



NTNU – Trondheim
Norwegian University of
Science and Technology

Novelty Detection in Knowledge Base Acceleration

Ellen Wiig Andresen

Master of Science in Computer Science

Submission date: June 2013

Supervisor: Kjetil Nørvåg, IDI

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

Abstract

Knowledge bases provide the users of the World Wide Web with a vast amount of structured information. They are meant to represent what we know about the world the way it is today. Therefore, every time something happens, knowledge bases need to be updated according to the new happening. A knowledge base is most often organized around entities and their relations. Entities represent an object in the real world, such as religions, persons or places, and a relation is a connection between two entities. Today, the process of updating knowledge bases is purely done by humans, who unfortunately are not able to keep up with everything that happen in the world. In order to make this job easier, systems for doing Knowledge base acceleration, KBA, are proposed. They are meant to, given a stream of news, pick out what is relevant updates for the different entities in a knowledge base. To make the most of such a system, and to make sure that it only return news that provide useful information to the content managers, it should only return news that contain *new* information, that is, it should perform novelty detection.

This thesis explore the properties a KBA system need to fulfil in order to solve the task it is supposed to as good as possible. It argues that a KBA system need to include novelty detection to be useful, and present a prototype for novelty detection in a KBA system. The prototype is implemented using different approaches to novelty detection, and compare these.

Sammendrag

Kunnskapsbaser gir brukerne av World Wide Web tilgang til en enorm mengde strukturert informasjon. De er ment til å representere det vi vet om verden slik den er i dag. Derfor, hver gang det skjer noe, må kunnskapsbaser oppdateres i henhold til den nye hendelsen i verden. En kunnskapsbase er som oftest organisert rundt entiteter og relasjoner mellom disse. Entiteter representerer gjerne et objekt i den virkelige verden, slik som religioner, personer eller steder. En relasjon er en forbindelse mellom to enheter. I dag er prosessen med å oppdatere kunnskapsbaser kun gjort av mennesker, som dessverre ikke er i stand til å holde tritt med alt som skjer i verden. For å gjøre denne jobben enklere, er systemer for å gjøre akselerasjon av kunnskapsbaser foreslått. De er ment å, gitt en strøm av nyhetsartikler, plukke ut hva som er relevante oppdateringer for de ulike enhetene i kunnskapsbasen. For å få mest mulig ut av et slikt system, og for å sørge for at det bare returnerer nyheter som gir nyttig informasjon til de som velikeholder innholdet, bør det bare returneres nyheter som inneholder *ny* informasjon. Det betyr at et slikt system bør utføre deteksjon av ny informasjon.

Denne avhandlingen utforsker egenskapene et system for akselerasjon av knunnskapsbaser må oppfylle for å kunne løse oppgaven det er ment til, så godt som mulig. Den påpeker at systemer for akselerasjon av kunnskapsbaser må inkludere deteksjon av ny informasjon for å være nyttig. Det presenteres også en prototype for deteksjon av ny informasjon som en del av et system for akselerasjon av kunnskapsbaser. Prototypen er implementert ved hjelp av ulike tilnærminger til deteksjon av ny informasjon, og sammenligner disse.

Preface

This thesis is written by Ellen Wiig Andresen, and is the result of a master's thesis at the Norwegian University of Science and Technology. The master's thesis is the conclusion of the five-year study in Computer Science at the Department of Computer and Information Science.

I would like to thank my supervisor, Kjetil Nørvåg, for being highly available and helpful, and giving valued feedback.

Special thanks are given to Marius Qvam Wollamo and Simen Kind Gulbrandsen for the time they have spent helping me debugging during stressing times.

Finally, I would like to thank my boyfriend, Stefan Hjørnevåg Karlsen for support, understanding, and fruitful discussions about scientific methodologies.

Trondheim, 7th of June 2013

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Motivation | 1 |
| 1.2 | Problem Definition | 2 |
| 1.3 | Goals and Contributions | 3 |
| 1.4 | Report Outline | 3 |
| 2 | Background | 5 |
| 2.1 | Knowledge Bases | 5 |
| 2.2 | Novelty detection | 7 |
| 2.3 | The Text REtrieval Conference - TREC | 7 |
| 2.3.1 | Knowledge Base Acceleration Track | 8 |
| 2.3.2 | Novelty Track | 8 |
| 2.4 | Novelty in KBA | 9 |
| 2.5 | Related Work | 12 |
| 2.5.1 | Novelty detection | 12 |
| 2.5.2 | Text Reuse | 12 |
| 2.5.3 | Plagiarism | 13 |
| 2.5.4 | Topic Detection and Tracking | 13 |
| 3 | Techniques Used In Novelty Detection | 15 |
| 3.1 | Preprocessing | 15 |
| 3.2 | Text units | 15 |
| 3.3 | Document representation | 17 |
| 3.3.1 | Term appearance | 17 |
| 3.3.2 | Weighted vector in Euclidean Space | 17 |
| 3.3.3 | Fingerprints | 17 |
| 3.4 | Similarity computation | 18 |
| 3.4.1 | Term overlap | 18 |
| 3.4.2 | Cosine similarity | 19 |
| 4 | Data set and annotation | 21 |
| 4.1 | Wikipedia Topics | 21 |
| 4.2 | Stream Corpus | 21 |
| 4.3 | Defining Novelty | 22 |
| 4.4 | Annotating the Corpus | 23 |
| 4.5 | Quality of Data Set | 25 |
| 5 | Approach | 27 |
| 5.1 | Preprocessing | 28 |
| 5.2 | Document representation and comparison methods | 29 |
| 5.2.1 | Similarity Values | 29 |
| 5.2.2 | Combinations of Similarity Measures and Document Representations | 29 |

| | | |
|----------|---------------------------------------|-----------|
| 5.2.3 | Partial Documents | 29 |
| 5.2.4 | N-grams | 31 |
| 5.3 | Output | 31 |
| 5.3.1 | Weighting of similarity values | 31 |
| 5.3.2 | Thresholds for similarity values | 32 |
| 6 | Experimental Setup and Results | 33 |
| 6.1 | Evaluation Metrics | 33 |
| 6.2 | Perspectives of Results | 34 |
| 6.3 | Baselines | 34 |
| 6.4 | Thresholds and Weights | 35 |
| 6.5 | Simple Text Similarity | 36 |
| 6.5.1 | Jaccard | 36 |
| 6.5.2 | Cosine Similarity | 37 |
| 6.5.3 | Containment | 37 |
| 6.6 | Using Partial Documents | 37 |
| 6.6.1 | Jaccard | 37 |
| 6.6.2 | Cosine | 39 |
| 6.6.3 | Containment | 39 |
| 6.7 | Using n -grams | 39 |
| 6.7.1 | Jaccard | 39 |
| 6.7.2 | Cosine | 40 |
| 6.7.3 | Containment | 40 |
| 6.8 | Observations Summary | 40 |
| 7 | Conclusion | 43 |
| 7.1 | Further work | 43 |
| A | Graphs | 47 |

Chapter 1

Introduction

This Chapter will give the introduction to the project. Firstly, the motivation for the project, and the reason for attempting to solve the task at hand is given. Then a detailed description of the tasks that has been solved in this project is given. After this, the project's contributions to the field is listed, along with the research questions to be answered in this report. At the end of this chapter, the outline of the rest of the report is given.

1.1 Motivation

The World Wide Web provides everyone connected to it with endless access to information, having the amount of web pages double every nine to twelve months[1]. However, having access to information does not mean that it is easily usable. Search engines like Yahoo and Google provide the users with the ability to find documents on the web that might satisfy the user's information need, based on a query the user give to the engine. Search engines are very useful tools in making the information available, but not always sufficient to make the it easy to find, or to use in a proper way, considering the quality of the information and source quality. In an effort to make all this information more accessible, more structured, and thus more easy to use, knowledge bases were created. Knowledge bases provide information on almost everything, and in enormous quantum. Some knowledge bases are restricted to enlighten people about a specific topic, such as IMDb¹ which is specialized in movies and series, MusicBrainz², specialized in music, or GeoNames³ that contain maps and information connected to place names. Others are more all-covering, not discriminating on language or field, such as Wikipedia⁴.

From time to time, the world changes. This mostly happens in small steps, by events that in some degree affect different parts of what we know. When such an event that relates to something that is represented by an article in a knowledge base, one of two things may happen: 1) The information in the original entity becomes outdated, for example because new studies show that what we once knew is no longer the truth, new discoveries or definitions being changed, such as when Pluto was demoted to dwarf planet, or 2) There is an extension to the information, for example, if a scientist discovers a new star, it must be added to the list of known stars in order for the list to present all known information. In both these cases, someone has to maintain the knowledge base to keep it up to date, either by changing the content of the knowledge base entity or by adding the new piece of information to what is already there.

¹IMDb.com

²MusicBrainz.org

³GeoNames.org

⁴www.wikipedia.org

Knowledge bases are mostly edited by the user community, which makes it possible to maintain the enormous number and variety of articles without employing experts on every field that is covered by the knowledge base articles. This is reflected in the size of some user maintained knowledge bases, such as Wikipedia. As of September 2012, Wikipedia had more than 23 million articles in 285 languages, had approximately 365 million readers worldwide, and about 100 000 active contributors who update the information in the articles. Looking at these numbers, there are something that is worth noting: Only one person out of 3650 users actively help to keep the knowledge base up to date, giving the active contributors close to 230 articles each to keep. In addition, it is safe to assume that not all contributors write in all the 285 languages, making the amount of work done by some contributors even more impressive.

Making complete and up-to-date information available to people is in everybody's interest. But since not all people use the same source to finding the information, the interest of keeping a specific source up to date is in the greatest interest of the users of said source. Given the ratio between users, articles and contributors given above, it is clear that not all information will be up to date at any point of time. The number of necessary updates are always much higher than the number of people performing the updates. Because of this mismatch, the contributors will never be able to keep up with all the changes, and thus articles stand in danger of not providing the reader with correct information. A solution to this problem might be to make those who contribute more efficient, making their work easier, and helping them find the newest information. Making contributing easier might also make users of the knowledge bases more interested in contributing. Knowledge base acceleration aims to do this.

1.2 Problem Definition

This thesis was inspired by the two TREC tasks, Novelty and Knowledge Base Acceleration, KBA, described in Sections 2.3.2 and 2.3.2. The first aims to find a way to detect new information in a stream of documents, returning the sentences that contain novel information. In knowledge base acceleration, the goal is to make the extension of knowledge bases more efficient, making it possible to keep them up to date. The overall goal of this thesis is to present an usable prototype for doing novelty detection as a part of a knowledge base acceleration system. Chapter 2 will give more information about the background for the thesis.

While the TREC tracks present tasks to be solved, this thesis will also discuss an important property of a KBA system, namely what it should return, and how the output should be presented to the users. From this evaluation, a prototype will be implemented for doing novelty detection as a part of a knowledge base acceleration system. The prototype will be used to test several possible approaches to the novelty detection, and compare how well each of these work for the specific problem described in this chapter. The problem the prototype is to solve may be broken down to the following sub-tasks:

1. Representing the content of a document
2. Comparing two documents
3. Based on the comparison done, decide if a document present new information.

For each of the sub-tasks above, the methods used in the prototype will be evaluated, discussing the properties of the approach, and considering how well it is suited for the task. All methods will be compared, and evaluated to see which of them are suitable to combine. At the end, a comparison of the performance of all the methods and combinations are given, and a discussion of the results.

In this thesis, the data stream corpus from the Knowledge Base Acceleration track has been used. The corpus is described in Section 4.2. The dataset was originally only labelled with respect to the

relevance of each stream item to each of the entity topics used. To be able to evaluate the performance of the system, a part of the corpus was relabelled with respect to novelty relative to each topic.

1.3 Goals and Contributions

Derived from the problem definition, this thesis has the following goals:

- Examine properties of a KBA system, and discuss how these should be considered in an implementation of such a system.
- Look at different ways of representing a text in a system, and evaluate these given the perspective of the thesis.
- Given the document representations obtained at the previous point, examine the possibilities for comparing the texts.
- Explore different possibilities for making a decision about novelty based on similarity values.

To reach these goals, a prototype was implemented, and a set of experiments were set up to compare different possibilities in the solution. Each of the methods used in the implementation is described in detail and evaluated with respect to novelty detection. Where the methods have been used in combination with each other, the combination have been discussed, and elsewhere, the results using different methods have been compared.

To be able to evaluate the methods that were used, a test set was created. The data set used was presented by the Text REtrieval Conference in 2012 for the Knowledge Base Acceleration track. The data set was previously only annotated with relevance tags, so novelty tags were added. To make the annotation process easier, a program was created to take the data set from a searchable index to a set of easily readable XML files.

1.4 Report Outline

Chapter 2 gives an overview of the background for this project, and explains some central concepts, such as knowledge bases and novelty. It also gives an overview of related work.

Chapter 3 go through different techniques used in novelty detection that were used in the implemented prototype, including methods for finding the similarity between two documents, and different ways to represent the information in a document.

Chapter 4 describe the data set used in the thesis, with details about the different parts of it. In addition, information about how the data set was handled before it was used to test the implemented system is given here.

Chapter 5 give a detailed description of the implemented prototype for novelty detection, describing the data flow in the system, and how the different methods described in Chapter 3 were used in the implementation, in addition to a description of the experiments that were done.

Chapter 6 present the results from the experiments described in Chapter 5, along with an interpretation and evaluation of the results.

Chapter 7 gives a conclusion and summary of the entire project, the implementation, and the results achieved. Then a discussion of the results is given, and derived from this, suggestions for further work and research are given.

Chapter 2

Background

This chapter explain the problem addressed in this thesis in more derail, giving an introduction to relevant background information and definition of terms, including a short introduction to Knowledge Bases and the Text REtrieval Conference. In addition, the Knowledge Base Acceleration and Novelty tracks of the Text REtrieval Conference are introduced. At the end of the chapter, an overview of related work and fields are given.

2.1 Knowledge Bases

A knowledge base, sometimes shortened to KB, is as the name suggest, a data base for storing knowledge. The knowledge stored in these collections may be facts, user guides, FAQs, articles, lists, or any other piece of information the managers of each base might believe to be useful to its users. To make the information more easily accessible to the users, the KB often supports searching, and provide different types of categorization of information.

A definition of knowledge bases is given by Wikipedia¹ as follows:

*A knowledge base is an information repository that provides a means for information to be collected, organized, shared, searched and utilized. It can be either machine-readable or intended for human use.*²

As the definition states, there are generally two types of knowledge bases. The first is the ones that are meant for humans to use directly. They are designed for browsing, searching and reading by human users. The most widely known human-readable knowledge base is Wikipedia. Wikipedia also has a machine-readable counterpart, DBpedia³. DBpedia contain the same information as Wikipedia, but is designed as an interface for machines to do queries against. Other types of machine readable knowledge bases may be used for storing knowledge and behaviour for an AI agent. These KBs may have a conditional structure, where the information are stored as if-then assertions, such as in [2].

An example in how a machine/readable knowledge base may be used can be found in [3], where they use Wikipedia to detect named entities in text, and to disambiguate them. Another example can be seen in [4]. Here, they use the first sentence in Wikipedia's entities to create a definition of named entities.


¹www.wikipedia.org

²en.wikipedia.org/wiki/Knowledge_base

³www.DBpedia.org

Trondheim

From Wikipedia, the free encyclopedia

Coordinates:  63°25′47″N 10°23′36″E﻿ / ﻿63.42972°N 10.39333°E﻿ / 63.42972; 10.39333

For other uses, see [Trondheim \(disambiguation\)](#).

Trondheim (Norwegian pronunciation: [ˈtʀɔŋjæm]), historically, **Nidaros** and **Trondhjem**, is a city and [municipality](#) in [Sør-Trøndelag](#) county, [Norway](#). With a population of 176,348, it is the third most populous municipality in Norway and city in the country, although the [fourth largest urban area](#). It is the administrative centre of [Sør-Trøndelag](#) county. Trondheim lies on the south shore of the [Trondheimsfjord](#) at the mouth of the river [Nidelva](#). The city is dominated by the [Norwegian University of Science and Technology](#) (NTNU), [SINTEF](#), [St. Olavs University Hospital](#) and other technology-oriented institutions.

The settlement was founded in 997 as a trading post, and was the capital of Norway during the [Viking Age](#) until 1217. From 1152 to 1537, the city was the seat of the [Archdiocese of Nidaros](#); since it has remained the seat of the [Diocese of Nidaros](#) and the [Nidaros Cathedral](#). It was incorporated in 1838. The current municipality dates from 1964, when Trondheim merged with [Byneset](#), [Leinstrand](#), [Strinda](#) and [Tiller](#).

