NTNU
Norwegian University of
Science and Technology

Knowledge discovery from large collections of unstructured information

Mateusz Siniarski

Master of Science in Computer Science
Submission date: June 2016
Supervisor: Pinar Öztürk, IDI
Co-supervisor: Erwin Marsi, IDI

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

NTNU – Trondheim
Norwegian University of
Science and Technology

Knowledge discovery from large collections of unstructured information

Mateusz Siniarski

Spring 2016

Master Thesis

Computer Science, The Intelligent Systems (AI) group
Department of Computer and Information Science
Norwegian University of Science and Technology

Supervisor: Pinar Öztürk

Co-Supervisor: Erwin Marsi

Abstract

The number of scientific publications is increasing by 3% per year, making it difficult for scientists to keep up with new research and to find relevant papers. As a result, time they could spent on research is used to stay up-to-date with the state-of-the-art in their fields. Wikipedia increasingly serves as a reliable source for up-to-date scientific knowledge. This fact is the main motivation for developing a knowledge discovery support system capable of retrieving relevant documents on a selected subject. This thesis focuses on state-of-the art solutions, aiming to support knowledge discovery from Wikipedia articles. While the majority of similar tools targets biomedical publications, this thesis focuses on Wikipedia pages related to marine science.

The four main components of the proposed system are: document retrieval, document filtration, information extraction and interactive search. The retrieval component extracts the core page content from the Wikipedia pages and removes all wiki markup, leaving only the plain text of the article. The document filtration component selects those articles related to marine science by using a machine learning solution called topic modelling, which uses statistical methods for finding topically similar documents. The information extraction component performs named entity recognition for geographical locations and marine species. In addition, it detects events involving variables that are increasing, decreasing or changing.

Topic models were evaluated against a gold standard that was created with the help of domain experts using a pooling method. The evaluation of document filtering indicated that the best performing topic model is Latent Semantic Analysis (LSA) configured with 500 topics, yielding an NDCG score of 0.70. This model was subsequently used to retrieve 4727 marine science related articles from an initial list of 22 seed articles.

The search interface is a single-page web application which provides faceted search. The Solr-based implementation allows retrieving documents by search terms and filtering them by facet, including location, species and changing variables. A visualisation method displays the extracted geographical locations as pinpoints on a map and presents changing variables in the text with colours indicating the direction of change.

Acknowledgment

I would like to thank my supervisors, Pinar Öztürk and Erwin Marsi for their patience, great help and guidance, in both the technical aspects of the project and the help during the thesis writing. I would also like to thank Murat V. Ardeland for the help on the marine science related subjects and for evaluating the data. Additionally, I would like to thank my family for their support during my studies.

Trondheim, 11th June 2016

Mateusz Siniarski

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Ocean-Certain is an EU project initialised by NTNU, with a main goal of improving understanding the impact of the food web and the biological pump process on future climate change.[1] It is a cooperation between 11 partners from Europe, Australia and Chile. The research is divided in multiple work packages. The most relevant work package for this project is WP1, having focus on data mining and knowledge discovery from scientific publications.

The motivation behind developing knowledge discovery systems is the rapid growth of scientific literature. Based on a study by Bornmann and Mutz (2015), the number of scientific publications has doubled in the period between 1980-2004, and is increasing by 3 percent each year. As a consequence of this growth, scientists need more time to stay updated with the state of the art in their field.

Solutions for literature-based discovery have so far been focused on biomedical publications. A method referred to as the ABC-pattern or Swanson linking has capabilities to discover knowledge based on finding relations of terms between unrelated documents [Swanson (1986)]. The principle of the method is finding an unpublished relation $a \rightarrow c$ based on relations $a \rightarrow b$ and $b \rightarrow c$ from two separate sources. A famous example was discovering the relation between Raynauds's disease and fish oil, where

[1] Ocean-Certain: `http://oceancertain.eu/`

one literature included information of Raynauds's disease and increased blood viscosity, and another literature had information of reduced blood viscosity and fish oil.

Current research in WP1 [Marsi et al. (2014)] involves using literature-based knowledge discovery from scientific publications in marine science. The work is based on analysing abstracts from scientific journal publications on climate change. The goal is to extract *quantitative variables*, describing events of *change*, *increase*, and *decrease* - e.g. "addition of labile dissolved organic carbon" where "addition of" is the trigger indicating the type of change (increase) and the rest is the variable to which the change is applied.

