the degree centrality, which we found to perform best for the co-occurrence networks, has complexity linear in the order of the network, others are more costly. Generally, the group of eigenvector-based centralities is the most expensive to compute, as it requires matrix operations on the entire adjacency matrix of the graph. There are also two measures even more expensive than the eigenvector measures: the current-flow versions of closeness and betweenness. The cost of these measures on large graphs tend to be high both in terms of processing and memory requirements.

For co-occurrence networks, as noted in Section 5.2.1, the context type also affect the computation needed. Smaller contexts lead to sparser graphs, which in turn require less computation in order to find the centralities. This effect also applies to dependency networks, which generally are more sparse than co-occurrence networks.

The really large computational cost comes first when information pertaining to the entire corpus is required for the TC-ICC measure. The computation of centrality on such large networks can become very heavy, again depending on the centrality measure used. With the use of global information in the representations, the corpus centrality should ideally be recomputed for each insertion or deletion of documents in the document collection, and the document vectors recomputed at that time. Depending on the used centrality measure, this might be impractical for very large and dynamic document collections.

It is important to note, however, that these are one-time costs, and the calculations do not need to be performed at the time of similarity assessment and retrieval. The computations need only be made once per document, so that the vector-representation can be stored and used for all subsequent document comparisons.

8.2 Conclusions

We have presented an approach to text representation that serve as an alternative to the way terms are weighted in commonly used frequency-based vector representations, such as TF and TF-IDF. The approach is based on the construction of text networks, which in turn are converted into vector representations. Unlike the TF and TF-IDF representations, our graph-based representations are able to incorporate information about the structure of the text such as term proximity and syntactic relations between different terms.

Stated briefly, the representation process consists of two steps. First, textual documents are converted to networks using terms as nodes, and inter-term relations as edges. Second, node centrality algorithms determine the importance value of each term in the network, which subsequently form the basis of a term-vector representation of the document. Once such term-vectors are created, standard Vector Space Models can be used to measure document similarity.

Two types of graph representations has been explored, based on different types of relations among terms. The first represents co-occurrences of terms in the same

contexts in form of edge weights. Contexts are defined as n-term windows or sentences within the text. The second representation uses more formally defined term-relations. Specifically, edges represent semantic relations within sentences as defined by the Standford Dependency grammar.

The graphs are converted to vector representations using two different weighting methods, one local and one global. The global measure, TC-ICC, incorporates information both about the terms' centrality in the document as well as their overall centrality in the rest of the corpus. The local measure, TC, is based solely on centralities within the particular document.

We have tested the representations using two evaluation tasks. The first is a standard text processing task: classification of documents. The second is a novel evaluation method based on the retrieval process from CBR, where we assess the quality of retrieved cases by the similarity of the solution of the retrieved case to that of the query.

Although the results were inconclusive as to whether co-occurrence networks or dependency networks form the better basis for term weighting, both were able to outperform TF and TF-IDF in all experiments performed. We found that local representations, including both TF and TC, performed better than their global counterparts in the classification task. For retrieval, on the other hand, global representations seemed to perform better.

Although providing improvements over the frequency-based representations, we do not believe that the case similarity measurement performance achieved by the graph-based representations is good enough to be used for the purpose of case retrieval alone. For that, the representations are still too shallow, and unable to take into account the deeper meaning of the contents in the documents. The representations could, nonetheless, be used as a first step in such a retrieval process, as a means to narrow the search to the most relevant cases.

We note that the improvements in performance come with a caveat; the process of graph construction and centrality measurement is considerably more complex than the basic frequency measures. This is, however, only a one-time cost required when new documents are added to the document collection, and needs not be done during the retrieval process.

Altogether, graph-based representations seem promising as a way to improve text similarity measurement, and as the basis for case representation in TCBR. The work presented in this thesis is only an initial exploration of this type of representations, and there are many possibilities left unexplored, both in terms of improvements for the representations and in their application.

Chapter 9

Further Work

Throughout the project we have on many occasions had ideas and seen possibilities that we for various reasons have not been able to pursue. Some of the ideas have not been directly relevant to the current research goals, and some we have been forced to give up because of time limitations. In this chapter, we present some of these ideas as suggestions for further research.

The following section briefly discusses the use of graphs for other aspects of TCBR than retrieval. Section 9.2 outlines several possible improvements to the graph-based representations. Section 9.3 next presents the idea of using graphs as a platform for extracting topics discussed in the documents, which, if incorporated into the document representations could allow for more fine-tuned similarity matching. Finally, Section 9.4 describes an idea of graph-based category models for text classification.

9.1 Other Aspects of TCBR

This thesis has focused on the assessment of document similarity, motivated by use in the *retrieval* phase of the CBR process. The retrieval phase is by far the most studied of the four REs in the CBR cycle (Section 2.1.2). We believe that the use of graph-based representations could prove useful also for other aspects of CBR.

A working TCBR system needs to be able to create solutions to query problems based on the retrieved cases. This requires a refinement of the case solution representations, and better mappings between the problem descriptions and solutions. An approach similar to that of TextRank (from Section 4.1.1) could be used to identify central parts of the solution texts. Networks built from entire cases could help identify which parts of the problem descriptions tie with which parts of the solutions. Network structures are also well suited to record various forms of domain knowledge, and could form a convenient interface for interactions with users, like the *subject graphs* discussed by Tomita and Kikui (2001).

Although we do not have an answer to how graphs could be utilized in all phases of TCBR, we feel that they have potential either as the basis of representations and reasoning, or as a support alongside other approaches.

9.2 Improvements of the Representations

Several ideas for improvements of the representations are left unexplored. We here briefly list three of them.

9.2.1 Use of Domain Knowledge

One of the main differences between TCBR and IR is that TCBR systems typically rely on domain knowledge to solve problems. In this regard, the text network representations studied in this thesis are more similar to representations from IR. We see the use of domain knowledge, as well as general semantic information from sources such as WordNet, as one of the most promising possibilities for improving the representations.

The use of semantic knowledge can be done on two levels. First, it could be incorporated as part of the term weighting measure. Terms that for various reasons are known to be important features for cases in the domain should be given more influence. Second, and more interestingly, this knowledge could be used as a part of the network representations. This could introduce new domain dependent relations between terms, and enable terms with similar meaning to be collapsed into the same concept.

9.2.2 Improved TC-ICC Measure

TC-ICC term weighting measure, as presented in in Section 4.2.2, is a novel idea. All previous research on this application of centrality in text networks, discussed in Section 4.1, use the within-document centrality directly. The idea of combining global with local centrality shows promise, at least in the retrieval experiments, and we would like to see it explored further.