Contents [\[show\]](#)

History

[\[edit\]](#)

For the ecclesiastical history, see [Archiepiscopate of Nidaros](#)

Trondheim was named [Kaupangen](#) (English: market place or trading place) by [Viking King Olav Trygvason](#) in 997. Fairly soon, it came to be called [Nidaros](#). In the beginning it was frequently used as a military retainer ([Old Norse](#): "hird"-man) of King Olav. It was frequently used as the seat of the [king](#), and was the capital of Norway until 1217.



Figure 2.1: Wikipedia article about Trondheim. The entity representing the town of Trondheim has relations to for example the entities representing the country Norway, the river Nidelva and the university NTNU. Relations are represented with links to the related entities, in the text marked with blue writing.

Knowledge bases used by humans are often organized in an entity-relational model, which is increasingly becoming a more common way to organize data [5]. The entities in a KB represents a topic, for example cities, persons, theories of science, or things. A relation connects two entities. In user-readable KBs, the connection between the two entities must be understood from the text in the entities, making it more difficult to extract information about the relation, more than that it exist between the two entities. Figure 2.1 show the Wikipedia article about Trondheim, that is, the entity representing Trondheim, along with the links to the entities Trondheim has a relation to, marked as blue text.

2.2 Novelty detection

When looking at a collection of information on the Web, it being a news wire or a stream from social media, there is very likely to exist repeated information. Detecting novelty is the task of finding the documents in a stream with a temporal perspective that provide new information given a set of topics, that is, finding the information that is not repeated from earlier in the stream. Repeated information may be different authors reporting on the same event, rewrites of publications as a part of a reference, or information rewritten for a different audience. Some repeated information may be presented identical to the original source, have some minor changes, or be complete differently written, but still containing the same information.

Novelty detection have several applications. One example is in the domain of finance, where one is interested in finding new information as soon as possible to use the content to predict the change in the market before anyone else. In this case, novelty detection would be used on a financial news wire, or several, passing through to the user only the news articles containing new information. New information may be generally in the stream, or concerning a set of companies or industries the user want to monitor, where this set will be seen as the topics used in the detection process. In this case, one might also look for specific words or phrases in articles related to an interesting company, that might be indicators of change.

Another example is plagiarism detection, where the goal is to find if a text contain reused text from an earlier time. Today, many system for plagiarism detection only looks at direct identical pieces text. In some cases this is sufficient, for example in student exercises, where each piece of text answer a short, simple question. However, in for example scientific publications, it might be interesting to see which parts of the paper is presenting unique information, and which is information derived form previous publication. In this case, it is necessary to look at possible rewrites of information.

2.3 The Text REtrieval Conference - TREC

The Text REtrieval Conference[6], TREC for short, is arranged annually by NIST⁴, and aim to make research on information retrieval better, more efficient, and make improvements reach the commercial market faster. To make this happen, they provide large collections of data to do research on, and provide evaluation techniques appropriate to different areas of research within information retrieval. In addition to providing the IR society with these tools, the TREC is a communication link between academics and the industry.

Every year, TREC presents a set of tracks that will be the focus for the coming year. Each track is defined by one or more tasks to be solved, often making more information about the data used in the track available for each task. Tracks may be reused, most often by making additions or alternations to the track description or the track tasks, over several years.

⁴National Institute of Standards and Technology, www.nist.gov/

Two tracks are presented in the following sections, Knowledge Base Acceleration and Novelty. The Novelty track was used in 2002 through 2004, and the Knowledge Base Acceleration track was new in 2012, and reused in 2013. This project aim to present an extension to the later track by adding a component for novelty detection. In this thesis, the corpus presented by TREC for the Knowledge Base Acceleration track was used, since the focus has been on knowledge bases. The corpus, and how it was adapted to fit this thesis, is described in Section 4.2.

2.3.1 Knowledge Base Acceleration Track

Knowledge base acceleration, KBA, aims to make the extension and updating of knowledge base articles more effective, making it possible for the same number of people to spend the same amount of time performing the updates, but covering a much larger amount of articles than today. The goal is to increase the amount of articles that are up to date, by making the job easier for the content managers. Hopefully, this will also help making more people interested in contributing.

The Knowledge Base Acceleration track was introduced in 2012, and reused in 2013. In 2012 the track only had one simple task: *filter a stream for documents relevant to a set of entities*. The initial goal of the KBA track was to make the extension of existing knowledge base entities more efficient, by suggesting articles from a stream that provide relevant information about the entity. Relevant articles may either be presented to human editors as documents containing information relevant as an extension to the information in the entity, or as a possible new reference to make the information in the entity more solid, if the source of the information is reliable.

TREC provided a corpus of test data which consists of news articles, collected from public news, and articles from social media, collected from for example blogs and forums, and links published in posts in social media. The corpus consists of data dated from October 2011 through April 2012. The data corpus is described in more detail in Section 4.2.

Novelty detection has not been a part of the KBA track. The only objective has been to filter out the relevant documents to each topic from the stream, not considering if the information is new or old. This results in a KBA system that might return the same information as relevant several times, thus returning more information than necessary, transferring work to the user of the KBA system.

2.3.2 Novelty Track

The Novelty track was first presented in 2002, and was reused in 2003 and 2004. The data used in the 2004 track consisted of 25 event and 25 opinion topics and a set of Documents. Each topic had at least 25 relevant sentences, and from zero and up irrelevant sentences. For each document, the tasks were to find the sentences that were relevant to the topic, and then determine whether or not the sentence provided new information, that is, detect novelty.

The track presented four tasks, in which the goal was to identify the relevant and novel sentences in the documents in the provided data corpus. For each of these tasks, the participants were provided with more information of relevance and novelty in the data set, giving the following definition of the tasks:

1. Identify all relevant and all novel sentences, where no information about relevance or novelty is known.
2. Identify all novel sentences, with all relevant sentences known.
3. Identify all relevant and all novel sentences, given that information of novelty and relevance in the five first documents is known.

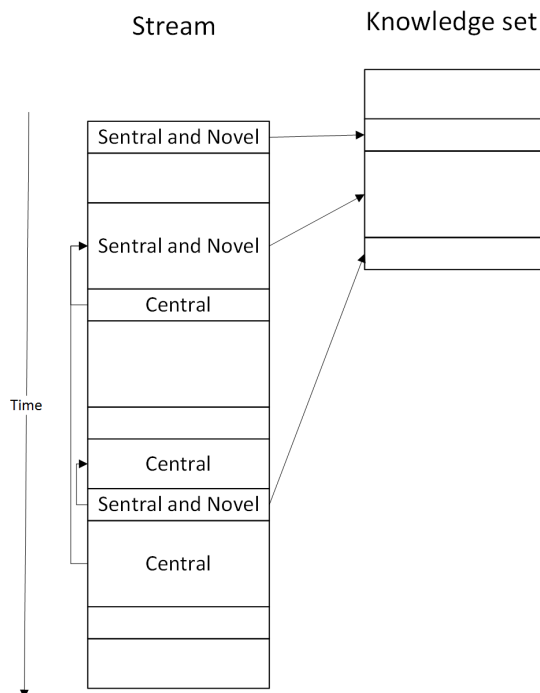


Figure 2.2: Novelty in stream. Each chunk represent a piece of information, in the stream, this will be an article. Information may either be central to the entity, central and contain novel information, or neither. Only novel information should be suggested as an extension to the KB article, and is added to the *knowledge set*. Documents that are central, but not novel may present information that was novel earlier in the stream.

4. Identify all novel sentences given relevance information about all sentences, and novelty information about the five first documents.

An overview of the entire Novelty track may be found in [7], including a temporal perspective, giving the development of the track through the tree years it ran. It points out some of the difficulties found during the track. One of these is that it proves difficult to spot relevant sentences, as they contain very little information, compared to the whole document in which they are a part of.

In the Novelty track, *new information* was defined as *new in this data set*. Which mean that the system does not consider what one might know about the topic initially. In a knowledge base acceleration perspective, it would be very valuable to consider the information present in the knowledge base entity at the beginning of the stream of news, since the output of the system should tell a KB user if each news article should be included in the KB entity.

2.4 Novelty in KBA

Considering a solution to the 2012 KBA track, it would result in a system returning a set of relevant information for each specific topic. The goal is to make human curators more efficient in collecting information by automatically linking relevant documents, such as news articles, to a knowledge base entity. This would be very useful, as the information presented is structured by which documents are relevant to each entity, and the amount of information to be considered by a human thus is of a much smaller volume. If the output of such a system is presented in a way that make it attractive to use, a KBA system might also contribute to make more people interested in actively seek out KB entities that need to be updated. However, much of the information will still be irrelevant for the extension of

the entity, given the description given to the 2012 track, as much of the information in the documents may already be present in the entity.

By sorting out which of the documents considered relevant to an entity contain *new* information, the data presented will be easier to use, since some of the surplus information is removed. There are two ways to present the data to be returned. One is to present the entire documents containing new information. The other is to present only the parts of the documents that present new information, it being sentences, as was done in the TREC Novelty track[7], paragraphs or other chunks of information. This report propose a solution returning entire documents. This thesis uses smaller pieces of text to decide if a document contain new information, but only returns whole documents. This is described further in Chapter 5 An overview of sentence level novelty detection is given in [8].

In the Novelty track, one considered only what one had learned from the stream, not what one might know about the topic before the stream started. When combined with KBA, one should consider all that is known about an entity, including the content of the KB article at the beginning of the news stream considered. Everything the system know about an entity will be collected in a *knowledge set*. This set would therefore initially only contain the information that is found in the KB article at the beginning of the stream, and as the system find new information, it would be added to this set. This knowledge set also contain all the information that is returned to the user, as a list of suggested relevant updates.

Figure 2.2 show a stream of documents, and a knowledge base article. The first piece of information in the knowledge set is the KB article as it was when the stream started. Some articles in the stream are central, marked as *C* in the figure, to the topic, and some are both central and presents novel, *N*, information. If a document is central, but not novel, the information presented by the document is either already in the KB article, or has been presented by a previous document in the stream, and has therefore previously been added to the knowledge set. Information that is already seen in the stream is shown by arrows pointing backwards in time. Only articles that are both central and novel should be added to the knowledge set.

There are two cases triggering use of the KBA system. These are illustrated in Figure 2.3. One is when a new article appear in a news stream that the system collect information from. When this happen, the system have to make a decision about if the document contain new information or not. This is done by first determining if it contain central information to any entity in the knowledge base. Then, based on the knowledge set for that entity, novelty is looked for. Articles containing novel information as added to the knowledge set.

The other case is when an user visits an entity's web page in a knowledge base using a KBA system. In this case, the knowledge base would check the knowledge set for new suggested updates. If any updates are found, the knowledge base would prompt the user that the entity may be out of date, and that there exist suggested updates. Should the user choose to update the entity, he is also given the choice to remove the updates he has done from the suggestions list.

Figure 2.4 show novelty detection as a part of a KBA system. The system implemented in this project will only cover the novelty detection component, the shaded component in the figure, and it is assumed that centrality detection is already implemented. As the system depends on an input containing only central documents, annotated documents from the KBA track will be used. The output of the novelty component should be the same as for the overall system, a judgement of the newest article in the input stream, either labelling it as central and novel, or discarding it as unrelated or repetitive information. The articles that are considered novel should be put in the knowledge set that is presented to the user, the knowledge base content manager.

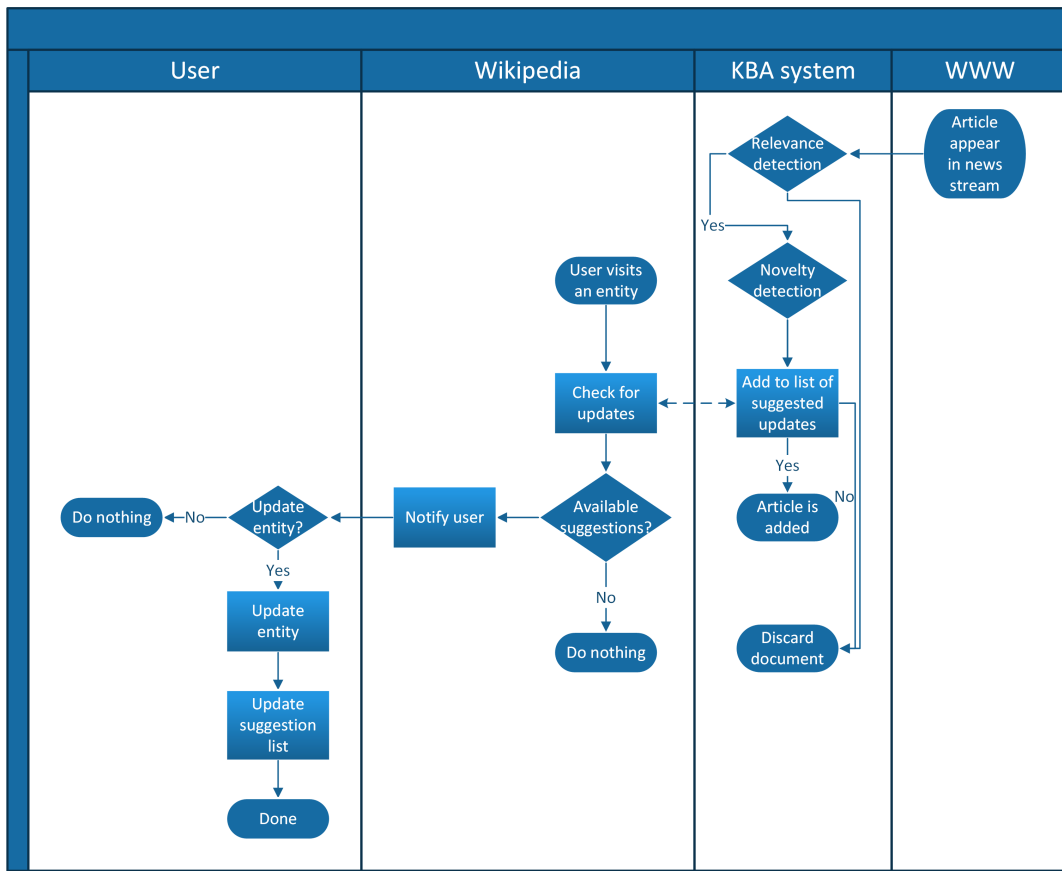


Figure 2.3: Flow chart showing the use of a knowledge base acceleration system.

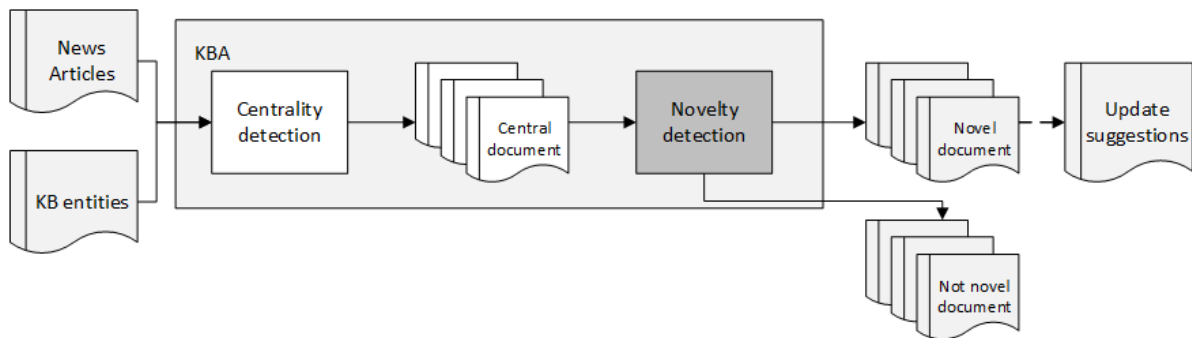


Figure 2.4: Overview of a KBA system. This project has focus on Novelty detection, shaded in grey, as a part of a KBA system. It is assumed that the Centrality detection is fully functioning, and thus, that the input for the Novelty detection component is only consisting of documents central to the input topic.

2.5 Related Work

Novelty detection has several applications, and several fields that have similar goals and approaches. Some of these fields will be looked at in the remainder of this chapter, along with some previous approaches to novelty detection.

An overview of the TREC Novelty track may be found in [7], covering all years, 2002 - 2004. It both present the different tasks used in the track, and give a summary of the methods and results from the track. In this summary, it is pointed out that in solving the novelty detection task, most get a good *precision* value, while the *recall* is significantly lower. Most approaches measured novelty in each sentence as the dissimilarity to the sentences previously considered relevant and novel.