Combinations of events are then used to extract rules. The three types of rules used are *causal*, *correlation* and *feedback*. Casual rules describe events where a change A is causing change B (e.q. "diminished calcification led to a reduction in the ratio of calcite precipitation to organic matter production"). Correlation rules apply for events of which change A occurs together with change B (e.q. "when bacterial growth rate was limited by mineral nutrients, extra organic carbon accumulated in the system"). Feedback rules are used in events where changes A and B affect each other (e.q. "our model suggests the existence of a positive feedback between temperature and atmospheric CO2 content). Work so far shows that extraction of rules is possible for a small collection of abstracts, and the developed methods enable ongoing work involving a similar approach on much more text.

1.2 Objectives

The goal of this project is to create a system allowing marine scientist efficient access to relevant information on Wikipedia, as a first step in semi-automatic knowledge discovery from Wikipedia. The project involves the following objectives:

1. Extraction of article text
2. Selection of pages of interest
3. Finding the information of interest
4. Presenting the information

1.2.1 Extraction of article text

Wikipedia is currently the biggest online encyclopedia, containing over 5 million articles, and growing at a rate of approximately 20 000 articles a month[2]. The size of all English Wikipedia articles from August 2015 was approximately 55 GB (uncompressed). Such an amount of data makes information extraction time consuming.

The articles themselves are formatted using *Wiki Markup*.[3] It is a syntax used for writing the articles in Wikipedia, which are then parsed to HTML code when viewing articles in the web browser. For the purposes of this project, the HTML format has to be converted into plain text, because HTML cannot serve directly as input for linguistic analysis and information extraction.

1.2.2 Selection of pages of interest

The target audience for this project are marine scientists. This means that the content of the collection of articles should be limited to subjects only related to marine science. The main point of narrowing the document collection is to reduce the amount of time on browsing and verifying if the content is relevant. For example when a researcher looks for articles related to the marine life, the retrieved documents should not only be related to marine life, but the content must have a scientific background. The documents related to fictional content about marine life (e.g. sci-fi and fantasy) must be ignored.

1.2.3 Finding information of interest

Even when the available articles are related to a given field, they will still not contain any defined structure. As a result, the only way of finding information is by using search queries, which only find documents containing terms of the query. Therefore it is necessary to extract additional data from the text, allowing for searching the article's context, and not only the content itself. Examples of data that can be extracted from terms are the geographical locations and species. Extraction of this so-called en-

[2]Information about the size of English Wikipedia: `https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia`

[3]Details about Wiki Markup: `https://en.wikipedia.org/wiki/Help:Wiki_markup`

tities can allow categorisation of the document collection and use these categories to filter the search results e.g. articles related to the Norwegian Sea and salmon.

The data should additionally be described using an ontology i.e. a description of objects in text showing their relations to each other [Tran et al. (2007)]. An example is applying metadata to a geographical location, where it is possible to get information of the type of location and its coordinates. A more advanced example include a food web, indicating relation between multiple species.

1.2.4 Presenting the information

The final objective of the project is to find a way of representing the Wikipedia articles and additional extracted knowledge using a graphical user interface. Requirements would be to have an ability to search specific topics, not only by a query structured of terms, but also by selecting filters and categories provided by the semantics of the documents. Additionally it could also provide an automatic recommendation system of articles being related to current document.

1.2.5 Research questions

In order to guide our research objectives, the following more specific research questions were formulated:

1. How to access the Wikipedia articles?
2. How to extract article text?
3. How to find articles relevant to marine scientists?
4. How to extract terms for implementation of the faceted search?
5. How to present the results in a user interface?

1.3 Method

The methods used for the literature study included finding what knowledge discovery support system implementations were developed before and finding state-of-the-art

approaches related to the objectives and research questions of this thesis. The literature included the approaches for representing document collections, information retrieval, information extraction, and evaluations methods. The main source of the literature were scientific papers, but textbooks and web articles were also used.