The measure, defined by Equation (4.1) on page 41, can undoubtedly be improved. This formula was our attempt at combining the local and global aspects of term relevance. The case study presented in Section 7.1 indicated that the local component might be overly influential, and that the global aspects should be given more weight. A reformulation of the measure might thus lead to improvements.

We also made a simplifying assumption when creating TC-ICC. It was assumed that the same measures should be used as basis for both TC and ICC. It would be interesting to see this assumption relaxed, as the case might be that different mechanisms are important for assessing the two types of term importance.

9.2.3 Feature Selection

Except for the removal of stop-words, the representations described in this thesis use all terms from the documents as features for the term-vectors. Many of these terms might have little or no relevance to the task of similarity measurement and retrieval, and should thus be removed. This could be done by performing feature selection on the representation, which would lead to both better performance and lower storage requirements.

Das (2001) divides feature selection algorithms into two classes: *wrapper methods* and *filter methods*. While wrapper methods employ the intended application, e.g. a classification algorithm, as a measure to directly evaluate the features, the filter methods do not rely on any information about what the representation will be used for. Since it would be unwieldy to wrap the task of TCBR retrieval for this purpose, we believe filter methods would be the better choice in this case. Yang and Pedersen (1997) present a comparative study of filter methods for text categorization.

Feature selection for our graph-based text representations could conceivably be done in several different ways. One method would be to apply a standard statistical method like those described by Yang and Pedersen. Another possibility is to define graph-based feature selection methods, either together with or instead of statistical methods.

Graph-based feature selection methods may be applied both before or after centrality calculation. Feature selection done prior to calculation of centralities will affect the weights of the terms not removed, while feature selection done later only changes which terms are included in the term-vectors. Possible filtering methods include filtering of node terms based on part-of-speech information — an approach that reportedly worked well for Mihalcea and Tarau (2004) — or removal based on graph metrics such as low centrality, degree, or clustering.

9.3 Topic-Based Similarity

One of the ideas we explored, but did not have the time to fully implement and evaluate, was the possible use of the network representations for extracting and using topic models from documents. We have here compared entire documents by representing them as term-vectors weighted based on node centrality. We believe that by extracting topics from the documents, an alternative approach to the representation and comparison process is possible.

The approach is inspired by the fact that each document, at least in reports such as the ones in the AIR and MIR datasets, deals with a number of different topics. Instead of comparing a document to another one directly, we could try to compare each topic to its counterpart in the other document. An overall similarity between documents could then be obtained by combining the similarities found between topics.

We know that the document networks have high clustering coefficients, since they were found to be small-world networks (Sections 5.4 and 6.5). This indicates the presence of cliques in the network, and could be an indication that topically related words tend to be connected.

Section 9.3.1 next describes how the topics could be extracted. How they could be used to represent documents is then discussed in Section 9.3.2. A new definition of document similarity, based on the topic-based representations, is described in Section 9.3.3.

9.3.1 Creating Topic Models

There has been done much research on how to model abstract topics that occur in documents. Several well known methods exists, among which Latent Dirichlet allocation (LDA) perhaps is the one most commonly used. What we consider here is a far simpler model, serving as an extension to the approach described in Section 4.2. The difference is that instead of representing the entire document, we would like to extract only those parts concerning a particular topic. Like documents, topics could be represented by a set — or bag — of weighted terms.

In order to determine which words should go into which "topic bag", we can define one or more seed words central to the topics which we wish to represent. Based on these seed words, we retrieve related terms based on relations in the text networks. This idea is based upon the assumptions that the text networks capture the suited relatedness between terms in such a way that terms closely linked to each other in the network are likely to be from the same topic.

The simplest way to extract topic related terms is to use terms directly linked to the seed words in a network created from the entire document collection. Since the number of terms connected to the seed words is likely to be very high, we need some way to filter what terms to include. We want only those terms that are important in the documents, and thus hopefully in the topics. To identify these, we suggest to apply a centrality measure and simply select those terms with the highest value. The result is a list of terms relevant to the topic.

Given enough seed words, we can create models for all topics known to be discussed in the documents. How to use these topic models to represent documents is discussed next.

9.3.2 Document Representation

Once we have identified a set of keywords to represent each topic, we can use these to represent documents. Instead of using terms as features in the document representation, we use topics instead. Each topic is represented by a vector, and each value in the topic vector is the centrality of that topic keyword in the network representation of the document. The documents are thus represented, not as a single feature-value vector as before, but as a set of vectors representing the various topics.

9.3.3 Document Similarity

Once documents are represented as sets of topic vectors a new method for measuring document similarity is needed. Before, document similarity was defined simply as the cosine of two document feature-vectors. The cosine measure is still applicable to compare each individual topic, but the topic similarities needs to be integrated to a total document measure. Thus, we define the new document similarity as

$$\mathrm{Sim}(d_1, d_2) = \frac{1}{|\mathrm{topics}|} \sum_{t \in \mathrm{topics}} \mathrm{cosine}(d_1^t, d_2^t) \tag{9.1}$$

where d_i^t is the vector representing topic t in document d_i. The normalization by the number of topics is needed in order to keep the similarity value in [0,1].

We believe this approach could improve the measure of document similarity by using both domain knowledge in form of seed words, and the information that is inherent to the text network representations. The extracted topic keywords might also be useful for other purposes.

9.4 Category Models for Classification

We used a *k*-NN classifier trained on a portion of the documents for the classification task presented in Section 4.3.1. This approach rests upon the assumption that the trained documents are representative for their respective categories. If this is not the case for some or all documents, classification accuracy might be reduced.

An alternate approach to this instance-based classification is to train explicit category models. This could be done by constructing category-wide text networks from all training documents pertaining to each category. TC or TC-ICC could then be used to construct term centrality vectors for each category, which could be used to compute document-category similarities. Classification would then simply be the process of assigning a document to the most similar category.

A disadvantage with this type of eager learning, compared to the lazy instance based *k*-NN approach is that it becomes more difficult to do continuous training. With the introduction of new training documents the category models must be updated, which could be a costly endeavour.

A benefit of having explicit category models is that the TC-ICC would be using category information. As defined in Section 4.2.2, TC-ICC gives preference to terms that are central in a document, but not in the rest of the corpus. With category networks, the preference is given instead to terms that are central for the given category relative to the rest of the corpus, and thus weighting higher terms characteristic to the category.