Information filtering is a wider field than novelty detection, ranging from filtering of search results to filtering personal e-mail based on content. The filter may be defined by a set of *profile queries* that define the information need of the user. An overview of different types of information filtering system is given in [9], along with an overview of the development in the area. Novelty detection may be seen as a subtype of filtering, where the information need is defined from the topics that are used.

Another approach that may be used in novelty detection is looking at *information patterns*. The user's need is transformed into questions for identifying information patterns consisting of queries and corresponding answer types. This is looked at in [10] and [11]. [10] uses indexing-trees, where each node in the tree represents an event. If a story is considered a new event, a new node is created for it. [11] uses a two-step approach. The first step, *query analysis*, the information need of the user is represented as one or more questions, with corresponding required answer forms. The second step, *new pattern detection*, answers to the questions are retrieved. Sentences that indicate *new* answers are marked as novel.

2.5.1 Novelty detection

Novelty detection as a part of an information filtering system was examined in [12]. They conclude that cosine similarity is effective in detecting redundant information in a stream of documents which is the opposite of what is done in this thesis. While they look for repetitive information, and filter this out, this thesis look for new information, and filter that out. The goal is the same, only the perspective is different. More about these two perspective is given in Section 6.3.

An alternative to the pre-defined topics used in this thesis may be found in [13] where the topics used in novelty detection were defined from the stream in a preliminary step to detecting novelty. To define similarity between documents, cosine similarity was used.

Using the data set and procedure from task 2 in the Novelty track in 2004, [14] make term distance graphs from sentences, and uses these to find mutual information in two sentences. In addition, they have extracted feature sets from the graphs, which were used in the novelty detection.

2.5.2 Text Reuse

Finding reused text is the opposite of finding novel information from text. The difference lie in how you filter the results. If you discard the information that is reused, you do novelty detection, but if you discard the novel information you do text reuse detection.

Text reuse detection has several applications, for example, [15] look at the issue of finding the original source of new information on the web, it being facts, rumours or other information, for use in web search. They use three different approaches to retrieve reused text, word overlap, query likelihood, a mixture of these two, and dependence models.

Another example of text reuse may be found in [16], where they look at text reuse in smaller parts of documents, as was done in the Novelty track. Here they have used different fingerprinting techniques in addition to looking at term overlap.

2.5.3 Plagiarism

Detecting plagiarism is the task of finding not only duplicate information, but also duplicate text. This field has been looked at by many, for example [17]. Here, they look at detecting complete and partial copies in a digital library, where plagiarism detection is necessary to protect intellectual property, as a step in detecting uncredited reuse. They argue that unauthorised and uncredited reuse of intellectual content may prevent people from wanting to share their research. They present a system for detecting full or partial overlap.

Three issues of plagiarism related reuse of text is explored in [18]. They look at cross language reuse by looking for exact translations, which kinds of re-phrasing is the most common, and reuse in Wikipedia, both within the same language, and between different languages. These aspects are similar to some of those discussed in this thesis.

2.5.4 Topic Detection and Tracking

Topic detection and Tracking(TDT) refers to the detection of topics in news stories, and tracking re-occurrences of these later in a stream of news items. The when a topic is detected after the first time, the information in the news story should be used to develop the topic, making it represent what we have seen about it from the beginning. [19] give a state of the art overview of the Topic Detection and Tracking study. They have generalized *topic* to be events.

In this project, the corpus is labelled by in which degree a news story is related to each topic, as further described in Section4.4. Hence, when looking at a news story, it is already known if it is related to a given topic. The problem to be solved is to find out if a story present *new* information, that is, if the story connects the topic to a new event.

An approach to topic detection and tracking using simple semantics is resented in [20], which is used to group terms based on their meaning, and thus being closely related to novelty detection.

Chapter 3

Techniques Used In Novelty Detection

This chapter will explain in detail some of the techniques used in the different parts of novelty detection, from text preprocessing and text representation to similarity computation. The techniques described are the ones used in the implemented system, further explained in Chapter 5.

3.1 Preprocessing

Preprocessing includes all that has been done to all the different parts of the input before they are looked at as a whole. There are a some preprocessing steps that are very common. The first thing that one might do, is to split a document into terms. If this is done by splitting a string by space, some of the terms might contain non alphanumeric characters. These is hard to extract information from, so they are typically removed, turning “it’s” into “its”, and “(The” into “The”.

An other common step is to set all letters to lower case. This is done to make sure that two identical words is not overlooked because one is in the beginning of a sentence, and one is mid-sentence. On another hand, this might remove information. If the meaning of the terms is looked at, a capital letter in the beginning of a word might indicate a name or a title, for example "Potter" versus "potter".

Table 3.1 give the most common words in the English language, provided by the Oxford English Dictionary¹. This list may be used as a list of stop words to be removed. This is done because the presence of very common words do not tell much of the content of a text. However, removing stop words may present a problem as some phrases may lose their meaning without the stop words, for example “flight to London”. If it is important that the flight is *to* London, removing stop words will remove the meaning of the phrase.

Most words have several different written forms. Most of these are connected to different times, or singular and plural forms. When matching two documents, it might be useful to reduce these to a basic form, making it possible to match a word written in two different forms. This is called *stemming*. Stemming has the same disadvantages as removing stop words.

3.2 Text units

A text unit is the smallest unit to look at when comparing two documents. In some methods for similarity measuring, single terms are the text units used in the comparison, and in others, n-grams are used. An n-gram is a sequence of n units that appear consecutive in a document. In this case, an

¹<http://oxforddictionaries.com/words/the-oc-facts-about-the-language>

| |
|---|
| 40 most common words |
| the, be, to, of, and, a, in, that, have, i, it, for, not, on, with, he, as, you, do, at, this, but, his, by, from, they, we, say, her, she, or, an, will, my, one, all, would, there, their, what |

Table 3.1: The stop words used are the 40 most common words in the English language

| The bird ate the eight worms | | | | |
|------------------------------|-------------|-----------------|---------------------|--------------------------|
| 1-grams | 2-grams | 3-grams | 4-grams | 5-grams |
| the | the bird | the bird ate | the bird ate the | the bird ate the eight |
| bird | bird ate | bird ate the | bird ate the eight | bird ate the eight worms |
| ate | ate the | ate the eight | ate the eight worms | |
| the | the eight | the eight worms | | |
| eight | eight worms | | | |
| worms | | | | |

Table 3.2: Sets of different n -grams of the string *The bird ate the eight worms*. The string consists of 6 terms, so the maximal n is 6. There is only one 6-gram for the string, and that is the string as it is.

unit may either be a character, a term, or what ever else it is defined to be. When n -gram units are terms, an 1-gram will be the same as a single term.

When working with n -grams, there are a few things to consider. Firstly, how large should n be? A bigger n make the n -grams more descriptive, containing more information about the content of the text being considered, making a hit a better indication of similarity between two documents. However, a big n will make it less likely to find a matching n -gram in an other text. This may cause that similar documents are not found, as a simple rewrite may cause the n -gram to not appear in the second text, even if they are almost identical in content. One way to solve this is to in addition to consider all the n -grams, consider all $(n-i)$ -grams, where i range from 1 to $n-1$. Doing this remove the problem with possibly overlooking documents that are slightly rewritten, but at the cost of considerably extra computing.

A second thing to consider is what to do with n -grams appearing more than once in a text. An example is the 1-gram "the" from the example in Table 3.2, which appear twice. One alternative is saving each n -gram as i , n -gram, where i is the number of times the n -gram appear in the text. Alternatively, the n -gram can be saved one time for each time it appear. Using the same example, with the 1-gram "the", the two alternatives are $\{2, the\}$ or $\{the\}\{the\}$. The first one require less space, but need more work to read.

Among existing system using n -grams, is The MEasuringText Reuse system, METER [21], which measure text similarity by finding term overlap in n -grams, in addition to finding the maximal length of common substrings in the two texts being compared. The system has as goal to identify reuse of text, with focus on news.

In [22], n -grams were used in text categorization. They look at the frequency of n -grams, and thus find which segments of n characters is common in a text. This is used to put the text into a category based on n -gram frequency profiles for different categories.

3.3 Document representation

Different approaches to measuring similarity between documents require different representations of the documents to be compared. Some representations require knowledge about the entire known document collection, making them more expensive to compute, while others only need consider the two documents being compared at the time. These issues are discussed along with the description of the different representations below.

3.3.1 Term appearance

The simplest document representation used in this thesis is the term appearance vector V_{TA} . This vector contain information about which terms in a collection appear in a document, and which do not. In this case, the collection consists of only the two documents being compared, and the length $|V_{TA}|$ is the total number of terms in the two documents. For each document, a vector is created, using the following formula

$$V_{TAi} = \begin{cases} 1, & \text{term appear in current document} \\ 0, & \text{else} \end{cases}, \quad (3.1)$$

where each value i represents the same term in both documents. Since the size of the vectors is dependent on both the documents to be considered, the document vector for each document have to be recreated each time it is to be compared to a new document.

3.3.2 Weighted vector in Euclidean Space

The document representation given above does not consider how often a term appear in the document, or if a term that exist a very common one, giving common words that have little information about the content of a document the same value as a rarer term that might be important to the content. A document representation vector that take this into consideration is the weighted vector V_W . V_W is a vector in euclidean space with n dimensions, where n is the number of terms existing in the document collection. n is also the same as $|V_{TA}'|$. The collection in this case is all documents previously having been labelled as *novel*, plus the document being considered and the *KB* article. The weight measurement used is $tf - idf$, where each value in V_{TA} is given by

$$tf - idf_{t,d} = tf_{t,d} \times idf_t = tf_{t,d} \times \log \frac{N}{df_t}. \quad (3.2)$$

$tf_{t,d}$ give how many times the term t appears in document d , N is the total number of documents in the collection, and df_t is the number of documents where t appear in the collection. This means that N changes for each new article appearing in the stream, and is different for each entity. Following this, V_W must be recreated for each document each time a new central document appear in the stream. As the collection grows, this may become very expensive.

3.3.3 Fingerprints

V_{TA} is cheap to compute, but must be recomputed for each document each time it is compared to a document, and does not take into consideration the importance of the terms. V_W consider term importance, but is more expensive to compute, and also have to be computed for every document every time a new document is considered.

A technique where each document may be presented as a set of values, that is permanent for the document, regardless of the collection it is a part of, is fingerprinting. The word fingerprinting refers to the process of generating fingerprints for documents. Where the values in V_{TA} is dependent on placement to know which value represent which unit of text, a fingerprint is a hash value, that is unique for that specific unit, regardless of the rest of the collection. Due to this property, fingerprints can not only be used to represent a document in a fashion usable for similarity computations, but also to represent the entire document. Doing this, it is no longer necessary to read and process the documents every time they are to be used. It is enough to get the fingerprints.

When fingerprinting, one have to choose how much text a fingerprint is to represent, and how much of a document that is to be represented through the fingerprints. Choosing too large chunks of text for each fingerprint rises the requirement for two documents to be similar, as it is less likely to find a large chunk of identical text, than to find smaller, given that the two documents to be compared are not close to identical. Choosing small chunks causes more fingerprints, and make the comparison process more expensive. It is also an option to only store samples of documents, in stead of fingerprinting the entire document. This also make the process cheaper, but at risk of not detecting similar documents as a result of the information loss.

Fingerprinting

The detection of near duplicate documents was looked at in [23]. They approached the issue by creating a light-weight vector for each document, and comparing the document vectors in stead of the entire documents. The solution was used in the AltaVista search engine to prevent the returned document set from a query contain duplicates. They used Rabin fingerprints[24], using a hash function for identifying units. Rabin fingerprints was also used in [16] to identify text reuse and near-duplicates.

3.4 Similarity computation

When having documents represented in different manners, presenting different information about the content in the documents, options opens for different ways to compare them. Comparison measures used in the implementation is described below.

3.4.1 Term overlap

Term overlap aim to measure the amount of terms that are shared by two term sets. The Jaccard coefficient is one of the most popular measures for computing the similarity between two documents, due to it's simplicity. Shortly stated, it measures the overlap of two documents. Given two term sets A and B , the Jaccard coefficient is given by

$$J_{A,B} = \frac{|A \cap B|}{|A \cup B|} = \frac{\sum A_i \cap B_i}{\sum A_i \cup B_i} \quad (3.3)$$

Where A and B are the two documents being compared, and A_i and B_i is the boolean value indicating presence or absence of term i in A and B . Thus, the numerator is the number of terms occurring in both documents, and the denominator is the number of terms occurring in one or both documents. [15], [16] and [21] uses a similar comparison function, given as

$$C_{q,r} = \frac{|q \cap r|}{|q|} \quad (3.4)$$

where the numerator is the same as for Jaccard, but the denominator is the count of terms in q . The big difference between Jaccard and this measure, is that the latter is non-symmetric, that is, $C_{q,r} \neq C_{r,q}$

The mathematical properties of the measurements given above are discussed in [25]. They use 3.3 to define resemblance of two documents, and 3.4 is used to define containment of a document in another. They use Rabin fingerprinting to represent their documents.

3.4.2 Cosine similarity

If two documents are represented by a vector consisting of the same number n of values, each between 0 and 1, *cosine similarity* may be used to measure the similarity between the two documents, by computing the cosine value of the angles between the two vectors, if they are put in a n -dimensional euclidean space. Cosine similarity is calculated by

$$Sim_{Cosine}(A,B) = \frac{\sum A_i \cdot B_i}{\sqrt{\sum A_i^2} \cdot \sqrt{\sum B_i^2}}, \quad (3.5)$$

where A_i and B_i are the i th term in the term vectors representing the documents. 3.2 satisfy the requirements for computing cosine similarity, and is therefore used as representation for this similarity computation.

Chapter 4

Data set and annotation

This chapter describes the data used in this thesis, including the Wikipedia entities used and the stream corpus containing the news articles used. It also give a description of how the data was processed before it was given as input to the implemented system. The data set used was presented by TREC to the KBA track in 2012. Providing a full data set for the participants makes the comparison of the solutions much easier, and gives all participants the same basis for solving the task.

4.1 Wikipedia Topics

Wikipedia entities are uniquely identified by the title of the article about the entity. In the case where a topic might mean different things, the topic itself refers to an entity listing the different meanings of the topic, for example "Rock"¹, where "Rock music" and "Rock (geology)" are examples of entity titles of entities about different meanings of "Rock".

The 2012 TREC KBA data corpus included a set of 29 Wikipedia entity titles identifying the entities to expand in the track. Most of these entities represent persons with different backgrounds, some political, and some from popular culture. The entities were chosen with focus on the complexity of their linking graphs to other active entities, making them less likely to be slumbering through the entire period of the stream corpus. This was balanced with avoiding topics that too frequently appear in discussions or news on the web, such as active actors, musicians or politicians.

4.2 Stream Corpus

The other part of the 2012 KBA data corpus, was the stream corpus, consisting of the news articles and the documents from social media that is the stream, where each document will be considered by a KBA system to find out if it should be a suggested update to any of the entities. The stream corpus is divided into three parts, depending on the origin of each article:

Social: The social articles are mainly collected from blog posts and forums. These has the advantage of rich metadata, such as tags that describe the content of the article. Many blogs and forums are focused on one or several main topics, this might be included in the metadata, and may be useful in the filtering.

News: The news articles were acquired by getting a set of URLs. These had a time stamp from which it was possible to get the content at the given time. Thus, the news part of the corpus consists of a set

¹en.wikipedia.org/wiki/Rock

JEFFREY BROWN: Now, Jim Steyer, what about this issue of the newer forms of interactive devices? Because I can imagine many parents would say, now, these are useful, right? Kids are learning from a lot of these things.

JAMES STEYER: Well, I think they can be. And I think it depends on the device and the age of child and the choice of content. But I agree with Dr. Brown basically about under the age of 2. There's just no proof that anyone's going to learn anything, and it will be a passive babysitter. That's basically it.

JEFFREY BROWN: All right, so take us up...

JAMES STEYER: As you get older, like 5-8, there's no question. The number-one category of iPhone apps now is for preschoolers. So clearly the market is responding to the idea that you can create educational interactive content. The issue, though, is, is making good choices there, if you're a parent or you're an educator who wants to use it. And so as kids get older, they can be exposed to screen time. They can use these devices in moderation. I think that's what the pediatricians are telling us ...

Table 4.1: Part of interview with Jim Steyer. He makes statements closely related to what he is known for. This is the first time this interview appear in the stream, and it is defiantly central to the topic, but a Wikipedia article does not typically contain all the interviews a person has done.

of global public news from the given period of time.

Links: This part was provided by Bitly². By using approximately 10 000 queries, that is the Wikipedia entity titles used in this project, along with all in and out linking entities, a subset of the URLs shortened during the time of the stream at Bitly was chosen. The web pages of the URLs were used as articles the same way as the two other types if the corpus.