The selection of software and tools was based on finding equivalents for the approaches found in the literature and how well they matched the chosen objectives and the research questions. Other aspects included the properties of each tool and how well could the tools be combined into a single system. In cases when similar software solutions were found, they were tested and evaluated to see which one performed the task closest to the expectation.

The evaluation method was to verify the multiple variants of the system, by pooling the outputs from every variant. These variants were evaluated by using quantitative evaluation against a gold standard, created by the domain experts. Other parts of the system were evaluated by using qualitative evaluation in form of manual error analysis.

The method for presenting the system was based on applying the implemented system and create a prototype, with the goal of checking if the solution works as planned. By putting the demo online, people cloud verify if it works as intended.

1.4 Structure

The report contains the following chapters:

- **Chapter 2** covers the document retrieval process, including accessing Wikipedia articles, extracting plain text and indexing plain text documents. It contains following sections:
  - Theoretical background about document retrieval from Wikipedia
  - Implementation of the document retrieval
  - Discussion about implementation and its benefits/issues
- **Chapter 3** is about the document filtration - a process responsible for extracting the documents relevant to marine science. The chapter has following structure:
  - Theoretical background

  - Implementation of topic models
  - Creation of gold standard
  - Evaluation of topic modelling
  - Discussion

- **Chapter 4** describes the implementation of the search interface. It includes the theoretical background on the information extraction, with main focus on the Named Entity Recognition, and the implementation of the search interface itself. The chapter has following structure:
  - Background
  - Implementation of Named Entity Recognition
  - Implementation of user interface
  - Discussion

- **Chapter 5** is the conclusion chapter, summarising the entire process by answering the research questions and suggests the future work

Chapter 2

Document retrieval

2.1 Introduction

This chapter describes how the retrieval of Wikipedia articles was implemented. The process included accessing the Wikipedia pages, extraction of plain text and indexing articles.

2.2 Background

2.2.1 Accessing Wikipedia articles

There are two methods of accessing the Wikipedia articles. The first method is based on using the MediaWiki API, which allows accessing individual articles from Wikipedia. The second method is to download so-called XML dump files, which are periodical back-ups of the Wikipedia content. They are available in different languages and published a couple of times a month.

Because of the computationally intensive operations that will be applied to the document collection, it is reasonable to choose the XML dump files. Local storage files will provide better performance and will not be dependent on the quality of a network connection. The initial dump version used for the implementation of the system was Simple English. The Simple English version of Wikipedia contains shorter versions of arti-

cles, which are meant to be easier to understand. These articles were used initially for testing, because the document collection contains only around 110 000 articles, while the English Wikipedia contains over 5 million[1].

The XML format of the articles is demonstrated by example 2.1. The most important parts for this project are the title and article text, where the title is stored as a string, and the text is formatted using Wiki Markup.

Code 2.1: Page structure of a Wikipedia dump file

```
<page>
   <title>Ocean</title>
   <ns>0</ns>
   <id>103595</id>
   <revision>
     <id>5180329</id>
     <parentid>5142406</parentid>
     <timestamp>2015-07-28T20:27:26Z</timestamp>
     <contributor>
       <ip>108.73.112.115</ip>
     </contributor>
     <comment>[[Plant]]</comment>
     <model>wikitext</model>
     <format>text/x-wiki</format>
     <text xml:space="preserve"> ...The article text in Wiki Markup... </
         text>
     <sha1>9qovmf6uxomxy4qtf2k7n19j4xrr13v</sha1>
   </revision>
 </page>
```

2.2.2 Wiki Markup

The articles accessed from the dump files are formatted using Wiki Markup[2], demonstrated in 2.2. The markup language provides description of elements including links, tables and sections. The markups functionality is meant as a tool for performing layout and rendering when a Wikipedia page is loaded. Some of the elements used in the example 2.2 include a blockquote (e.g. ´´´ocean´´´) used for separating a block of text (appearing as bold text), internal links (e.g. [[salt]]), used for navigating to the Wikipedia

[1] Size of Wikipedia `https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia`
[2] Wiki Markup syntax `https://en.wikipedia.org/wiki/Help:Wiki_markup`

page having the same title as the internal link and references (e.g. <ref>name=fishy</ref>), pointing to the list of references at the bottom of the page. Some of the Unicode characters in the markup text are escaped (e.g. ndash; < >) and need to be decoded before displaying.