Since the focus of this thesis was not to optimize the classification process as such, but rather to assess the performance of the different representations in the measurement of document similarity, we have not needed explicit category models. The idea is, however, interesting as a way to reduce the training noise for classification, and we would like to see how such an approach would compare to other eager learning methods for classification.

References

Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59.

Albert, R. and Barabási, A. (2002). Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47–97.

Barabási, A. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, pages 1–11.

Barabási, A. (2003). *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, And Everyday Life*. Plume New York.

Batal, I. and Hauskrecht, M. (2009). Boosting KNN Text Classification Accuracy by using Supervised Term Weighting Schemes. In *Proceedings of the 18th ACM conference on Information and knowledge management - CIKM '09*, pages 2041–2044. ACM Press.

Bisson, G. and Hussain, F. (2008). Chi-sim: A new similarity measure for the co-clustering task. In *2008 Seventh International Conference on Machine Learning and Applications*, pages 211–217. IEEE.

Bonacich, P. (1972). Factoring and weighting approaches to status scores and clique identification. *The Journal of Mathematical Sociology*, 2(1):113–120.

Brandes, U. (2008). On Variants of Shortest-Path Betweenness Centrality and their Generic Computation. *Social Networks*, 30(2):136–145.

Brandes, U. and Fleischer, D. (2005). Centrality Measures Based on Current Flow. *STACS 2005*, pages 533–544.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117.

Burke, R., Hammond, K., Kulyukin, V., Lytinen, S., Tomuro, N., and Schoenberg, S. (1997). Question Answering from Frequently-Asked Question Files: Experiences with the FAQ Finder System. *AI magazine*, 18(2):57.

Chakraborti, S., Wiratunga, N., Lothian, R., and S (2007). Acquiring Word Similarities with Higher Order Association Mining. *Case-Based Reasoning*, pages 61–76.

Choudhury, M. and Chatterjee, D. (2010). Global topology of word co-occurrence networks: Beyond the two-regime power-law. *23rd International Conference on*

Computational Linguistics, pages 162–170.

Clauset, A., Shalizi, C. R., and Newman, M. E. J. (2009). Power-Law Distributions in Empirical Data. *SIAM Review*, 51(4):661.

Cunningham, C., Weber, R., Proctor, J., Fowler, C., and Murphy, M. (2004). Investigating graphs in textual case-based reasoning. *Lecture notes in computer science*, 3155:573–586.

Das, S. (2001). Filters, wrappers and a boosting-based hybrid for feature selection. In *Proceeding of the Eighteenth International Conferences on Machine Learning*, pages 74–81.

De Marneffe, M. and Manning, C. (2008a). Stanford typed dependencies manual.

De Marneffe, M. and Manning, C. (2008b). The Stanford typed dependencies representation. *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation - CrossParser '08*, (ii):1–8.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Dhillon, I. S. (2001). *Co-clustering documents and words using bipartite spectral graph partitioning*. ACM Press.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

Dorogovtsev, S. N. and Mendes, J. F. (2001). Language as an evolving word web. *Proceedings. Biological sciences / The Royal Society*, 268(1485):2603–6.

Erdős, P. and Rényi, A. (1960). On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–60.

Erkan, G., Ozgür, A., and Radev, D. R. (2007). Semi-supervised classification for extracting protein interaction sentences using dependency parsing.

Erkan, G. and Radev, D. R. (2004). Lexrank: Graph-based Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22.

Ferrer i Cancho, R. (2004). Euclidean distance between syntactically linked words. *Physical Review E*, 70(5):1–5.

Ferrer i Cancho, R. and Solé, R. (2001). The small world of human language. *Proceedings. Biological sciences / The Royal Society*, 268(1482):2261–5.

Ferrer i Cancho, R., Solé, R., and Köhler, R. (2004). Patterns in syntactic dependency networks. *Physical Review E*, 69(5):1–8.

Freeman, L. C. (1978). Centrality in Social Networks Conceptual Clarification. *Social Networks*, 1(3):215–239.

Fundel, K., Küffner, R., and Zimmer, R. (2007). RelEx–relation extraction using de-

pendency parse trees. *Bioinformatics (Oxford, England)*, 23(3):365–71.

Gao, J., Nie, J.-Y., Wu, G., and Cao, G. (2004). Dependence Language Model for Information Retrieval. *Proceedings of the 27th annual international conference on Research and development in information retrieval - SIGIR '04*, pages 170–177.

Goh, K., Kahng, B., and Kim, D. (2001). Universal behavior of load distribution in scale-free networks. *Physical Review Letters*, 87(27).

Guare, J. (1990). *Six degrees of separation*. Dramatists Play Service, Inc.

Hussain, S. and Bisson, G. (2010). Text Categorization Using Word Similarities Based on Higher Order Co-occurrences. In *SIAM International Conference on Data Mining (SDM 10). Columbus, OH*, pages 1–12.

Jeong, H., Albert, R., and Barabási, A. (1999). Diameter of the World Wide Web. *Nature*, 401(September):398–399.

Jeong, H., Tombor, B., Albert, R., Oltvai, Z., and Barabási, A. (2000). The large-scale organization of metabolic networks. *Nature*, pages 651–654.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

Kolodner, J. (1993). Case-based Reasoning.

Kontostathis, A. and Pottenger, W. (2003). A framework for understanding LSI performance. In *Proceedings of ACM SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*, number 3, pages 249–268. Citeseer.

Leake, D. (1996). *CBR in context: The present and future*, chapter 1, pages 1–35. Number Schank 1982. AAAI Press/MIT Press.

Lemaire, B. and Denhière, G. (2006). Effects of High-Order Co-occurrences on Word Semantic Similarity. *Word Journal Of The International Linguistic Association*, 1(18):1–12.

Lenz, M. (1998). Textual CBR and Information Retrieval – A Comparison. In *Proceedings 6th German workshop on CBR*, number c.

Lenz, M., Hübner, A., and Kunze, M. (1998). *Case-Based Reasoning Technology, From Foundations to Applications*, chapter 5. Textual CBR, pages 115–137. Springer-Verlag, London, UK.

Liu, J., Wang, J., and Wang, C. (2008). A Text Network Representation Model. *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, pages 150–154.

Livesay, K. and Burgess, C. (1998). Mediated priming in high-dimensional semantic space: No effect of direct semantic relationships or co-occurrence. *Brain and Cognition*, pages 102–105.