In this thesis, only a part of the corpus have been used, to have a more manageable data set. It included only social and news articles from October to December 2011, labelled according to relevance for each topic. The input of the implemented system was a searchable Lucene³ index covering all these articulated, 21164 articles in total.

4.3 Defining Novelty

Definition: *A novel document is one that presents central information that is not previously seen in the stream, and should be added to the KB article.*

The articles were considered the same way as sentences were in the Novelty track, where novelty was defined as *new in this stream*. They were looked at one at the time, and in chronological order, not letting the system know anything about the articles appearing at a later time. Each article was labelled as novel if it presented any new information given what was seen earlier than the time the article appeared in the stream.

The definition of novelty given above does not consider the perhaps most important article to each entity, that is the Wikipedia article containing the information we had about each entity at the beginning of the stream. Considering the goal of making a KBA system, it become quite useless if it does not consider *all* information we know about an entity at any time of the stream. If the Wikipedia article is left out, potentially a lot of the articles returned by the system contain only information already in the entity. The easiest solution to this issue is to take the Wikipedia article in as the first article, before beginning the reading of the stream.

At this point it is necessary to introduce a new concept. When considering novelty, the *knowledge set* is the set of information that contain everything we know about a given entity. The Wikipedia article

²bitly.com

³<http://lucene.apache.org/>

| | | | | |
|---------------------|----------------|----------------|-----------------|----------------|
| <u>Mentions</u> | A | S | D | F |
| <u>Zero Mention</u> | Z | X | C | V |
| | <u>Garbage</u> | <u>Neutral</u> | <u>Relevant</u> | <u>Central</u> |

Figure 4.1: Relevance categories, as presented by TREC. Only documents in category F were considered in this thesis.

as it is at the beginning of the stream is the original content of the knowledge set of an entity.

In this thesis, the knowledge about the entities at the beginning of the stream has been considered to be non-existent. As a result of this, the first document in the stream central to a given entity will always be novel. This simplification has been due to the manual annotation done on the data set to have a test set, as it would require much more resources to manually annotate the corpus having to first learn all that is already written about the entities at the start of the stream. This is described further in Section 4.4.

4.4 Annotating the Corpus

The data set described in Section 4.2 was initially labelled by their relevance to each topic. Figure 4.1 presents the different classes used to describe this relevance. If a document *mentions* a topic, it is specifically mentioned by name, either partial or full name, or by other identifiers such as title or stage name. If the article only implies a connection to the topic, by metonymic⁴ references or by using synecdoches⁵, it goes under categories *zero mention*. The four categories along the x-axis tells us to which degree the article is relevant to the topic. The yellow categories are the ones that might be interesting in KBA, but only the upper row was used in the 2012 track. Only the articles in category F were considered in this thesis, as only central documents may be defined as novel.

Initially, the articles were only labelled using the categories described above, not considering whether or not the central information is duplicate information. Only articles that *mentions* an entity was labelled in detail, that is looking at the horizontal dimension in Figure 4.1. All articles that fell under the *Zero mention* group of classes were given the same value as class A. The articles in the *mentions* group were given a value indication one of the four categories. This labelling was not usable as a test set to test the implemented novelty detection system on. Therefore, labels for novelty needed to be added. This was done manually by reading each article and then deciding if it presented any new information, and labelling thereafter.

The annotating process was distributed throughout the entire period of this thesis, to be able to test from an early stage, but still have an as complete as possible test set when the final testing of the system were done. At the beginning of the period, the data set was provided as a searchable index, along with a text document giving the annotations for all the documents. Each document is identified by a `stream_id`. The structure of the information in the annotation document, along with examples of all the possible combinations of labels are given in Table 4.2.

Since novelty has been defined to only include articles containing central information, only articles in class F have been considered. A simple system was implemented to get only the central documents, and present them in a human-readable fashion, to make the annotation process easier. This system

⁴Metonymy is when something is called by the name of something one closely associates with it.

⁵A synecdoche is when something is either generalized or when a small part of something is used to name what it is part of.

| stream_id | Entity title | Horizontal class | Vertical class |
|---|---------------|------------------|----------------|
| 1325264488-2d6cd17e4d10708d79e0e49270a010ea | Darren_Rowse | 2 | 1 |
| 1325263651-b5f3a7ebffbafc2e65fbed670defe6c5 | Darren_Rowse | 1 | 1 |
| 1325264488-c1eff2e57b92993763e088750c2eea3 | Darren_Rowse | 0 | 1 |
| 1325305197-cf14a649a5e5b9af851f5b41739d3ee9 | William_Cohen | -1 | 1 |
| 1325244840-60f94da2ca216a7d10e1f2d0d4d08e68 | William_Cohen | -1 | 0 |

Table 4.2: The structure of the annotation text file. A Vertical class value of *1* indicates an article in the *Mentions* group, where *0* indicates *Zero mention*. Horizontal class may have values from *-1* to *2*, where *2* is equivalent with class *F*, and *-1* cover classes *A* and all *Zero mention*

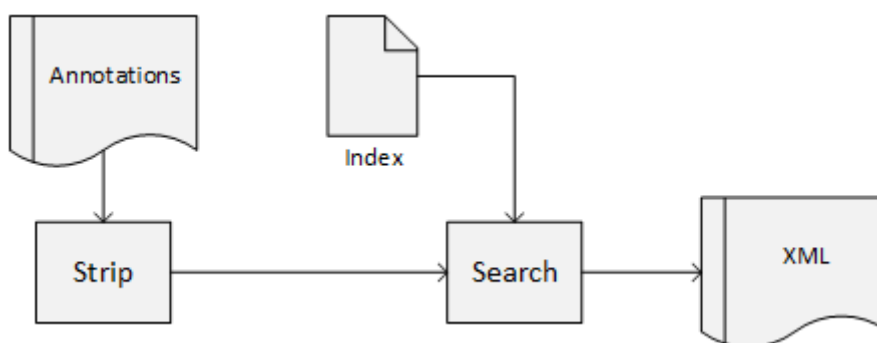


Figure 4.2: The structure of the system for presenting the articles. Input is the annotations file and the index containing all the articles. Output is a set of XML files, where each file contain all the articles central to each entity, ordered chronologically.

is shown in Figure 4.2. It take in the file containing the annotations and the index, and puts out a set of XML files, each containing the articulated central to one entity, chronologically ordered.

With the XML files at hand, reading the articles was straight forward, being able to look at one entity at the time, one article at the time. Since the articles were presented chronologically, novelty was easily determined, as each article would only be compared to previously read articles.

Table 4.3 show the number of annotated articles, and how many of these were labelled as *novel*. The norm is that there are a much bigger part of the central documents that are *no novel* that those that are *novel*. In choosing which entities to use in the test set, two goals were set to fulfil as throughout as possible, *1*) make the test set cover as many entity topics as possible, and *2*) get some entities with a small number of central articles, and some with many. The first is necessary because different entities appear in different types of articles. For example will Lovebug_Starski appear more in social media, and Boris_Berezovsky_(businessman) appear in a lot of news articles, and covering a larger set of entities make the test set cover as many situations as possible. The second is useful to make the system consider situations where there are little information available, and situations where what is known grow large. Some approaches to comparison of text will be impractical if the amount of text to compare each article get big. This might happen in cases where the entity appear often on the web, but the list of suggested updates to the Wikipedia article is not considered by any content managers, and may then grow to impractical sizes. In addition, some of the previously done suggestions may be wrong. Where this is the case, new articles will be compared to information that should not be in the knowledge set, and the error might cascade and grow.

| Entity name | Number of documents | Number of novel documents |
|--------------------------------|---------------------|---------------------------|
| Alex_Kopranos | 13 | 7 |
| Bill_Coen | 7 | 4 |
| Boris_Berezovsky_(businessman) | 80 | 17 |
| Boris_Berezovsky_(pianist) | 7 | 1 |
| Frederick_M._Lawrence | 11 | 3 |
| Jim_Steyer | 17 | 5 |
| Lovebug_Starski | 8 | 3 |
| In total | 143 | 40 |

Table 4.3: The annotated entity topics used in this project, with the total number of annotated documents and the number of novel documents in each set.

4.5 Quality of Data Set

By the choice of entities used in the data set may be discussed. As said earlier, the entities were chosen to avoid entities that were dormant during the frame of the stream, but also that do not appear too frequent. This has, however, not worked as well as perhaps was planned.

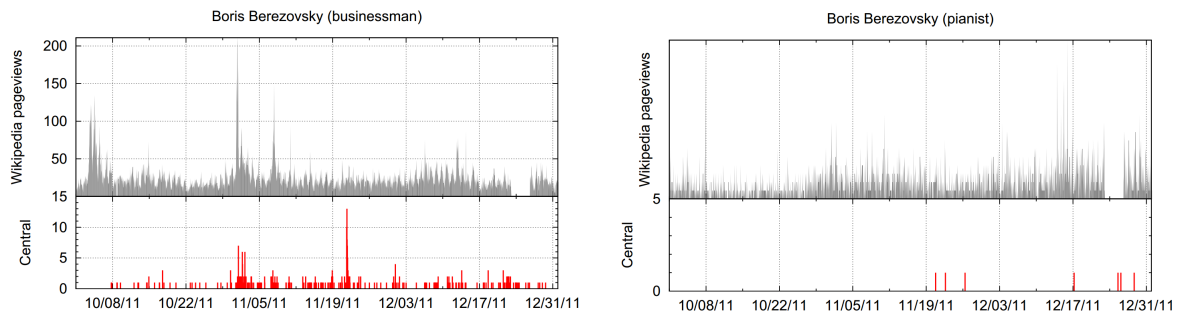
Even though there are an advantage of having entities that appear with different frequencies in the stream, giving opportunity to test for more cases, some of the entities are dormant more or less the entire period, see Figures 4.3a and 4.3⁶. These contribute less to the cases one wish to explore; how to keep entities up to date. If little happen relevant to the entity, there are few updates needed.

Examples of entities that do not appear often in the stream are *Boris_Berezovsky_(pianist)* and *Lovebug_Starski*. In the sub set used in this thesis, they only appear 7 and 8 times, respectively. These two entities has more in common. They both represent musicians, and both have several documents that contain almost no information. Some documents seem to be comments from social media on a song preformed by the musician, or a list of the content on a CD, which may include a song from the musician.

This cause the evaluation of these entities to be unstable, and small alterations may have big impact on the results. This may generally be a problem, as the *novelty* labelling was done manually during the project time, and there are therefore a limited number of labelled articles.

Another issue with the data set is that the entities are not random, besides the restrictions that has been set. It is unclear how exactly the entities were chosen, but they are not evenly distributed by which nationality they have, for example. Where this might seem less important, a data set specifically meant to be used to evaluate a specific system, should represent the data the system would be used on, and this property is better met if the entities are chosen at random, than if they have been chosen with a reason.

⁶Thanks to Krisztian Balog for providing these



(a) Entity having even occurrences of central documents in the stream over a period

(b) Entity having only occasional occurrences of central documents in the stream over a period.

Figure 4.3: The upper half show how many pageviews there was on the entity's Wikipedia page over the period, and the lower half show how many central documents that appeared in the stream in the same period.

Chapter 5

Approach

This chapter will show how the techniques described in Chapter 3 were used in the architecture of the implemented system. It will also explain how the experiments were set up and done. This include the pre-experiment steps done to the data set, and how the different methods were tested and combined.

In short, the prototype does the following things:

1. Apply wished representation technique to the documents to be compared
2. Compare the documents
3. Based on the comparison done, decide if the newest document in the stream present new information.

This is illustrated in Figure 5.1. As can be seen from the figure, the input to the system is the newest central document in the stream, the current KB article, and the knowledge set, containing previously suggested updates. Figure 5.2 give the timeline of the system. It show how the list of suggestions grow if a new central document contain novel information.

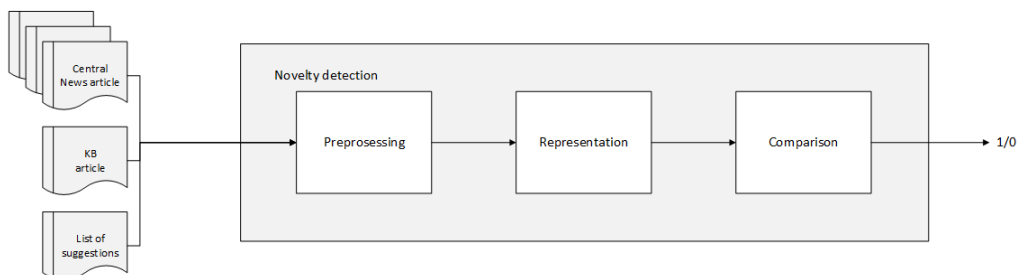


Figure 5.1: The Novelty component of a KBA system, the focus of this project. The *Novelty detection* component is the black box from Figure 5.2. First the input articles are preprocessed. Then they are represented as vectors, using *tf-idf* weighting and a simple boolean representation where the vector show which terms are present in each document. The *tf-idf*-vector is input for computing cosine similarity, and the boolean vector is input to the computation of the Jaccard coefficient. The output for each similarity computation is an *1* for novelty, or a *0* for non novelty.

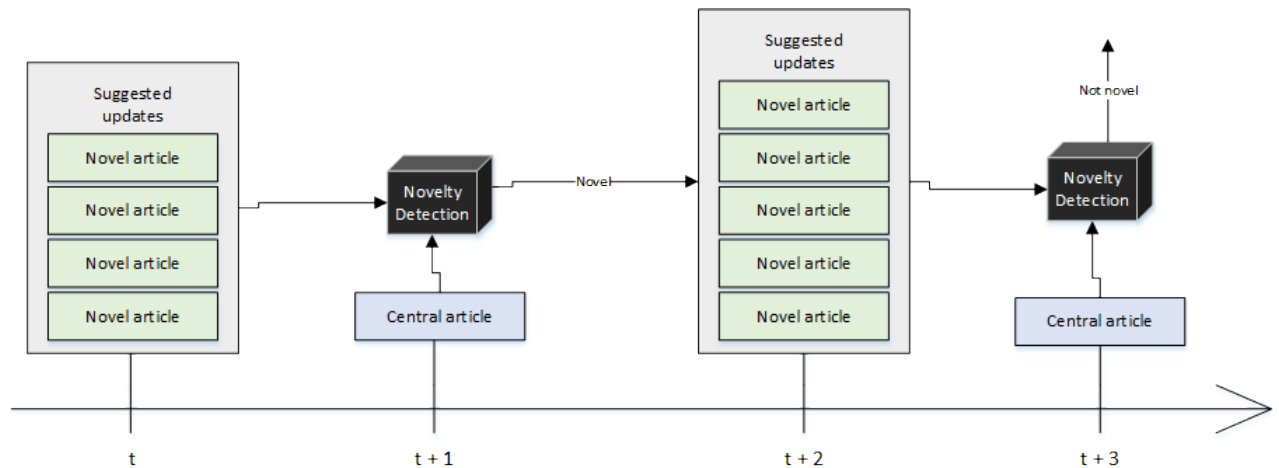


Figure 5.2: Looking at the time line for novelty detection. At time t , a knowledge set to a given KB topic exist. at a time $t + 1$, a new article appear in the stream. At this point, the system performing novelty detection, seen as a black box, will find out if the new article should be added to the list of suggestions. The list at time $t + 2$ represents the update. At time $t + 3$ a new central document appear in the stream. The system discards it as it does not present any new information about the topic in question.

| Before preprocessing | Characters removed | Lower case | Stop words removed |
|----------------------|--------------------|------------|--------------------|
| 398 | 371 | 339 | 318 |
| 51 | 48 | 48 | 39 |
| 848 | 784 | 700 | 668 |

Table 5.1: Three examples of number of terms in different documents before and after each step in the preprocessing.

5.1 Preprocessing

Preprocessing was the first step in the process the documents went through. The methods described in Section 3.1 were used. Where preprocessing make possible problems, for example by removing information, a decision has to be made if it is going to be done or not. Removing non-alphanumeric characters removes far more noise that keeping them might be worth. An alternative is to filter out the ones that give meaning to a word, such as in *president's*, however there are relatively few of these characters compared to the ones making noise, so all have been removed.

Another example is the step of setting all letters to lower case. When doing this step, it is becoming more usual to keep capital letters in names, by identifying these as named entities. This have not been done here, and again, the removal of noise is deemed more valuable than the information that is removed.

Stop words was removed using a static list of the most common words in the English language was used. This approach to removing stop words is simple, and cheap.