2.2.3 Information Retrieval

Information Retrieval (IR) is the fundamental element of every search engine. The goal is to create an index of a collection of documents, for allowing fast retrieving of documents related to a search term. Without the IR, every search attempt would have to traverse entire document collection, which would be very time consuming and a waist of processing power.

A usual method to index a document collection is to use an *inversed index*. The method transforms the document collection into a *term-document matrix* (examples 3.1 and 3.2), which allows to retrieve a document by a term it contains (more detail about terms in section 3.2.1). The matrix contains a *term frequency weight* indicating how often the word occurs.

By using pure term frequency, a document having two instances of term will be twice as relevant then another document having only one instance. A method to solve this issue is to use a *Vector Space Model*, with a *TF-IDF* representation of the terms (more detail on Vector Space Model and TF-IDF in sections 3.2.3 and 3.2.3).

The basic evaluation techniques for IR systems are calculating *precision* and *recall* levels. Precision is defined as the fraction of retrieved documents which is relevant and recall is the fraction of the relevant documents which has been retrieved. More about evaluation methods in section 3.2.5.

2.3 Implementation

2.3.1 Extraction of the plain text

For further processing with NLP tools, it is necessary to extract the plain text of the articles. This goal is achieved by using a parser - a program able to convert text between

multiple formats. Because of the popularity of Wikipedia, there are many parsers[3] available, implemented in many programming languages and having many types of functionality. Two examples of parsers are the Bliki engine[4] and WikiExtractor[5]. Reason of presenting these two parsers is to compare their different approaches of extracting the plain text, and then deciding which one of them works best for the rest of the project.

Code 2.2: Fragment of Ocean article in Wiki Markup

```
An '''ocean''' is a large area of [[salt]] [[water]] between [[continent]]
    s. Oceans are very big and they join smaller [[sea]]s together.
...
Below the thermocline, temperature in the deep zone is so cold it is just
    above freezing - between 32 &amp;ndash; 37.4 F (0 &amp;ndash; 3 C).&lt
    ;ref name=fishy/&gt;
```

The Bliki engine is a Wikipedia parser based on Java. Its main purpose is to parse the markup into HTML, but it also has a possibility of converting into plain text. The implementation allows for both parsing the XML dump files directly and extracting the plain text from a single Wiki Markup formatted file. The extractor is based on jar libraries, which requires embedding into an existent Java project. Example 2.3 demonstrates the plain text extracted extracted from the Wiki Markup format shown in 2.2.

Code 2.3: Fragment of Ocean article extracted by Bliki engine

```
An ocean is a large area of salt water between continents. Oceans are very
     big and they join smaller seas together.
...
Below the thermocline, temperature in the deep zone is so cold it is just
    above freezing - between 32 &ndash; 37.4 F (0 &ndash; 3 C).
```

The WikiExtractor is a Python script used for extraction of the plain text or HTML from Wiki Markup formatted articles, stored in the XML dump files. For each "page" node in the XML, the script extracts the id of the article, the url of the Wikipedia page containing the article, and the plain text of article with the title in top line. This script is designed to run independently of other projects, and only requires setting up ar-

[3] Wikipedia parsers https://www.mediawiki.org/wiki/Alternative_parsers

[4] The Java Wikipedia API (Bliki engine) https://bitbucket.org/axelclk/info.bliki.wiki/wiki/Home

[5] WikiExtractor https://github.com/bwbaugh/wikipedia-extractor

guments including the path of the XML dump file and an output path. Example 2.4 demonstrates the plain text representation of an article.

Code 2.4: Fragment of Ocean article extracted by WikiExtractor

```
<doc id="103595" url="https://simple.wikipedia.org/wiki?curid=103595"
    title="Ocean">
Ocean
An ocean is a large area of salt water between continents. Oceans are very
     big and they join smaller seas together.
...
Below the thermocline, temperature in the deep zone is so cold it is just
    above freezing - between 32 - 37.4 F (0 - 3 C).
</doc>
```

The main differences between these approaches are that the Bliki engine allows accessing the articles during the extraction process, while WikiExtractor can only produce a hierarchy of text files containing the plain text of all articles. The Bliki engine does not include any serialization methods, but it can be used with custom serialization or as a partial step of processing. There are also differences in quality of the results, where the Bliki engine does not decode some of the escaped characters (e.g. – in example 2.3).