Mengxiao, Z., Zhi, C., and Qingsheng, C. (2004). Automatic keywords extraction of Chinese document using small world structure. In *Proceedings of International*

Conference on Natural Language Processing and Knowledge Engineering, 2003, number 70171052, pages 438–443. IEEE.

Mihalcea, R. (2004). Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, number 4, Morristown, NJ, USA. Association for Computational Linguistics.

Mihalcea, R. and Tarau, P. (2004). TextRank: Bringing order into texts. In *Proceedings of EMNLP*, pages 404–411. Barcelona: ACL.

Mihalcea, R. and Tarau, P. (2005). A language independent algorithm for single and multiple document summarization. *Proceedings of IJCNLP2005*, pages 19–24.

Mihalcea, R., Tarau, P., and Figa, E. (2004). PageRank on semantic networks, with application to word sense disambiguation. In *Proceedings of the 20th international conference on Computational Linguistics - COLING '04*, Morristown, NJ, USA. Association for Computational Linguistics.

Milgram, S. (1967). The small world problem. *Psychology Today*, 1:60–67.

Mustafaraj, E. (2007). Knowledge Extraction and Summarization for Textual Case-Based Reasoning. *Philipps-Universitat Marburg*.

Mustafaraj, E. and Freisleben, B. (2006). On an event-oriented perspective for textual case-based reasoning. In *Textual Case-Based Reasoning Workshop (TCBR) at the 8th European Conference on Case-Based Reasoning (ECCBR'06), Fethiye, Turkey*, pages 21–32.

Navigli, R. and Lapata, M. (2007). Graph connectivity measures for unsupervised word sense disambiguation. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1683–1688.

Newman, M. (2005). A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54.

Newman, M. E. J. (2001). Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. *Physical Review E*, 64(1):1–7.

Nivre, J. (2005). Dependency grammar and dependency parsing. *MSI report*, 5133(1959):1–32.

Opsahl, T., Agneessens, F., and Skvoretz, J. (2010). Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3):245–251.

Ozgür, A., Vu, T., Erkan, G., and Radev, D. R. (2008). Identifying gene-disease associations using centrality on a literature mined gene-interaction network. *Bioinformatics (Oxford, England)*, 24(13):i277–85.

Padó, S. and Lapata, M. (2007). Dependency-Based Construction of Semantic Space Models. *Computational Linguistics*, 33(2):161–199.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The pagerank citation

ranking: Bringing order to the web. *World Wide Web Internet And Web Information Systems*, pages 1–17.

Porter, B., Bareiss, R., and Holte, R. (1990). Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45(1-2):229–263.

Porter, M. (1980). An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137.

Schenker, A., Last, M., Bunke, H., and Kandel, A. (2003a). Classification of Web documents using a graph model. *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, (Icdar):240–244.

Schenker, A., Last, M., Bunke, H., and Kandel, A. (2003b). *Clustering of web documents using a graph model*, pages 1–16.

Solé, R., Corominas-Murtra, B., Valverde, S., and Steels, L. (2005). Language networks: Their structure, function, and evolution. *Trends in Cognitive Sciences*.

Soucy, P. and Mineau, G. (2005). Beyond TFIDF Weighting for Text Categorization in the Vector Space Model. In *International Joint Conference on Artificial Intelligence*, volume 19, pages 1130–1135.

Terra, E. and Clarke, C. L. a. (2003). Frequency estimates for statistical word similarity measures. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - NAACL '03*, (1997):165–172.

Tesnière, L. (1959). Eléments de syntaxe structurale. *Kliencksieck, Paris*.

Tomita, J. and Kikui, G. (2001). Interactive Web search by graphical query refinement. *Poster Proceedings of the 10th international World Wide Web conference (WWW10)*.

Tomita, J., Nakawatase, H., and Ishii, M. (2004a). Calculating similarity between texts using graph-based text representation model. *Proceedings of the Thirteenth ACM conference on Information and knowledge management - CIKM '04*, pages 248–249.

Tomita, J., Nakawatase, H., and Ishii, M. (2004b). Graph-based text database for knowledge discovery. *Poster Proceedings of the 13th International World Wide Web Conference (WWW2004)*, pages 454–455.

Tsatsaronis, G., Varlamis, I., and Nørvåg, K. (2010). An Experimental Study on Unsupervised Graph-based Word Sense Disambiguation. *Computational Linguistics and Intelligent Text Processing*, pages 184–198.

Valle, K. (2010). A Study of Graph-Based Representations for Textual CBR. Unpublished report, specialization project towards master thesis.

Wang, W., Do, D., and Lin, X. (2005). Term Graph Model for Text Classification. *Lecture notes in computer science*, 3584:19.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–2.

Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420. Morgan Kaufmann Publishers Inc.

Appendix A

Detection of Power-Laws

Section 3.2.3 introduced the notion of scale-free networks, and explained that these are networks in which the degree distributions follow power-laws. We here discuss the concept of power-laws in more detail.

In Sections 6.5 and 5.4 we investigated whether our text network representations, the co-occurrence and dependency networks were scale free, i.e. whether they had degree distributions that follow power-laws. This appendix explains how we did this, and what we found out.

A.1 Powers-Laws

Many properties tend to cluster around a central value that is useful for describing typical outcomes of measurements of that property. It is, for example, meaningful to discuss typical heights of people, typical temperatures, or typical weights of pineapples. The reason this makes sense is that although these quantities vary, they do so with a negligible probability far from the typical value. The typical value is thus representable for most observations. Many quantities follow such distributions. The perhaps best known, the normal distribution, is expressed by a *mean* and a *standard deviation*. Some distributions, however, cannot be expressed in terms of typical values, since they allow non-negligible probabilities for extreme variations.

One such distribution, the *power-law*, has been detected in a wide range of natural and man-made phenomena. For a given quantity x it is described mathematically as

$$p(x) \propto x^{-\alpha} \tag{A.1}$$

where α is a constant parameter for the specific power-law governing x. The α is known as the distribution's *exponent* or *scaling parameter*. Although there are exceptions, the value of the exponent typically lies in the range $2 < \alpha < 3$.

Few phenomena follow power-laws for all values of x. In most cases there exist a $x_{\min}$ that is the minimal value for which the power-law distribution holds. If this is the case, we say that the *tail* of the distribution follows a power-law.