In addition to affect the output of the system, preprocessing also make the workload smaller, as it significantly reduce the amount of text units to be considered. An example of this is given in Table 5.1, counting 1-grams in random documents.

| | Overlap | Containment | Cosine |
|--------------|---------|-------------|--------|
| Weights | 0 | 0 | 1 |
| Boolean | 1 | 1 | 0 |
| Fingerprints | 1 | 1 | 1 |

Table 5.2

5.2 Document representation and comparison methods

Given the different methods for document representation and comparison given in Chapter 3, the combinations used in the experiments are given in Table 5.2. Some combinations give the same results, such as the use of boolean vectors and fingerprints when measuring overlap. The difference between these two are the time it take to compute similarity, and how the documents are stored.

5.2.1 Similarity Values

In general, two values have been used when comparing a document to the knowledge set. The most important difference of the two is what the new document from the stream is being compared to. Equation 5.1 compare the new document to the entire knowledge set, while Equation 5.2 compare it to each of the documents in the knowledge set.

$$Sim_S = \frac{Sim(A, S)}{|S|}, \quad (5.1)$$

$$Sim_{S_i} = Max(Sim(A, S_i)) \quad (5.2)$$

where A is the document currently being considered, S is the knowledge set at the time document A appear in the stream and S_i is the i th document in S .

Sim_S will tell how similar the new document is in general to the knowledge set. Sim_{S_i} tell how similar the newcomer is to the document in the knowledge set it is most alike.

Sim_S is multiplied by the size of S , to avoid the value to get smaller as S grows. Sim_S do not consider if the new document is very similar to a smaller document in the knowledge set. This would drown in the rest of information in the set. Therefore, Sim_{S_i} is necessary. It only look at the document in the set that the new document is *most* similar to, making it unable to find if the new document contain reused information from several different documents.

5.2.2 Combinations of Similarity Measures and Document Representations

The document representations techniques described in Section 3.3 and the similarity computation measures described in Section 3.4 has been used in different combinations. An overview of which combinations has been used is given in Table 5.2.

5.2.3 Partial Documents

In some cases, when a document may contain new information, the new information os only a small part of all the information in the document. When such a case appear, the document may be labelled as *not novel* if the new information "drown" in the rest.

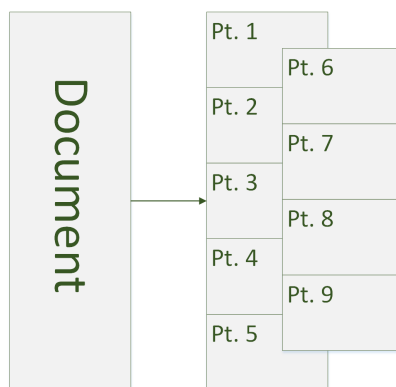


Figure 5.3: A document is firstly parted in five equal parts, then in four overlapping partitions.

To avoid this problem, incoming documents were partitioned into overlapping parts, as shown in Figure 5.3. The document was first divided in five parts. Then four new parts were created, the first starting at the middle of the first of the original ones, then the following three come consecutively.

The partitioning process is done by defining a partition size. The minimal size a partition may be is set to be 15 terms, to avoid small documents being divided into so small pieces, that they do not contain sufficient information to judge if they contain new information. If

$$\frac{\text{Number of terms in } D}{5} < 15, \quad (5.3)$$

the document is divided into as many pieces of 15 terms as possible. This means that a document containing only 15 terms is not split into smaller parts. The partition size is also an absolute lower limit for how small a partition may be. If, after making a set of 15-term-partitions, only 10 terms are left, these are added to the last of the partitions.

The partitions are used as input in the similarity methods, instead of using the entire document as input. This means that novelty is decided for each partition. When a partition is judged to contain *novel* information, the document it is of is also labelled as *novel*. However, only the partitions that actually contain new information are added to the knowledge set. This helps prevent the amount of noise information in the knowledge set from getting too big.

In the TREC Novelty track, each sentence was considered alone, and only the novel sentences were returned. Since the returned text in a KBA system will be read and considered by a human before it is used, it is desirable to return whole documents, so the person doing an update knows the context of the information they have at hand. However, a large document of 100 sentences, might only have one that contains new information. When comparing whole documents, and making a decision about novelty based on the entire content, the one sentence containing new information might not be enough to label the document as *novel*. Following this, it might be interesting to label smaller pieces of text, while still returning the entire document. This way, the large document only having 1% new information might still be returned as *novel*.

A question to be considered is what should be added to the knowledge set, that the system sees. While the list available to humans should contain full documents, the 99% of the document described above will make "noise" in the list used by the system. Only the novel part of a document is useful for the system, and here, only the piece of text judged to contain novel information was added.

5.2.4 N-grams

Working with n-grams are potentially very expensive. As said in Section 3.2, a too large n will make it difficult to catch slightly rewritten passes of text, while a too small n capture less information.

To catch as much information as possible, several different n will be used. In this thesis, n always started at 1, causing the system to first consider only single terms. Then, n was incremented, making the system give the document a similarity score for each n up to an upper threshold.

Setting an upper limit for n may be hard, as different n capture different information. A general rule should be that n should not be incremented when a larger n no longer provide new information.

If n is incremented until a higher n no longer provide new information, the comparison of different document will stop at different n . This make it less trivial to compare different similarity values. As n-grams with high n contain more information than those with low n , documents that reach a high, useful n should be bore similar than those reaching a lower n . From this there are a few questions to be answered.

- When is a n useful?
- How should two similarity values for documents reaching different n be compared?

The first question is simply about defining what "useful" mean in this case. An useful n has been interpreted to be a n that provide new information about the content of a document in the setting of comparing it to another document. A n provide new information as long as a document has a n-gram in common with the one it is compared to. Therefore, n is incremented until the numerator in Equation 3.3 is 0. An upper limit of $n = 15$ is set. This limit was set to avoid doing too much calculation when two large pieces of text are identical.

In the implemented prototype, n has been combined with the other similarity measures by

$$Sim = sim \cdot 0.5 + \frac{n \cdot 0.5}{15}, \quad (5.4)$$

where sim may be either Sim_S or Sim_{Si} , and Sim is an adapted value for Sim_S or Sim_{Si} for n . This mean that when n-grams are used, $\frac{n}{15}$, being a normalized n , is weighted 50% of the entire similarity value.

5.3 Output

The similarity values found using Equations 5.1 and 5.2 contain information about how similar a piece of text is to a set of texts. From this, a decision has to be made. Do the piece of text in question contain new information? How this decision was made is described in this section.

5.3.1 Weighting of similarity values

The similarity values Sim_S and Sim_{Si} are both in the range [0, 1]. To make a decision, these has been combined to one value. One thing that is not known is which of these values are most important in finding novel documents. Therefore, they were weighted, so that

$$Sim_w = w_x \cdot Sim_S + w_y \cdot Sim_{Si}, \quad (5.5)$$

where x and y are the weights of Sim_S and Sim_{Si} . To make Sim have the same range as the other values, x , y and z can be given the following properties:

$$w_x + w_y = 1 \quad (5.6)$$

$$w_x, w_y \in [0, 1]. \quad (5.7)$$

All combinations of values for w_x and w_y , using intervals of 0.1 were tested. In addition, a threshold T is needed so that if $Sim_w < T$ for a new document in the stream, the document is labelled as *novel*. As Sim_S and Sim_{Si} are computed in different ways, different weightings of the two values need different thresholds to give good results. Sim_{Si} will always have a higher value than Sim_S , so weights where Sim_{Si} is given most weight will need a higher threshold to perform well. To find ideal threshold for each weight combination, different thresholds were tried for each of them.

5.3.2 Thresholds for similarity values

Another approach to deciding if a text contains novel information or not, based on the similarity values Sim_S and Sim_{Si} , is to set a threshold for each, so that the information in new text is labelled by the following rules:

$$\text{Information is } \begin{cases} \text{novel} & \text{if } Sim_S < t_x \\ \text{novel} & \text{or } Sim_{Si} < t_y \\ \text{not novel} & \text{otherwise,} \end{cases} \quad (5.8)$$

where t_x and t_y are the thresholds used. Here too, different values for t_x and t_y need to be considered.

Chapter 6

Experimental Setup and Results

6.1 Evaluation Metrics

For each topic in the data set, the number of rightly and wrongly labelled documents were counted. The values retrieved were

- True positives, TP , the number of documents correctly labelled *novel*
- True negatives, TN , the number of documents correctly labelled *not novel*
- False positives, FP , documents wrongly labelled as *novel*
- False negatives, FN , documents wrongly labelled as *not novel*

From these, three evaluation measures were found, *precision* P , *recall* R and *F-measure* F_β .

$$P = \frac{TP}{TP + FP} \quad (6.1)$$

$$R = \frac{TP}{TP + FN} \quad (6.2)$$

$$F_\beta = \frac{(1 + \beta^2)TP}{(1 + \beta^2)TP + \beta^2FN + FP} \quad (6.3)$$

Precision measure how much of the returned *novel* documents that are correctly labelled. Recall measure how big part of the total number of *novel* documents that were actually labelled as *novel*. These two are classical evaluation measures in information retrieval and related fields, but presents some issues in how to interpret the results. Given a topic that mostly contain *not novel* documents, and the system returns many *novel* documents, recall will most likely be high. A topic containing mostly *novel* documents, having the same system detect only one, will give a high precision. Combining these two topics, this system will have an average score of about 0.5 on both precision and recall. This is a good score considering that the system is actually perform horribly.

Because of these issues concerning precision and recall, F-measure is a good alternative. F-measure is the harmonic mean of recall and precision. It allows a weighting of precision and recall, where β is how many times more important recall is that precision. β equal to one give them the same importance, and is used in this thesis.

6.2 Perspectives of Results

The task the prototype is to solve may be looked at from two perspectives. The one that is mainly used in this thesis is to *find the articles that contain new information, and filter out these*. The alternative perspective is to *find the articles that contain repetitive information, and filter out these*. The difference is how one choose to look at the construction of the output of the system. Is the output the articles that are filtered out, or are the output the same stream as the input, where repetitive information has been filtered out?

The perspective do not affect the task, but it is important in presenting and interpreting the results. The next section will introduce the three baselines that were used to compare the system's results to. Figure 6.1 demonstrate the two perspectives for the baselines.

6.3 Baselines

Results say little when they are only presented as numbers, without anything to show what the numbers mean. Therefore, some baselines were used as indication on how well the system labelled the documents. The task the system is set to solve may be looked at from two perspectives. Firstly, one may look at the task to be to find *novel* documents, and return these. Secondly, the task may be seen as the process of finding the documents that are *not novel*, and filtering out these. These two perspectives is useful because there are significantly more *not novel* documents than *novel* in the stream. Depending on how one choose to look at the task, the system preform differently. The three baselines used are chosen to capture both views.

- **All novel** This baseline is found by labelling all documents as *novel*.
- **All not novel** This is the opposite of the first baseline, by labelling all documents *not novel*.
- **Random** This baseline was made by giving all documents a random label, each document having a 50 % chance of being labelled as either.

The two first baselines are only useful in one of the perspectives described above. The *all not novel* baseline only give usable results when looking at the task of finding *not novel*. Compared to the results from the *all novel* baseline when attempting to find *novel* documents, the *all not novel* baseline preform far better, due to the overweight of *not novel* documents in the stream. This make the usefulness of the two perspectives clear, as they help to give a more complete evaluation of the performance of the prototype.

Figure 6.1 show how well the three baseline methods do. The *all novel* and *all not novel* score 0 at finding repetitive information and finding novel information, respectively. This is because they do not find any of those we are looking for based on each perspective.

Looking at the *random* baseline, it is clear that it is easier to find repetitive information, because there are significantly more *not novel* documents than *novel* in the data set. This is also why the two perspectives are important. It is easy to make the results look better by choosing to use the perspective that make the results look best.

To avoid cheating, both perspectives have been included in the evaluation metrics. All evaluation data is given using the following metric

$$F_{mean} = \frac{F_N + F_R}{2}, \quad (6.4)$$

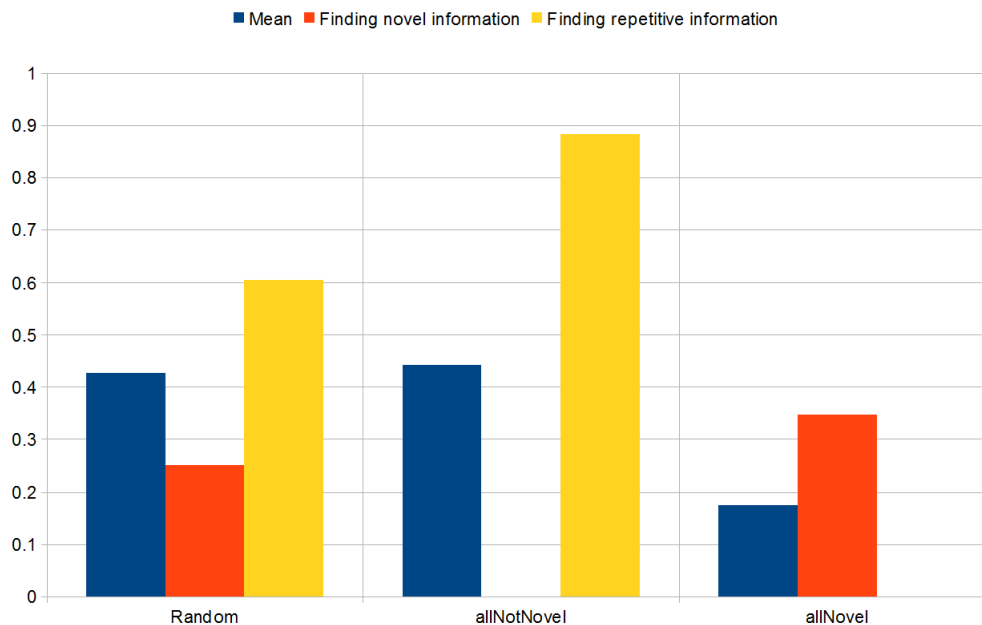


Figure 6.1: Results from baseline methods, giving both perspectives, and a mean of the performance based on perspective. Performance is measured using f-measure.

where F_N is the F-measure when looking for novel information, and F_R is the F-measure when looking for repetitive information.

6.4 Thresholds and Weights

Using the weights described in Section 5.3.1 along with the similarity measures described in Section 3.4, a similarity value is obtained, containing information about how similar the new document in the stream is to the knowledge set. However, this value do not have any meaning when it is seen alone. To make decisions about what the value mean, a threshold is needed, where a value above the threshold mean that the document in question should be discarded as *not novel*, as its content is too similar to previously seen information, and a document below the threshold is considered to contain enough new information to be labelled as *novel*.

In some cases when working with document similarities, this threshold may be set by looking at a set of values of a test set, trying to find a boundary that separate the similarity measured of one class of documents from another. As the task was in the TREC KBA track, to find relevant documents, this is easy, as a document mostly is relevant to a topic regardless of time, and of which information has appeared previously in the stream. However, when working with novelty detection, this is not the case. For each document that is judged to be *novel*, the set to compare new documents to changes. That is, each decision is dependent on bot a temporal factor, and what information has appeared in the stream.