Because of multiple traversals required for analysis of the document collection, the natural choice was to use the serializable solution provided by WikiExtractor. This script uses a folder hierarchy of text files - each folder containing up to 100 text files and each file being approximately 1 MB large. An advantage of partitioning the document collection is a faster access to articles due to less file IO, as long there is an index containing information of the file location for each article. Finding the article text in this case requires only traversing approximately 1 MB of data, instead of a couple of gigabytes.

2.3.2 Indexing documents

The first attempt of implementing search on the extracted articles was done by traversing the entire document collection. It was performed by using the Bliki engine to extract the titles of the articles. Then, by using regular expressions, a match between a search query and a title would result in finding a document. It was meant as a naive attempt

to test the performance of accessing the document collection.

The problem was the huge number of articles in the document collection. For the Simple English Wikipedia the search time was around 5 seconds, but for the full English Wikipedia the process took over 10 minutes. A way to improve such long search times is to index the document collection.

By using Solr for indexing the document collection, the retrieval of documents could be performed immediately. Additional data added to the index are the corpus id of each article (required for identifying articles returned by the Topic Modeling 3.3) and location path pointing to the plain text version of the article (plain text extracted by WikiExtractor 2.3.1). The results were returned in a format similar to the XML used for the indexing. Search queries in Solr can either perform search in all fields, or by choosing a specific field with following notation: "field:query".

Solr is a search platform providing indexing of large document collections[6]. It is based on Lucene[7], an open-source information retrieval system, offering features as indexing, searching, spell checking, hit highlighting, and tokenization. The solution creates a server where the data will be indexed and stored. The indexing process requires a pre-formatted xml file, specifying the metadata and text to be stored. The example 2.5 demonstrates the data indexed from the Wikipedia articles, including the ID of the article, title, article text, URL to the Wikipedia page and a version number of the article.

Code 2.5: XML format required by Solr for indexing

```
<add>
  <doc>
    <field name="article_id"> </field>
    <field name="corpus_id"> </field>
    <field name="title"> </field>
    <field name="text"> </field>
    <field name="url"> </field>
    <field name="location"> </field>
    <field name="version"> </field>
  </doc>
</add>
```

After indexing the desired data, it can be retrieved by using queries. The example

[6] Solr `http://lucene.apache.org/solr/`
[7] Lucene `http://lucene.apache.org/core/`

2.6 shows a query where it searches for terms "ocean" and "acidification" occurring together in the text and the example 2.7 shows the first retrieved document corresponding to the query.

Code 2.6: Example of a query in Solr

```
text:(ocean acidification)
```

Code 2.7: Example of a document retrieved from Solr


```
{
    "article_id":2801560,
    "corpus_id":3879558,
    "title":"Ocean acidification",
    "text":"Ocean acidification\n\nOcean acidification is the ongoing
        decrease in the pH of the Earth's oceans...",
    "url":"https://en.wikipedia.org/wiki?curid=2801560",
    "location":"BH/wiki_39",
    "_version_":1533051034897743873
},
```


2.4 Discussion

2.4.1 Access to Wikipedia content

The selected approach was using the dump files for extracting the plain text and further analysis. The advantage of this approach is the consistency of data during processing, because there is no risk that parts of the data would be changed. Additionally the processing of data is limited only by the performance of the computer, and not the Internet connection if the API approach would be used.

2.4.2 Quality of extracted plain text

The major difference between WikiExtractor and the Bliki Engine is the cleaner output of the WikiExtractor - more unicode characters are decoded and most of the metadata are filtered out. The WikiExtractor is not a perfect parser, there are some documents which where metadata has not been removed e.g. HTML tags. Some parts of the project (e.q. variable change extraction) needed to be modified to avoid the parts of text ignored by the parser.