A.2 Detection of Power-Laws

It is in many situations desirable to determine whether a set of empirical data follows a power-law. In practice, one can generally not be certain that observations are drawn from a power-law distribution, only that the observations are consistent with the hypothesis that x is drawn from a distribution of the form of Equation (A.1). It is, however, often possible to rule out competing hypotheses, and to provide an estimate of how well the empirical data fit the power-law hypothesis.

The simplest and perhaps most widely used method for probing for power-law behaviour is to inspect the data visually. Taking the logarithm on both sides of Equation (A.1), we find that

$$\ln p(x) = \alpha \ln x + \text{constant} \tag{A.2}$$

This equation implies that a histogram of the frequency distribution plotted with both axes on logarithmic scales should follow a straight line. If the data indeed follow a power-laws distribution, then the value of α can be read as the slope of this line. Unfortunately, the straight-line behaviour is necessary, but by no means sufficient to detect true power-laws behaviour. The histogram method can still, in any case, be a helpful tool to rule out power-laws or to get an idea about possible values for $x_{\min}$ and α.

Clauset et al. (2009) discuss the problem of detecting power-laws in detail, and provide a method allowing the detection and estimation of power-laws from empirical data. The procedure can be summarized as follows:

1. Estimate the parameters α and $x_{\min}$ of the power-law model.
2. Calculate the goodness-of-fit of the observed data for the power-law model using the *Kolmogorov-Smirnov* (KS) statistic. Given a p-value grater than 0.1, the power-law is a plausible hypothesis, otherwise it should be rejected.
3. If the power-law hypothesis is not rejected in step 2, compare it with alternate hypotheses via a likelihood ratio test. For each alternate hypothesis, if the likelihood ratio is significantly far from zero, it indicates whether the alternate hypothesis is favoured over the power-law model or not.

The first of these steps, the fitting of a power-laws model to the empirical data, can be done by the method of maximum likelihood. We will not go into details about how this is done. Methods for estimating α and $x_{\min}$ are described far better by Clauset et al. (2009, Section 3) than we could have done here.

The second step of the process is to determine how well the power-law model estimated in the first step actually fits the data. As stated by Clauset et al. (2009, p.14):

> *Regardless of the true distribution from which our data were drawn, we can always fit a power-law. We need some way to tell whether the fit is a good match of the data.*

Even if our data were drawn from a power-law distribution, there would be some deviations because of sampling. We need to distinguish these deviations from those occurring given data drawn from non-power-law distributions. The basic approach to do this is to sample many datasets from a true power-law distribution and

measure their fluctuations from the power-law form. Then, these fluctuations can be compared to those of the empirical data. The distance between distributions can be measured using a goodness-of-fit measure such as the KS statistic. The measure generate a p-value defined as the fraction of generated deviations that are larger than the empirical deviation. A large p-value gives confidence in the power-law hypothesis, while a value of $p \leq 0.1$ should cause the power-law hypothesis to be rejected.

If the power-law hypothesis survives step 2, we can be reasonably sure that the estimated power-law is a good description of the data. There might also be other equally or better suited models to describe the same observations. This is what is tested by the *likelihood ratio test* in the third step of the method.

The idea behind the likelihood ratio test is to compute the likelihood of the observed data under two competing distributions. The distribution with higher likelihood is the better fit. Alternately, the ratio of the likelihood may be used as an indicator. A value for this ratio near zero indicate a tie, i.e. that the distributions fit the data about equally well, while a positive or negative value favours one of the distributions or the other.

A.3 Our Experiments

This section explains the process we went through in order to detect whether the degree distributions of our networks followed power-laws, and whether these laws really fitted the data good enough to be an appropriate model.

To do the actual evaluation, we have made us of software developed by Clauset et al. (2009) which they have made available online[1]. We initially experimented with the Python implementation, but experienced problems [2] , and ended up using the MatLab implementation instead.

Since all the research reported in Section 3.4 focused on text-based networks crated from large corpora, we started by investigating networks created from our datasets in whole. This gave us four networks to look at: two datasets, each with two network representations.

As a first approach, we investigated the frequency histograms for each network as a log-log plots. These can be seen as the line of blue dots in Figure A.1. It is immediately clear that we will not find power-laws to describe any of the distributions completely, since neither is linear in the log-log plots.

However, as we described in Section 3.4, Ferrer i Cancho and Solé (2001) discovered that this kind of networks often do not follow one single, but rather two separate power-laws. In order to determine if this was the case also for our networks, we performed the following experiment using the software developed by Clauset et al. (2009).

1. Estimate parameters α and $x_{\min}$ for the tail of each of the distributions. The beginning of each tail-distribution is determined by the value of $x_{\min}$.

[1] `http://tuvalu.santafe.edu/~aaronc/powerlaws/`

[2] It is more than likely that the problems occurred as a result us not configuring the software correctly, or perhaps by an incompatible version of other libraries such as NumPy. In any case, we were unable to resolve the errors.

2. Estimate parameters for the rest of each distribution, i.e. for all samples with value $x < x_{\min}$ as calculated in step 1.
3. Determine whether the fitted power-laws are plausible models, using the KS goodness-of-fit measure.