This dependency can be seen in Equations 5.1 and 5.2. The set S is dependent on which of the previous documents from the stream that has been labelled as *novel*. Therefore, to find a threshold, experiments were done using different thresholds, ranging from where most document are returned as *not novel* to where most are returned as *novel*. Using these limits, the goal is to consider then entire range of thresholds where each document comparison method might perform best. The safest way to ensure to use the entire range in interest is to use the range $<0, 1>$, since each comparison method

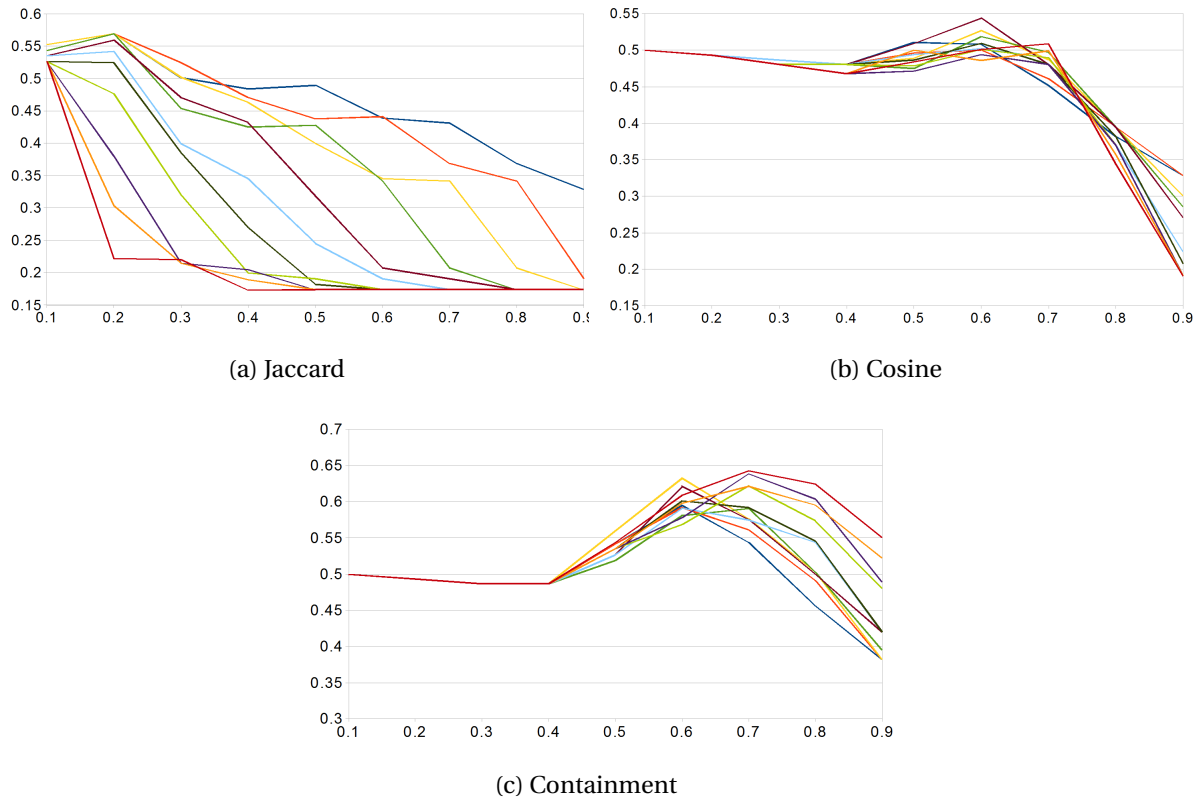


Figure 6.2: Results from looking at thresholds and weight sets using the different similarity measures. The distribution of the graphs for each similarity measure differ significantly. Jaccard and Containment favour the similarity values from Equations 5.1 and 5.2 opposite of each other, while Cosine favour a more even distribution.

return a value between 0 and 1. Due to computational load, a thresholds within the range, using a step size of 0.1 was used, giving 9 thresholds used in each experiment for each weight set.

A F_{mean} value of 0.5 when using a low threshold may appear when all but one document is returned as *novel*. In this case, the one returned as *novel* is the first in the stream, which is always returned as *novel*. Therefore, to have started with a low enough threshold, evaluation values should be close to 0.5 at the lowest thresholds used.

6.5 Simple Text Similarity

The first set of experiments done was done using the text similarity measures described in Section 3.4, only using single terms as text units, and considering each document as a whole, not partitioning them. Figure 6.2 show an overview of the results from these experiments. Full figures in larger scale is given in Appendix A.

6.5.1 Jaccard

Figure 6.2a show the results when using Jaccard as similarity measure. Where the lines meet at the bottom, recall is 1. What is interesting with this figure is the distribution of values. The general rule is that cases with a higher weight on Sim_{S_i} do better.

A peak is found at threshold = 0.2, so another experiment was done using thresholds between 0.1 and 0.3, to find if using a smaller step size for the threshold would give better results. The result from this experiment is presented in Figure A.2. Here there are no clear peak. However, there are a maximum for most weights at threshold 0.16. The results for all weights fall after this point, or, hold the same value for a few of the higher thresholds before dropping. From this, choosing a threshold in [0.16, 0.22] is justifiable.

When looking at the results in Figure A.2, it is clear that the output is not very stable, that is, a small change in threshold or weight might have a large impact on the results. Ideally, this should not be the case. Preferably, the graphs should be more even, and should not cross each other too much. This may come from the quality of the data set, as discussed in Section 4.5, and the small size of the annotated test set.

6.5.2 Cosine Similarity

From the graph in Figure A.3, it can be seen that the relationships between thresholds and weights for Cosine similarity give a different distribution in results. There, results for different weight sets are less important, as the graphs are more collected. The best result from this was the same as may be seen in Figure A.3; a threshold of 0.6 for weights [0.4, 0.6] is the best alternative for Cosine.

6.5.3 Containment

Figure A.4 give the results from the threshold experiment done with a full range of threshold values, and using the Containment similarity measure. The shape of the graph have much in common with the corresponding results graph for Cosine similarity. However, there are one major difference. While how well the different weight sets does it compared to each other using Cosine differs largely from threshold to threshold, the weight sets are more consistent compared to each other when using Containment. Now, comparing the results for Containment to those for Jaccard, there are one more point that is worth making. Using Jaccard, the weight sets weighting Sim_{Si} higher than Sim_S does it better. With Containment, this has been turned around, having the weight sets favoring Sim_S come out on top.

6.6 Using Partial Documents

This approach did not work as well as assumed. Properties of each similarity measure are discussed in the sections below. In general, the only similarity measure doing better than bad is Containment, but the results from this experiment are very unstable.

6.6.1 Jaccard

When using Jaccard to compare partial documents, the similarity values for Sim_S drop fast, as the number of texts in the knowledge set grows faster when partitions are added in stead of whole documents. This cause weight sets that favour Sim_S to preform only when the threshold is very low. The weight sets favouring Sim_{Si} and those having a more even weighting of the two do better overall, but sinks further from threshold 0.1.

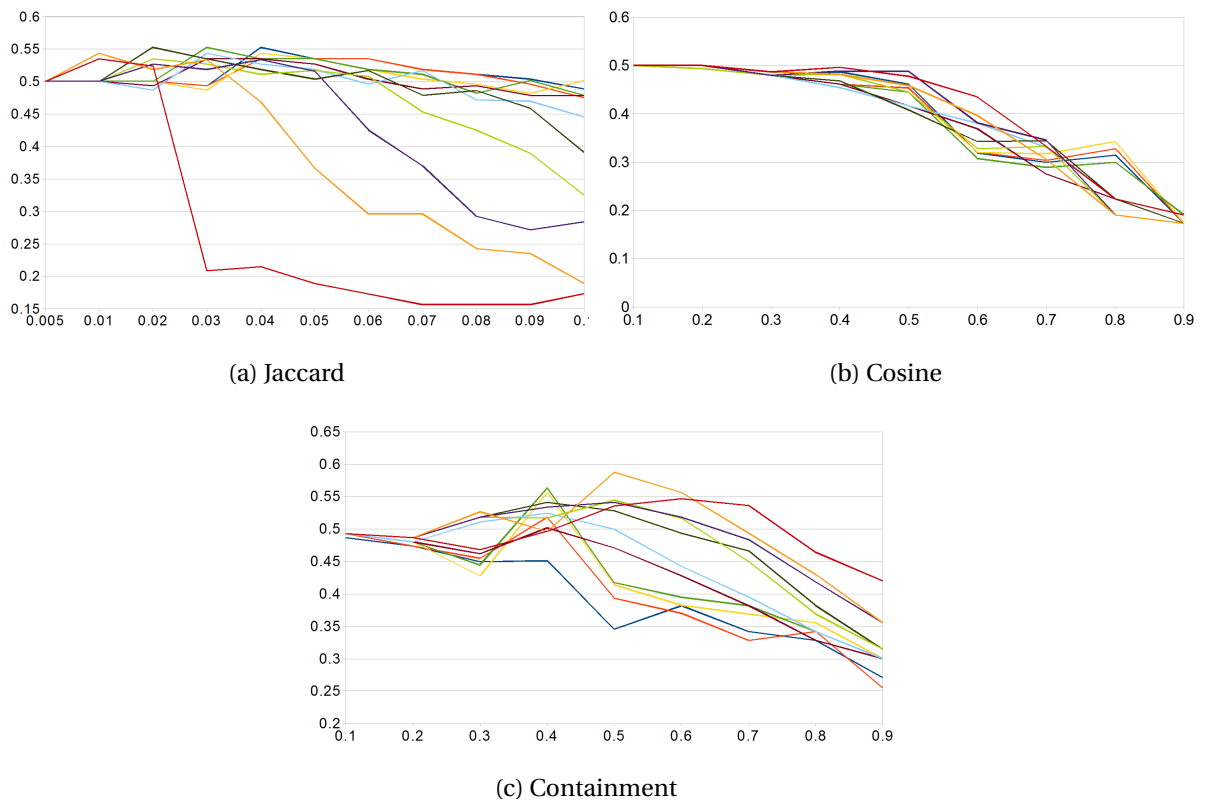


Figure 6.3: Results from experiments using partitioned documents. Jaccard mostly return very low values from Equation 5.1, and the thresholds in the experiment was adapted thereafter. Using Cosine in combination with partial documents seem to provide any usable information. Containment get a very noisy result, perhaps as a result of the size and quality of the data set.

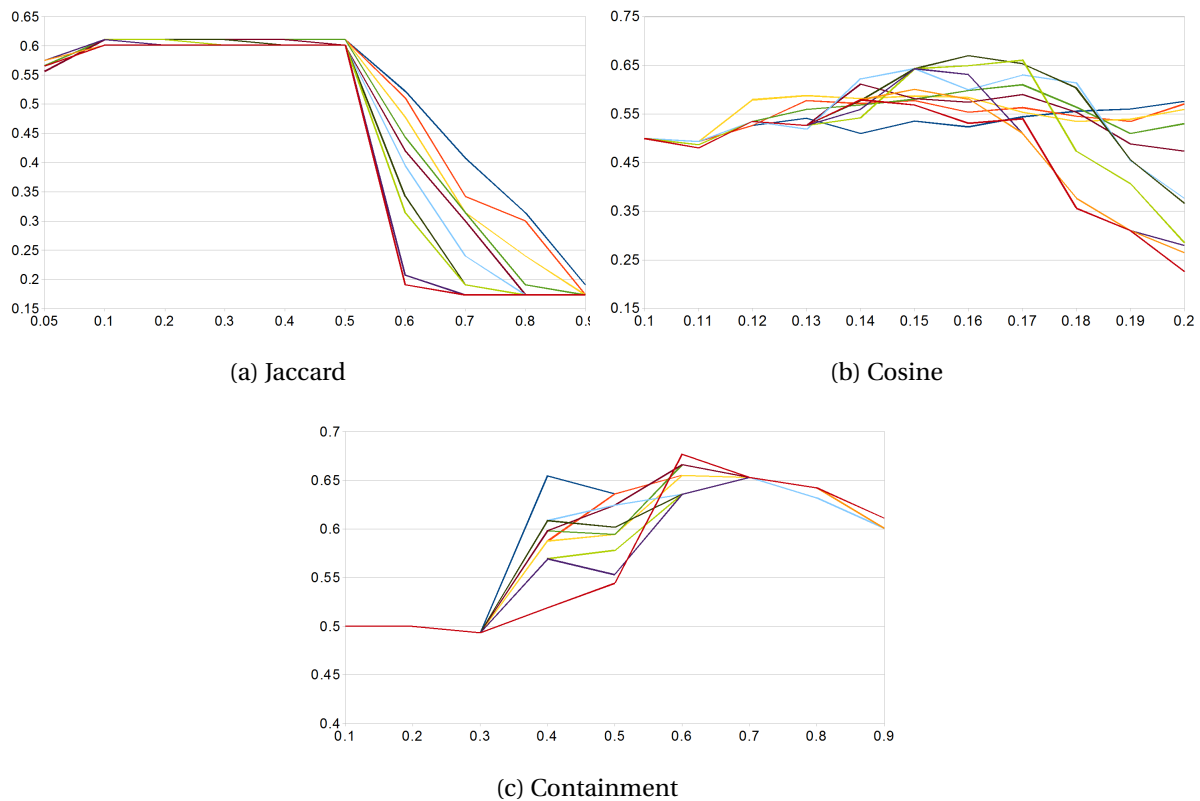


Figure 6.4: Results from using n-grams. The results from using Jaccard is dominated by the high weight of n . Containment suffer less from this, and give reasonable results. Cosine judge most documents to be very dissimilar to the knowledge set, and the thresholds in the experiments are adjusted to fit this.

6.6.2 Cosine

The results from this experiment start where all documents but one are returned as *not novel*, giving a F_{mean} of 0.5.

6.6.3 Containment

Containment in combination with partial documents give very unstable results, see Section 4.5. However, there are still a pattern for Sim_S being the most important similarity value.

6.7 Using n-grams

6.7.1 Jaccard

The results graph for using n-grams with Jaccard stand out in Figure 6.4. The top of the graphs are very flat, and when they drop, they mostly drop to the same value, before all drop far at threshold 0.5. This flat structure are common for all graphs in Figure 6.4, but stand most out in Figure 6.4a.

This is explained by how the weighting of n was done. This was described in Section 5.2.4. The result of this is that those documents having a large n to get a very high similarity value to the knowledge

set. From looking at the "crack" at $threshold = 0.5$, it is clear that many of these have a similarity value between 0.5 and 0.6 .

While this causes most weight sets to do reasonably well, it is safe to assume that some information is lost, and it should be possible to refine the results by differencing more between the documents having a large n . This could be done by trying out different weights for n , to see if weighting it less prevent a large n from "drowning" the other similarity values.

6.7.2 Cosine

When using Cosine similarity in combination with n-grams, most documents are judged less similar than when using the other similarity measures, and the threshold range have been adjusted there after. Having a threshold at 0.2 , most documents are judged *not novel*. From this, the threshold have been given a range of $[0.1, 0.2]$, using a step size of 0.02 . Computing Cosine similarity using n-grams is very expensive, as there are several computations that need to be done for the entire knowledge set each time a new document appear, for example document frequency. This cause the experiment to be time consuming to perform as the number of documents judged *novel* get higher.

Generally, Sim_S have a significantly higher value than Sim_{Si} , often having values closer to 0.5 . This mean that weight set $(0.0, 1.0)$, being the one weighting Sim_S 100% , would return considerable results up to threshold 0.5 . The high distribution of results for the different weight sets at threshold 0.2 suggests that higher thresholds should be examined. This is left for future work.

6.7.3 Containment

Using n-grams in combination with Containment make the differences between the weight sets less significant. Throughout the entire threshold range, the weight sets weighting Sim_{Si} and Sim_S switch on being the best performing. This may be due to the nature of the data set, as described in Section 4.5.

6.8 Observations Summary

When looking at the similarity measures alone, without using the specializations, an interesting observation is how the importance of the two similarity values vary from method to method. While Jaccard have better use of Sim_{Si} , the results graphs for the weight sets being evenly distributed, barely crossing each other. Containment, on the other hand, have better use of Sim_S . The results graphs cross more, but there are still a clear pattern. Cosine discriminate less on the two values, having the graphs cross each other several times. However, at the peak, the weight sets performing best are those favouring Sim_{Si} $60 - 80\%$.

Partitioned document need further investigation. In this thesis, each partition judged to contain new information was added to the knowledge set, in stead of the entire document, as was done in the other cases. It would be interesting to see how the method worked if the entire document was added to the knowledge set. This would perhaps most significantly affect the Sim_S value.

N-grams too, have room for investigating further the findings from this thesis. First of all, different weightings of n in Equation 5.4. This would at least remove the flat part of the results graph for Jaccard. Containment have an interesting distribution of results, as the graphs spread out like they do in Figure 6.2c, but then gather as the threshold get higher. This, again come from the high weight of n .

In general, the results are more or less unstable, having small changes in weights or threshold significantly affect the results. As said before, this may have some root in the quality of the test set, see Section 4.5.

Chapter 7

Conclusion

Facing the massive amount of information available through the Internet, knowledge bases provide their users with structured, organized and relational information. This help information seekers satisfy their information needs in one place, either browsing relations between entities, or by searching. They rely on the knowledge bases' content to be up to date, despite the fact that only a small part of the users contribute in keeping it so. Knowledge base acceleration aim to make this part of contributors bigger, and making them more efficient, and thus ensuring a bigger part of the knowledge base to be up to date.

In this thesis, knowledge base acceleration systems was examined, identifying important requirements for what such a system should return to the user of the knowledge base. Among the most important features a KBA system need to provide the output it need to preform as wished, is novelty detection.

With base in the TREC KBA track, this thesis propose a prototype for a system for preforming novelty detection as a part of a KBA. This prototype explore different possible approaches to novelty detection, an test them in the KBA setting.

7.1 Further work

Using the methods presented in this thesis, there are still combinations that have not been tested, and experiments connected to them that may be done.¹

Piece of new information: Is this actually central to the entity? Or is it just gibberish in a document containing *some* central information to the entity?²

A KBA system that is to be used in a knowledge base should always consider the actual knowledge base article about the topics. Without doing this, a KBA system is not useful any further than as a proof of concept. Adding the KB article may be done in different ways. Either, one may add the entire article as a piece of knowledge in the knowledge set, or one may split it into smaller parts, for example by sections. The latter is probably preferable, as a news article is more likely to contain information about a field within the entity, rather than to contain information generally about it. Because of this, splitting it may make it easier to match repetitive information.