2.4.3 The indexed document collection

The Lucene powered Solr is a powerful tool for indexing and querying large document collection, but some limitations were found. A problem occurred while indexing entire English Wikipedia corpus (10 GB of plain text), causing the server instance to crash during indexing. The problem was occurring because of attempting to index and store the text of each document. After testing multiple configurations and versions of Solr, a successful workaround has been found. Instead of storing the entire text, only the path to the extracted plain text file has been stored to the index. Since it was possible to index the text without storing, there was no limitation in querying the text content of the documents in collection. Retrieving of the text itself were done outside of Solr, by locating the text file and traversing it until the article was found. The performance was not noticibaly weakened, because the size of the text files was only approximately 1 MB.

This problem occurs only when indexing full Wikipedia corpus. The further parts of the projects also uses Solr for indexing smaller sets of documents, where all metadata including the plain text are stored.

Chapter 3

Document Filtering

3.1 Introduction

The document filtering chapter focuses on how to extract documents related to the requested topic, which in this case is the marine science. The chapter starts with theoretical background about methods of pre-processing text of the documents, how to represent them using the vector space model, the main principles of topic modelling and the evaluation methods used in information retrieval. Further it explains the implementation of the topic modelling and its evaluation. The chapter ends with the discussion of each mentioned part.

3.2 Background

3.2.1 Document representation

Document features

Text mining requires creating a representation of the unstructured text for further processing. This kind of representation is often refered to as a structured text, which contains a set of features. Based on Ronen Feldman (2007), the four most used document features are *characters*, *words*, *terms* and *concepts*.

Characters Using *characters* as document features results in a representation of all characters or a set of characters in a given document. Example of a character representation method is the *bag-of-characters* approach, where the characters are analysed without positional information (i.e. order is ignored) by their frequencies in a document or set of documents. By adding positional information, sequences of multiple characters (e.g. bigram, trigram) can be analysed in a similar manner.

However, it is mentioned by Ronen Feldman (2007), that character based features can be unwieldy in usage of text mining, because of the lack of optimisation of the feature space. Using characters only will also cause loss of the semantic meaning of the document, because most characters in isolation do not carry any meaning. Exceptions might be signs like exclamation mark (!) or ampersand (&) which can provide additional expressiveness to the text.

Words Using *words* as document features is called a *bag-of-words* representation. It is generated by extracting all words from the document collection and storing them in a dictionary. Bag-of-word representations of a document can represent a document by for example calculating the count of each word that occurs in the document. A downside of such approaches is that the word order is lost, so e.g. "Venetian blind" cannot be distinguished from "blind Venetian"

According to Ricardo Baeza-Yates (2011), if a word appears in over 80 percent of the documents of the collection, it will have no value for analysis. Those words are referred to as stopwords, and are removed to give more accurate results. Additionally it reduces dictionary size and can improve indexing structure. More detail on stopwords in section 3.2.2.

Following example demonstrates how the BOW representation is created based on a list of simple documents:

- Alice likes to read books. Bob prefers to watch movies.
- Alice also likes to watch movies.

A dictionary will be created based by extracting all unique words from the document collection. Each document is then represented as a set of occurrences of each

Table 3.1: Word-document matrix demonstrating bag of words representation of documents using term frequency weight

Dictionary	Document 1	Document 2
Alice	1	1
also	0	1
Bob	1	0
books	1	0
likes	1	1
movies	1	1
prefers	1	0
read	1	0
to	2	1
watch	1	1

word. There are different strategies available, with possibilities of using a Boolean representation (word exists in document), term frequency (how many times word occurs) or more advanced weight strategies (more detail in section 3.2.3). The document collection can be presented as a word-document matrix (Table 3.1).

Terms *Terms* are an augmentation of the words feature space, where multiword expressions are considered as a single feature. For example when considering a term "The White House", a single word representation would lose the semantic meaning of the residence of the American president.

Concepts *Concepts* focuses on defining categories of document features. Categorisation allows defining relations between features that can be used for example to find terms of similar meaning or defining a hierarchy a term belongs to. An example of such concept is a geographical location, where a location has multiple names (e.g. Holland and The Netherlands) and has a hierarchical relation (e.g. Holland is part of Europe). These properties allow concepts to be retrieved without referring to them by the term literally.

3.2.2 Document preprocessing

Representation of documents can be improved by first modifying the document itself. Such modifications often result in reduction of the dimensionality, smaller dictionar-

ies and improved retrieval result. The three most common approaches mentioned by Andreas Hotho (2005) are stop word removal, lemmatization and stemming.