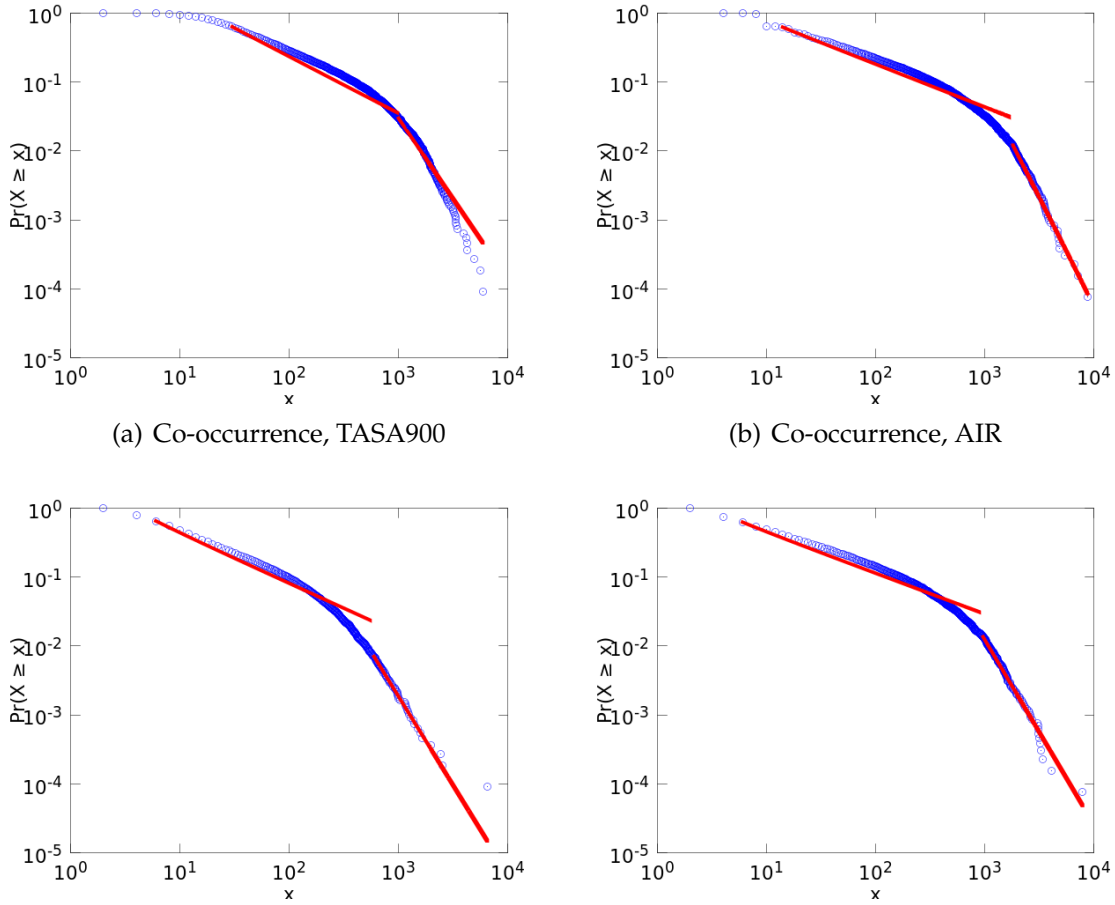
For all networks, except the co-occurrence network from TASA900, the $x_{\min}$ values detected for the tail distributions were approximately centered on the bends in the histograms. For the TASA900 co-occurrence network, the software tried to fit a power-law to the entire distribution. We therefore had to force a value of $x_{\min} = 1000$ to estimate a reasonable α for the tail of the distribution. Values for α and $x_{\min}$ were thus estimated for all the distributions. The superimposed lines in Figure A.1 show the estimated power-laws.

Table A.1: Results from the power-law detection experiment.

		Co-occurrence		**Dependency**	
		TASA900	AIR	TASA900	AIR
tail	α	3.34	4.12	3.58	3.68
	$x_{\min}$	1000	1806	596	962
	p	0	0	0.912	0.562
	n	325	158	76	176
base	α	1.82	1.62	1.72	1.59
	$x_{\min}$	30	14	6	6
	p	0	0	0	0
	n	6315	7533	5893	6909
	n_{rest}	4251	4373	4920	5903

From the plots, it seems that the two-regime power-laws could possibly be reasonable models for the distributions, but the results of the goodness-of-fit tests in Table A.1 disagree. As we see, only the tail distributions for the dependency networks have p-values above zero. These values are on the other hand high enough to mark the estimated power-laws as plausible. We note that the tail distributions cover only a miniscule fraction of the total number of samples compared to the base distributions. There are also a very high number of samples, n_{rest}, that is unaccounted for by any of the fitted power-laws. We are therefore forced to conclude that the degree distributions of networks created by our text network representations over the entire corpora do not follow power-laws, two-regime or otherwise. This is an unexpected result, as it is goes against results of previous research on such networks.

We have also studied the individual document networks to see whether they behaved different from the larger corpus networks. After preliminary studies of the frequency histograms, we found that they varied greatly between networks. Some networks had distributions that when plotted looked almost like those for the corpus networks, while some plots looked more linear in log-log. Some of the latter actually turned out to fit power-law models reasonably well, but they were few and far between.



(a) Co-occurrence, TASA900

(b) Co-occurrence, AIR

(c) Dependency, TASA900

(d) Dependency, AIR

Figure A.1: Frequency histograms of degree distributions from the four corpora networks plotted with both axes logarithmic, with estimated two-regime power-laws superimposed.

There were two main problems with determining whether the distributions governing the document networks followed power-laws: in addition to the variations between networks, the number of samples was very limited in most cases. It was therefore hard, even when a power-law model could be fitted, to determine with high confidence whether it really was plausible. Over all document networks, we found p-values ranging from 0 to 0.983, with an average of 0.192. For TASA900, the number of networks with p-values higher than the rejection threshold of 0.1 was 254 for co-occurrence and 145 dependency networks. The corresponding numbers for AIR were 131 and 201, respectively. Given this, and the low number of samples n for many of the networks, we do not feel confident predicting anything about the underlying distributions based on this.

We note that, despite not actually being power-laws, the distributions still display some of the same properties. Most notably, the distributions have long tails, which implies the presence of hubs, i.e. that while the majority of nodes have a relatively low degree, there exist a few nodes with very high degrees. Of the 10894 nodes in the co-occurrence network created from the entire TASA900 corpus, the biggest hub is connected to 5906 other nodes, which is more than half of the net-

work. Just as in true scale-free networks, these hubs are very rare; in the entire TASA900 network only 17 nodes have degrees higher than 3000. As many as 7775 (71.4%) of the nodes in fact have degrees lower than 100. The other networks also share similar properties.

Appendix B

Implementation

This appendix is intended as a general description of how the representations and experiments are implemented, as well as an introduction to the code should anyone wish to extend upon or use it at a later time.

The appendix start with a bird's-eye view of the code, depicting the architecture and describing how the various modules interact with each other. We then move on to a more detailed explanation of each of the modules, with descriptions of what they do and their most important functions. In Section B.3, the experiments contained in the various `*_experiments.py` files are outlined. Section B.4 then describe the various libraries utilized by the code. The final section contain some final thoughts about the implementation, and experiences made during the project. These are included in the hope that they may guide anyone interested in using the code, or attempting to implement something similar.

For a more detailed view of how everything is implemented, please see the HTML documentation bundled with the code, or the code itself. The documentation contains much of the information from this appendix, and additional details about the functions and classes in the various modules. The code is available for download from my GitHub account: `http://github.com/kvalle`.

B.1 Architecture

This section expands upon Section 4.3.5, explaining in more details what is done, and how everything fits into the process described there. We refer to Chapter 4 for a fuller conceptual description of the general process.

Figure B.1 presents the architecture, the modules and how they interact. This figure is based largely on Figure 4.9 on page 48. Differences are that a few additional modules are included, and that the directed arrows represent not the flow of data, as in Figure 4.9, but the interconnections between modules, i.e. references between modules.