There are a danger of that the knowledge set may grow to an inconvenient size. On might assume that entities that appear often in the news is more often visited in the knowledge base than those that appear less frequent, and thus, those that are seen more in the news is probably updated more

¹This must be written when all experiments are defined...

²Put thoughts about the data set in a separate section. Ask Kjetil where it belongs.

often. However, this assumption may not hold enough to keep the knowledge set to an usable size. Therefore, a KBA system should include a mechanism for keeping the knowledge set from growing too big. Examples on how to do this is updating the set every time the entity is updated, removing information that might have been added, or having stricter requirements for adding an article for very active entities.

The goal of a KBA system is to make more people contribute to keeping KB entities up to date, and making the actual updating process more efficient. To reach this overall goal, the presentation of the system output is important. Among factors to consider at this point is how the user should be notified that there are available updates, how each document in the suggestion list should be presented, and how the system is to register what updates have been done. This is far beyond the scope of this thesis, but is an important thing to consider in developing a KBA system.

Users who update the suggestion list provide a possibility for valuable user feedback information. They may remove information that they themselves have added to the entity, or information that someone else have added. It is also desirable to give them the option to give a reason for the update they have done on the list. Collecting this information may help identifying articles that have been wrongly labelled as *novel*. This information may be used to improve the novelty detection algorithm.

Bibliography

- [1] Krishna Bharat and Andrei Broder. Estimating the relative size and overlap of public web search engines. In *7th International World Wide Web Conference (WWW7)*. Citeseer, 1998.
- [2] Daniel Lehmann and Menachem Magidor. What does a conditional knowledge base entail? *Artificial Intelligence*, 55(1):1–60, 1992.
- [3] Razvan C. Bunescu, Marius Pasca, and Marius Pasca. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the 11st Conference of the European Chapter of the Association for Computational Linguistics, EACL*, 2006.
- [4] Jun'ichi Kazama, Kentaro Torisawa, and Kentaro Torisawa. Exploiting wikipedia as external knowledge for named entity recognition. In *Proceedings of the 10th annual Conference on Empirical Methods in Natural Language Processing Conference on Computational Natural Language Learning, EMNLP-CoNLL*, pages 698–707, 2007.
- [5] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. In *International Journal on Semantic Web and Information Systems*, pages 1–22, 2009.
- [6] Ellen M. Voorhees and Donna K. Harman. *TREC*. The MIT Press, 2005.
- [7] Ian Soboroff, Donna Harman, and Donna Harman. Novelty detection: The trec experience. In *Proceedings of the 43th annual Human Language Technology Conference Conference on Empirical Methods in Natural Language Processing, HLT/EMNLP*, 2005.
- [8] Le Zhao, Min Zhang, and Shaoping Ma. The nature of novelty detection. *Information Retrieval*, pages 521–541, 2006.
- [9] Uri Hanani, Bracha Shapira, Peretz Shoval, and Peretz Shoval. Information filtering: Overview of issues, research and systems. In *User Modeling and User-Adapted Interaction*, pages 203–259, 2001.
- [10] Xiaoyan Li, W. Bruce Croft, and W. Bruce Croft. An information-pattern-based approach to novelty detection. In *Information Processing and Management*, pages 1159–1188, 2008.
- [11] Xiaoyan Li and W. Bruce Croft. Improving novelty detection for general topics using sentence level information patterns. In *Proceedings of the 15th ACM international conference on Information and knowledge management, CIKM '06*, pages 238–247, 2006.
- [12] Y. Zhang, J. Callan, and T. Minka. Novelty and redundancy detection in adaptive filtering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 81–88. ACM, 2002.
- [13] Y. Yang, J. Zhang, J. Carbonell, and C. Jin. Topic-conditioned novelty detection. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 688–693. ACM, 2002.
- [14] Michael Gamon. Graph-based text representation for novelty detection. In *Proceedings of the*

- First Workshop on Graph Based Methods for Natural Language Processing*, TextGraphs-1, pages 17–24, 2006.
- [15] Michael Benersky and W. Bruce Croft. Finding text reuse on the web. In *Proceedings of the 2nd ACM International Conference on Web Search and Data Mining, WSDM*, 2009.
- [16] Jangwon Seo, W. Bruce Croft, and W. Bruce Croft. Local text reuse detection. In *Proceedings of the 31st ACM Special Interest Group on Information Retrieval, SIGIR, Conference*, pages 571–578, 2008.
- [17] Sergey Brin, James Davis, and Hector Garcia-Molina. Copy detection mechanisms for digital documents. In *Proceedings of the 20th ACM Special Interest Group on Management of Data Conference*, pages 398–409. ACM, 1995.
- [18] Alberto Barrón-Cedeño. On the mono- and cross-language detection of text re-use and plagiarism. *Procesamiento del Lenguaje Natural*, 50:103–105, 2013.
- [19] J. Allan, J.G. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998.
- [20] Juha Makkonen, Helena Ahonen-Myka, Marko Salmenkivi, and Marko Salmenkivi. Simple semantics in topic detection and tracking. pages 347–368, 2004.
- [21] Paul Clough, Robert J. Gaizauskas, Scott Songlin Piao, Yorick Wilks, and Yorick Wilks. Measuring text reuse. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 152–159, 2002.
- [22] William B Cavnar and John M Trenkle. N-gram-based text categorization. *Ann Arbor MI*, 48113(2):161–175, 1994.
- [23] Andrei Broder. Identifying and filtering near-duplicate documents. In *Combinatorial Pattern Matching*, pages 1–10. Springer, 2000.
- [24] Michael O. Rabin. *Fingerprinting by random polynomials*. Center for Research in Computing Techn., Aiken Computation Laboratory, Univ., 1981.
- [25] Andrei Z. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE, 1997.
- [26] Martin F Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

Appendix A

Graphs

This appendix contain graphs showing the entire results from the experiments, form which only a summery have been given in the presentation.

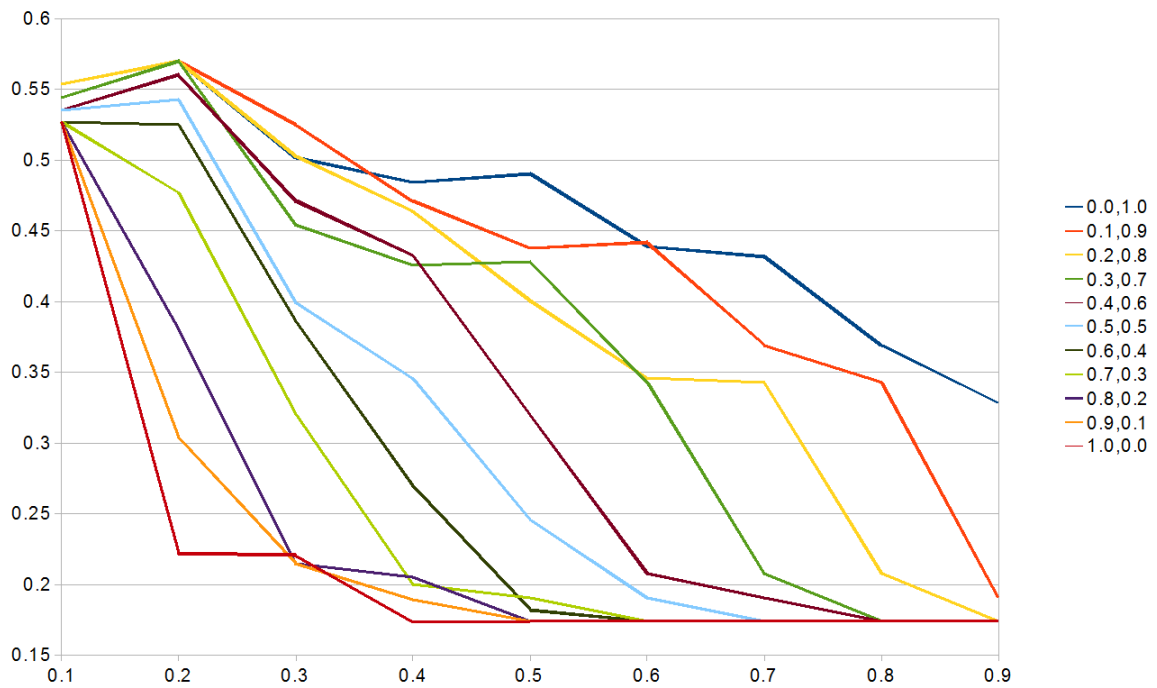


Figure A.1: Results from trying different thresholds in combination with different weight sets. Worth noticing is the pattern of the graphs, that represent how well each weight set does it for each threshold. They rarely cross each other, and are largely distributed. .

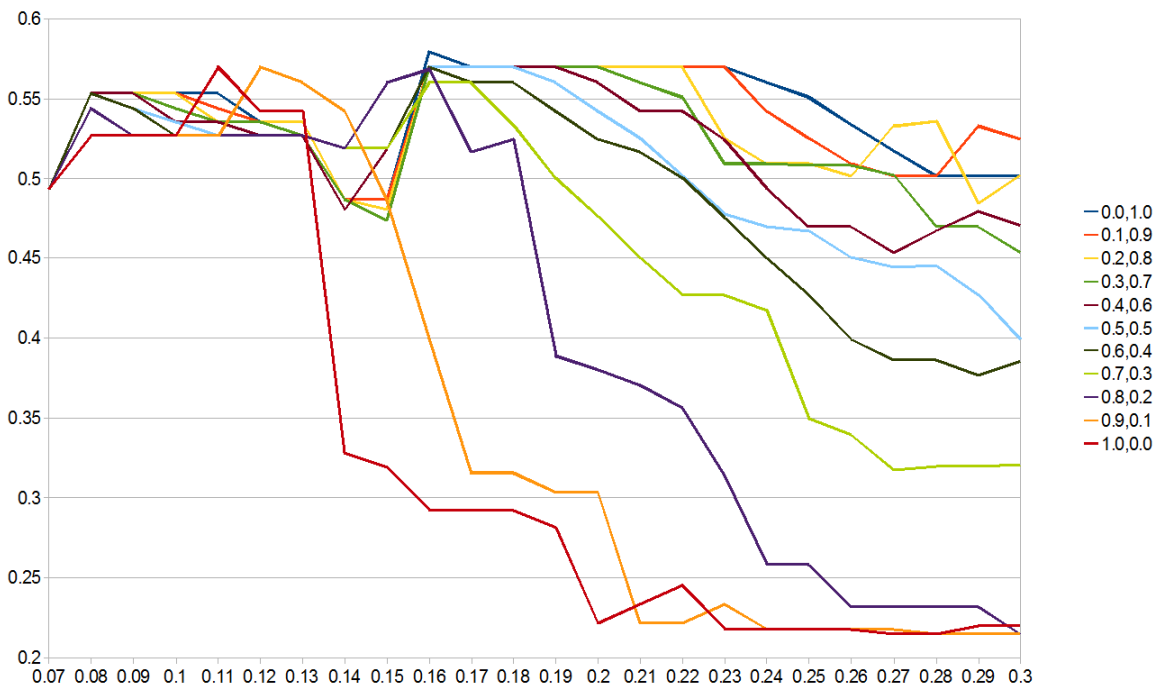


Figure A.2: Using a smaller range of threshold to find the best combination of weight sets and thresholds using Jaccard. The results here show how unstable the results really are, sensitive to small changes in threshold or weights.

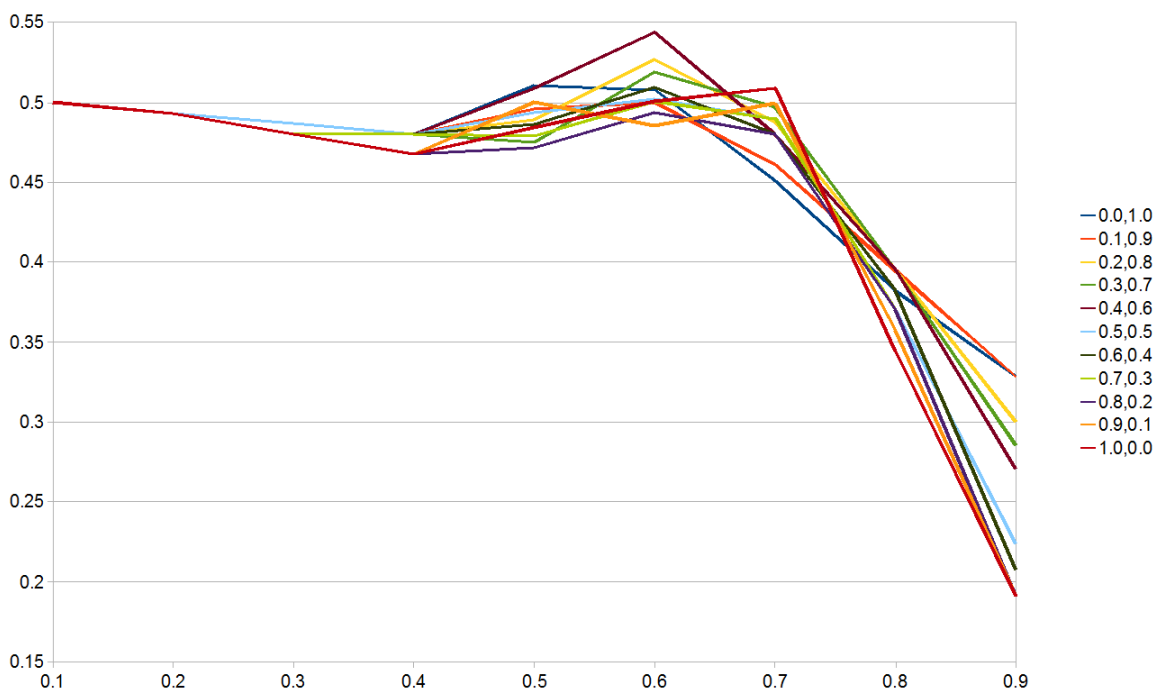


Figure A.3: Combinations of thresholds and weights using Cosine similarity. weights have less to say, but those favoring Equation 5.2 60 to 80 % do slightly better.

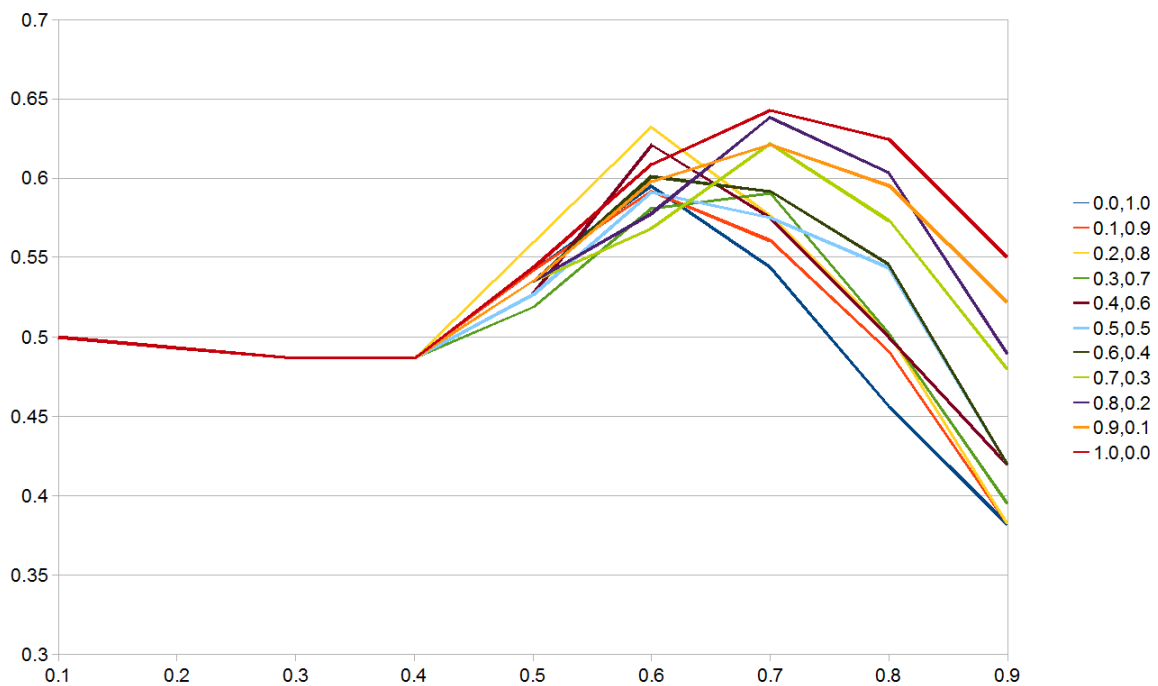


Figure A.4: When using Containment as similarity measure, weight set favouring Equation 5.1 generally do better.

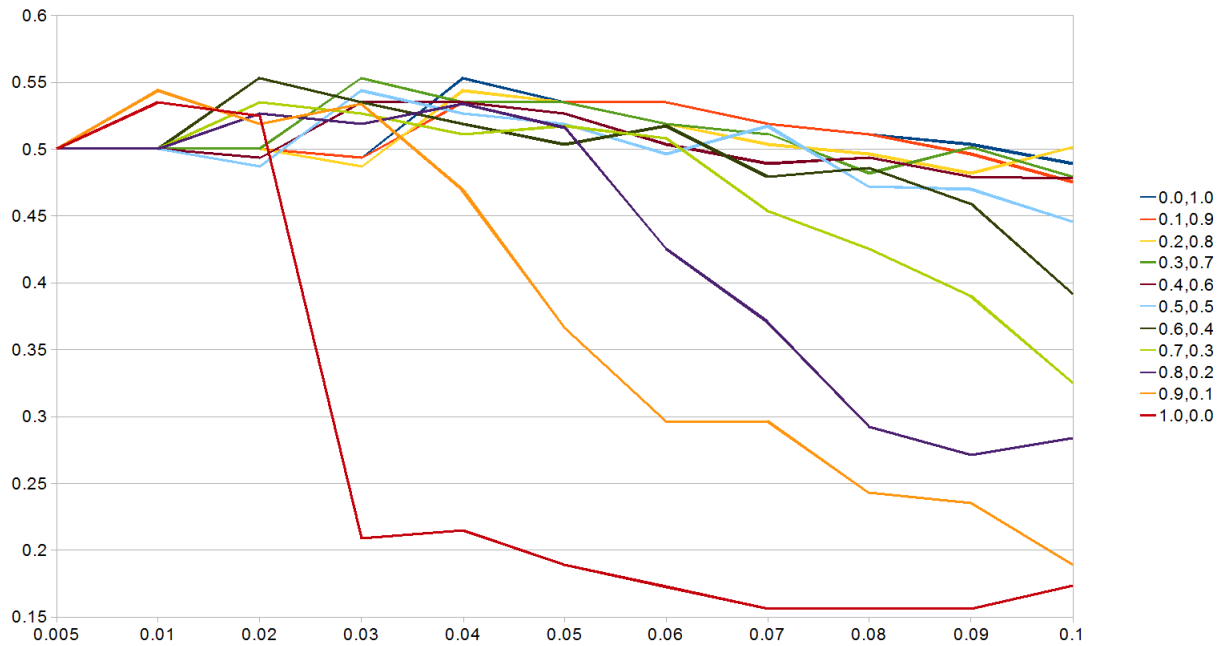


Figure A.5: Combining Jaccard and partitioned documents. Equation 5.1 generally return small similarity values, as the size of the knowledge set is larger when adding up to nine partitions per document in stead of only one document.

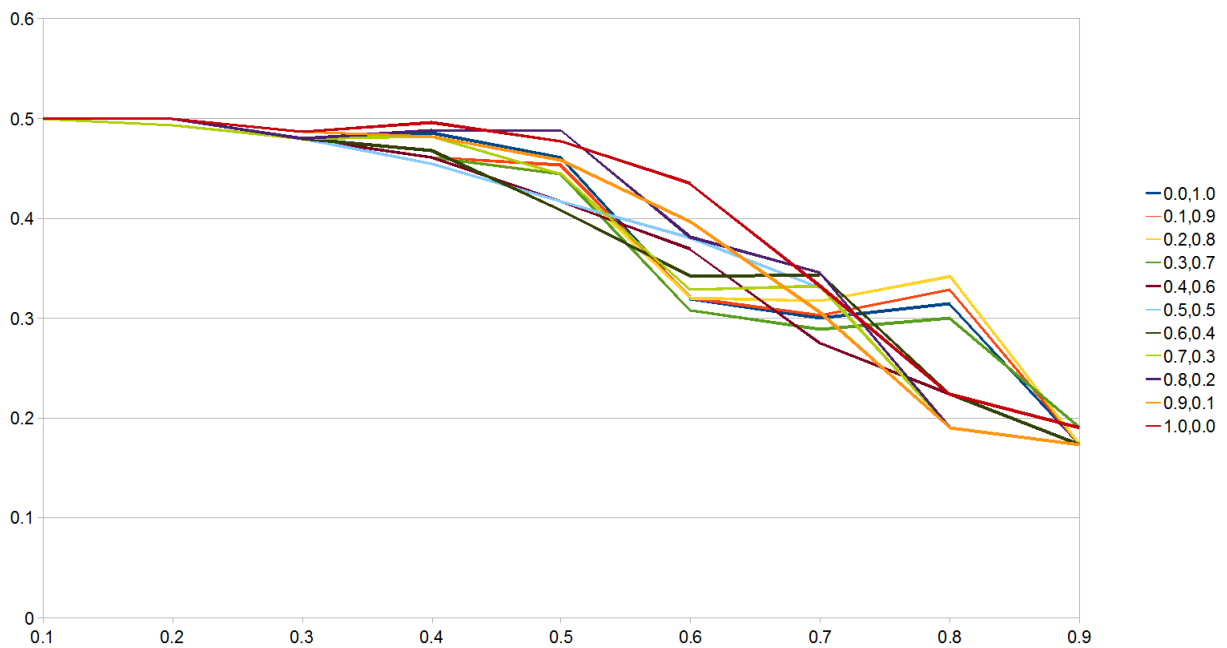


Figure A.6: Cosine using partial documents.

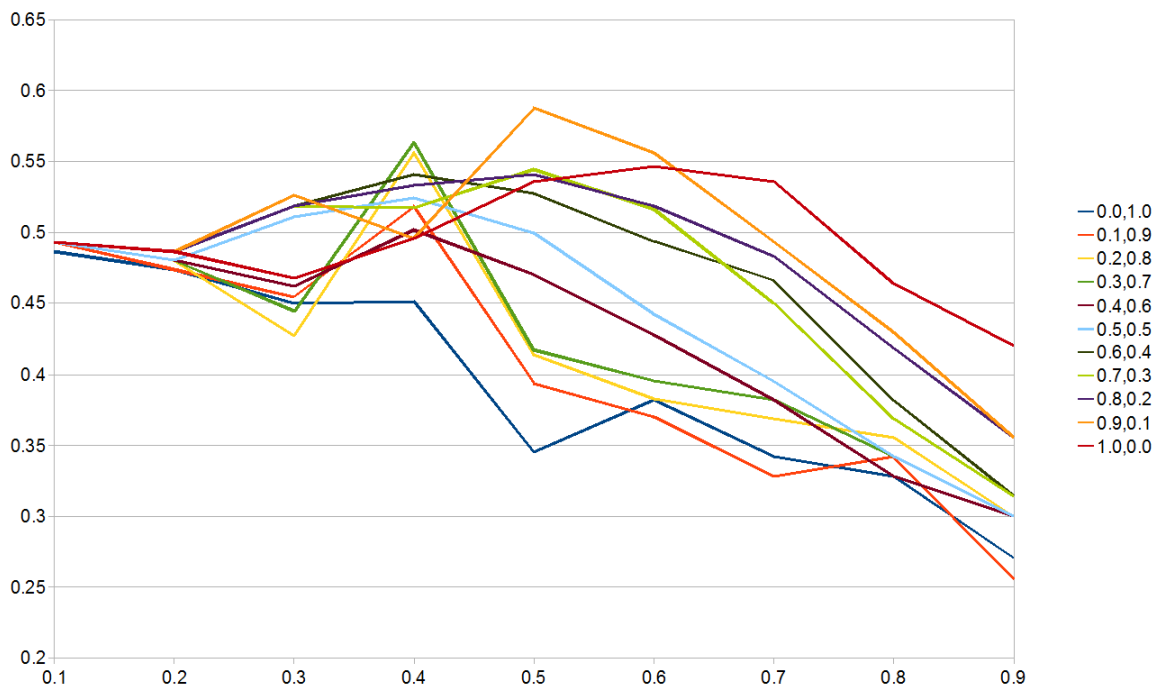


Figure A.7: Containment using partial documents. The results are very unstable, but weight sets favouring Equation 5.1 still outperform the ones favouring 5.2

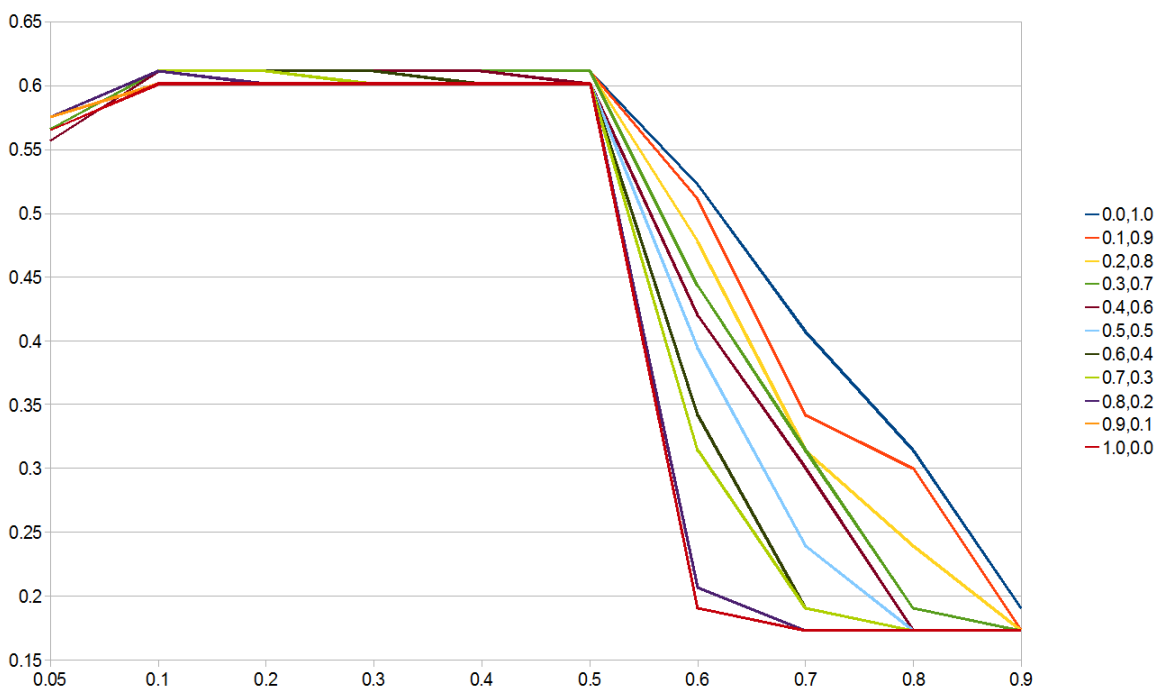


Figure A.8: Jaccard using n-grams. The flat area appear because of the high weight n have been given in Equation 5.4.

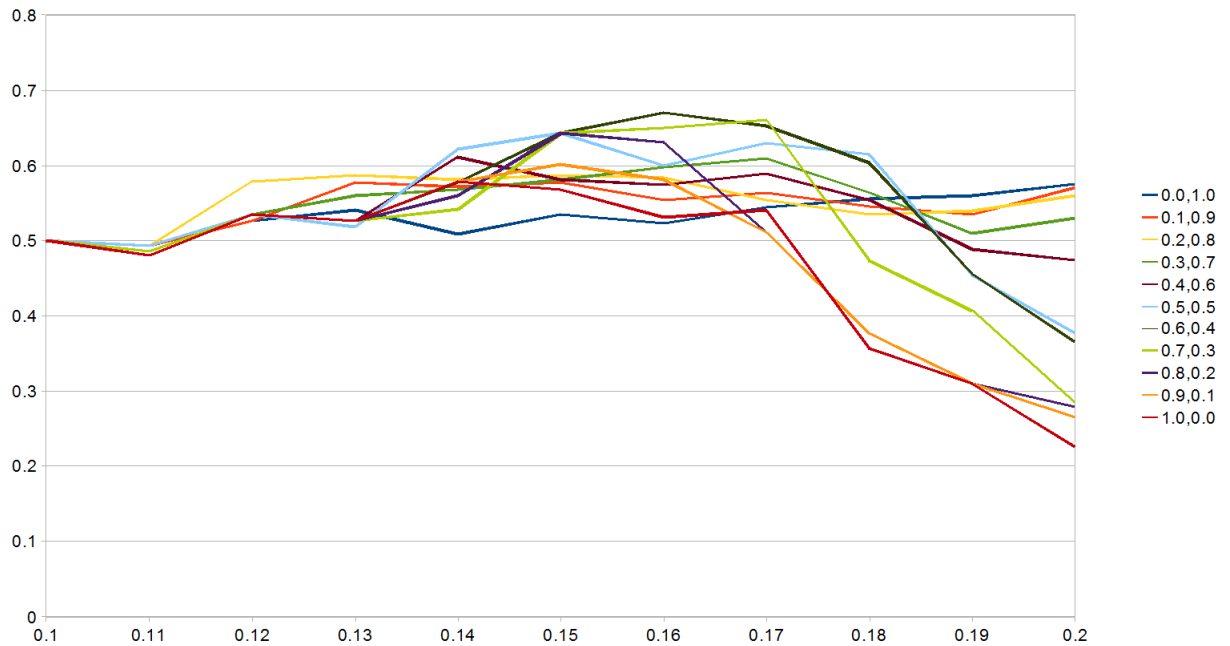


Figure A.9: Cosine in combination with n-grams. The results are less affected by the high weight of n . Generally, Equation 5.2 give the highest value, causing weight sets favouring this doing it better as the threshold get higher.

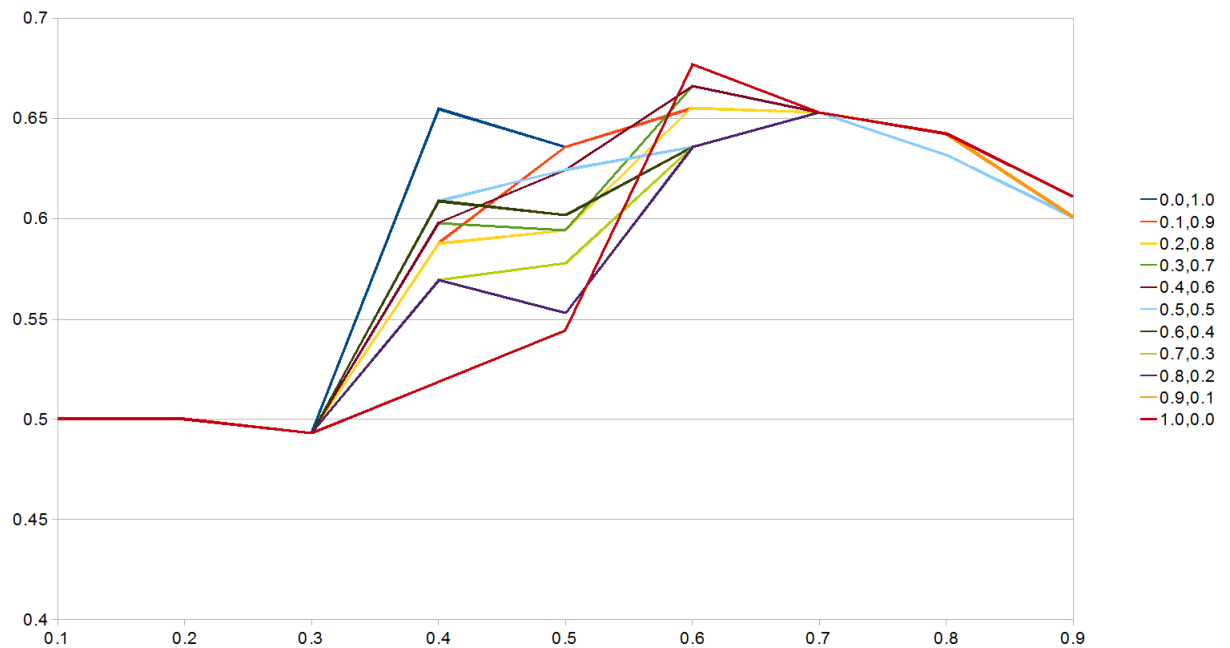


Figure A.10: Containment using n-grams. The impact of the high weight of n is not visual in the results graph until the threshold reach 0.7.