The *experiments* module is not like the others. First, it is not a functional module, but rather the glue that keeps everything together. This is the module that utilize the other modules in order to perform the various experiments. Second, it is not really one module, but rather a collection of `*_experiments.py` files, containing experiments concerning the various representations. Of the connections from *experiments* to the other modules in Figure B.1, only the most important ones are shown to avoid

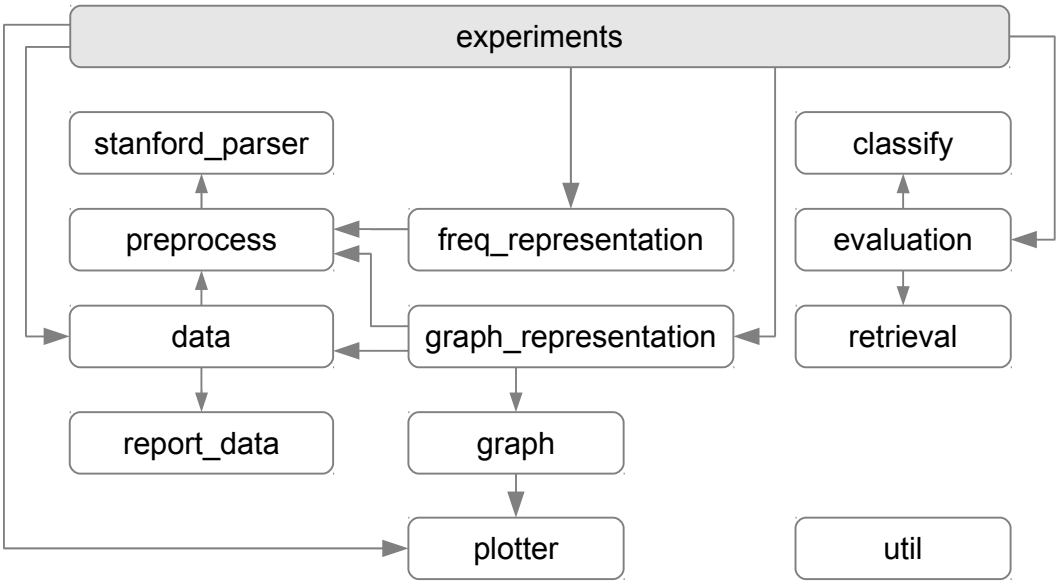


Figure B.1: Overview of the architecture.

cluttering the diagram. The contents of this "module" are described in Section B.3 below.

The *util* module is used by many of the modules. Also here have we left out the dependencies in order do avoid cluttering. The module contains miscellaneous utility functions that does not naturally fit into any of the other modules.

The three leftmost modules, *preprocess*, *data*, and *report_data*, are responsible for reading and doing textual preprocessing of the cases. *data* handles all file I/O, and utilize *preprocess* to make the necessary changes to the text. All preprocessing tasks are done by *preprocess* itself, except dependency parsing which is handled by *stanford_parser*. The *report_data* module is used by *data* to retrieve textual cases from HTML-formatted documents in the AIR dataset.

The middle column of modules handles representation of documents as feature vectors. *freq_representation* represents documents as TF and TF-IDF vectors, while *graph_representation* build networks from the text and create vectors based on node centrality. The actual graph data structures and functions are contained in *graph*. The *plotter* module is used to visualize the networks.

The three modules on the right are used to evaluate the feature vectors created by the above modules. The two evaluation methods described in Section 4.3 are implemented in *classify* and *retrieval*, and *evaluate* provide an interface to these.

B.2 Modules

The modules are presented in more details below. Each module is given an overall description, both about its purpose and how it is used. Only the most important methods are mentioned. For more details, see the code itself or the accompanying documentation.

B.2.1 data

Module for reading and writing case files. The `read_*` methods provide reading of cases in various formats from dataset. For converting dataset between formats, use the appropriate `create_dataset_*` function. It is also possible to provide custom conversion functions to the `create_dataset()` function. The module expects to work with datasets structured so that each category is in a separate subfolder named after the category.

The following formats are supported for dataset conversion:

HTML: Expected to be formatted similarly to reports from the AIR or MIR datasets if conversion to cases is intended. Conversion to text/dependencies should work regardless.

Text: Raw text. This is anything from within $\langle p \rangle$-tags if extracted from HTML.

Preprocessed text: Text preprocessed by the *preprocess* module. The default parameters from `preprocess.preprocess_text()` can usually be applied.

Dependencies: Dependencies extracted from raw text by the Stanford Lexical Parser.

B.2.2 report_data

Helper module for *data*, used to extract problem description and solution parts from cases. Can parse reports formatted in HTML, structured as those in the AIR or MIR datasets, and split them into problem description and solution parts of textual CBR cases. Solutions are identified based on section titles in the reports. Titles matching words such as *finding* or *conclusion* are considered as part of the solution. The remaining report is by default the problem description.

This module is based largely upon code received from Gleb Sizov.

B.2.3 preprocess

Toolbox of functions for preprocessing text.

The module contains methods for a variety of preprocessing tasks, such as filtering out words with special characters, stemming, stop-word removal, case folding and more, as well as functions for splitting text into lists of tokens or sentences. Use `preprocess_text()` and `preprocess_token()` for full preprocessing.

Extraction of within-sentence word dependnecies is also available through the `extract_dependencies()` function, which works as an interface to the *stanford_parser* module.

The Natural Language Toolkit (NLTK) is used for most of the heavy lifting.

B.2.4 stanford_parser

Python interface to the stanford parser.

The module wrapps the `edu.stanford.npl` Stanford Parser, which is implemented in Java, using the JPype library. The `StanfordParser` class wraps the actual parser, and the `parse()` function can be used to parse sentences.

B.2.5 freq_representation

Functions for creating frequency-based feature vector from text.

The function of interest is `text_to_vector()`, which creates term frequency (TF) or term frequency-inverse document frequency (TF-IDF) vectors from lists of documents. Results are output in form of a term-document matrix.

B.2.6 graph_representation

Module contains functions from creating networks based on text documents, and for converting the networks into feature-vectors. Feature vectors are created based on node centrality in the text networks.

The following text representations are supported:

random: Will create a network with all distinct terms in the provided document as nodes. Edges are created at random between the nodes, based on provided probabilities.

co-occurrence: Distinct terms in the document are used as nodes. Edges are created between any terms that occurs closely together in the text.

dependency: Words as nodes. Edges represent dependencies extracted from the text using the Stanford dependency parser (see the *stanford_parser* module).

B.2.7 graph

Toolbox module for working with networkx graphs. The module contains functions for calculating graph centrality, visualizing graphs and finding various network properties, in addition to various other useful functions.

Graph centralities are accessed using the `centralities()` function, which takes as arguments a graph and the metric to use as a constant of the `GraphMetrics` class.

B.2.8 classify

Module responsible for classification of datapoints represented as feature-vectors using K-Nearest Neighbors classifier.

The KNN class contains the classifier. It can `classify()` new datapoints as soon as it is properly trained using the `train()` method. The `test()` method provides a way to classify many vectors at once, and return the classifiers accuracy compared to a gold standard.

B.2.9 retrieval

Evaluates lists of cases with `evaluate_retrieval()`. For each problem description the remaining descriptions are assessed, and the solution corresponding to the best matching description is retrieved. Actual solution is compared to retrieved solution using cosine of solution vectors. The overall evaluation score is the average solution-solution similarity over the case base.

B.2.10 evaluation

Module containing methods for evaluating representations.

This module acts as an interface to evaluation against the *classify* and *retrieval* modules through the `evaluate_classification()` and `evaluate_retrieval()` functions, respectively.

B.2.11 plotter

Utility functions facilitating easy plotting with matplotlib.

Functions of note:

`plot`**:** Plots a regular plot, given input x,y-coordinates.
`bar_graph`**:** Plots a horizontal bar graph from x-coordinates and named groups of lists of y-coordinates.
`histogram`**:** Plots a histogram from a set of samples and a given number of bins.
`plot_degree_distribution`**:** Plots the degree distribution for a graph given as input.
`tikz_plot`**:** Writes LaTeX for crating a TikZ plot of the data series given as input.
`tikz_barchart`**:** Writes LaTeX for crating a TikZ bar chart of the data series given as input.

B.2.12 util

Module containing miscellaneous utility functions without a proper home anywhere else.

B.3 Experiments

The experiment modules use the rest of the framework to evaluate different versions and aspects of the text network representations. The various experiments are implemented as functions.

There are four modules. *co_occurrence_experiments* for experiments with regular co-occurrence, and *higher_order_experiments* for higher order co-occurrence networks. The dependency network representation is tested in *dependency_experiments*. The general *experiments* module contain experiments concerned with several representations, or functions not directly tied to any representation such as, for example, the `dataset_stats` function.

For descriptions of each individual experiment, please see the *experiments* page in the documentation.

These modules are a mess and contain a lot of redundant code. This is because they contains experiments constructed for specific purposes that were hard to predict ahead of time. When done, the experiment functions were left as is, to be available for re-runs later if needed. As a consequence of many of the experiments, the representations and/or other parts of the code have been changed. Most of the experiments should still, however, hopefully work as expected.

B.4 Libraries Used

The implementation relies heavily on a few central third party libraries. These are briefly described in this section.

NumPy and SciPy: SciPy is an open-source library for mathematics, science, and engineering. SciPy depends on NumPy, a library which provides convenient and fast N-dimensional array and matrix manipulation, as well as many tools for numerical computation. The libraries are easy to use, but powerful for manipulating numbers in many ways.

The most useful aspects in this project were the nd-arrays and matrices from NumPy, and the sparse matrix representations from SciPy. A function from the distance module, `scipy.spatial.distance.cdist`, also proved valuable for efficient computation of vector similarities.

SciPy is available at `http://scipy.org` and NumPy can be downloaded from `http://numpy.scipy.org`.

NetworkX: NetworkX is a Python package for the creation and manipulation of graphs and complex networks. It enables study of structure and dynamics of networks, and comes with many useful functions.

We have used NetworkX's `DiGraphs` as our basic datastructure for the network representations. Among the more useful features of the library are some of the graph centrality algorithms, and functions for extracting global and local properties from the graphs.

NetworkX is available from `http://networkx.lanl.gov`.

NLTK: Python's Natural Language Toolkit (NLTK) is a powerful tool for working with natural language processing, providing functionality for a wide variety of tasks.

Only a small, but useful part of the library is used in this project. Of most use were the stemmers, stop-word lists, tokenizers for tokens and sentences, and probability distributions for calculating the frequency-based measures.

NLTK is available from `http://nltk.org`.

Matplotlib: Matplotlib is a 2D plotting library able to produce high quality graphics of many types. We have utilized it to create plots, histograms and bar charts, many of which are used in this report. The library has a relatively easy interface, able to produce figures with a few lines of code.

Matplotlib is available at `http://matplotlib.sourceforge.net`.

JPype: JPype is a Java-to-Python integration library, allowing python programs full access to java class libraries. This is not done through re-implementing python on the Java Virtual Machine, as in the JPython project, but rather through interfacing at the native level in both virtual machines.

Using JPype, we were able to use the Stanford dependency parser directly, even though it was implemented in Java.

Available from `http://jpype.sourceforge.net`.

The Stanford Parser: The Stanford parser is actually a set of probabilistic natural language parsers. It is developed and released by the Natural Language Processing Group at Stanford University. The parser is, among other things, able to produce sentence structure hierarchies, POS tags and dependency relations from sentences. Our interest in the parser was the dependency relations, which proved intuitive and easy to work with.

It is available from `http://nlp.stanford.edu/software/lex-parser.shtml`

B.5 Final Remarks

We conclude the appendix with some final remarks about the implementation in general.

As noted in the previous section, several libraries are used in the implementation, which proved very helpful. Using specialized libraries enabled us to avoid reinventing the wheel, and quickly implement what would otherwise have taken much time. Especially NetworkX proved very useful for working with large graphs, both for its graph representations and several of its graph algorithms.

There are some aspects of the code we feel might have been improved. For one, the implementation does not have a sufficiently good framework for visualizing large graphs and networks. The trouble is that networks on the scale generated from our corpora usually are too large to be drawn in a meaningful way.

The code also marked by the way in which it was implemented. The models and representations changed throughout the project, as result of experiments and new ideas. Parts of the code thus looks rather ad hoc since many design decisions were based on or influenced by conditions that have since changed.

Despite this, the code works and does what it is intended to. It has been able to answer our research questions, and provided insight in the usefulness of graph centrality weighted feature vectors for text similarity. We hope that in case anyone intends to look more at these topics, this implementation can provide some utility.