Stop word removal

Most languages (including English) have a property where particular words occur very often in each sentence or document. When a search query contains some of the most frequent words, this might lower the precision, because all documents contain those words. These words having this property are called *stop words*, and many text mining applications remove those words, because it provides better precision.

A typical way of performing removal of stop words is to compare the documents in the collection with a list of all stop words, and then removing all occurring words in the list. Another approach is to look up statistics of the document collection and remove words that occurs in most of the documents.

Here are some of the stop words occurring in English[1]:

"*a, an, and, are, as, at, be, but, by, for, he, if, in, into, is, it, no, not, of, on, or, she, such, that, the, their, then, there, these, they, this, thus, to, was, will, with*".

An example of stop word removal can be demonstrated on following sentence:

"A swimmer likes swimming, thus he swims."

Each word is compared with the stop word list, and if there is a match, the word is removed. The result of stop word removal is shown by following sentence:

"swimmer likes swimming, swims."

Lemmatization

Lemmatization is a process which computes the lemma of a word. A lemma is also referred as a canonical form of a word, which means that words of different inflected forms will be treated as a single form. For nouns it is common to use singular form, for verbs infinitive tense.

To achieve lemmatization, a system must be able to identify the form of each word. This process is considered as time consuming and not perfect. An example can be shown by using the sentence below. The assumption is that the word "swimming" is

[1] List of common stop words in English `http://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html`

detected as a noun, but a lemmatization software could also detect it as a verb, which in this case would be incorrect.

"A swimmer likes swimming, thus he swims."

Each word is changed to its canonical form:

- likes → like
- swims → swim

Swimmer and swimming are nouns that are already in their canonical form. Result is the sentence:

"A swimmer like swimming, thus he swim."

The process allows for finding a document containing the lemma by using all the variants of the term as query. For example, a search term "swam" would return the document from the example above, because the verb would be converted to its singular form "swim" before the search.

Stemming

There is often a case where multiple words have the same stem and different prefix and/or suffix. Assumption is that words having the same stem have similar meaning, thus should be handled as one term. In a normal bag-of-words representation those words would appear as distinct. *Stemming's* task is to extract the stem from the surrounding prefixes or postfixes. As result, the frequency of words having same stem will increase.

The stemming process can be demonstrated on following sentence:

"A swimmer likes swimming, thus he swims."

A usual approach is to remove suffixes of type "-s" or -"ing". This will result in changes:

- likes → like
- swimming → swim
- thus → thu
- swims → swim

Table 3.2: Table demonstrates term-document matrix

	d_1	d_2	d_3	d_4	d_5
t_1	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$	$w_{1,4}$	$w_{1,5}$
t_2	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$	$w_{2,4}$	$w_{2,5}$
t_3	$w_{3,1}$	$w_{3,2}$	$w_{3,3}$	$w_{4,4}$	$w_{3,5}$
t_4	$w_{4,1}$	$w_{4,2}$	$w_{4,3}$	$w_{4,4}$	$w_{4,5}$
t_5	$w_{5,1}$	$w_{5,2}$	$w_{5,3}$	$w_{5,4}$	$w_{5,5}$

The result is the following sentence:

"A swimmer like swim, thu he swim."

As demonstrated, simple stemming techniques can be too aggressive, where the word "thus" has been incorrectly stemmed. This means that stemming systems must have additional steps considering if stemming can be correct.

3.2.3 Vector Space Model

The *Vector Space Model* (VSM) [Sparck Jones (1972)] is often used as a document representation for text mining. The representation is based on a term-document matrix, where rows represent terms and columns represent the documents. Each element in the matrix contains the weight for term i in document j. The weight's task is to define the importance of each term in the document.

Term weights

Simplest way to represent a term weight in a document is to count the number of times a term is occurring in a document. This is also called the raw term frequency. Main problem of using raw term frequency for describing a document is that the relevance does not increase proportionally with term frequency. For example, a document having 10 occurrences of a term will not be 10 times as relevant as a document having only 1 occurrence.

A solution to this issue is using a logarithm of raw term frequency. A logarithmic representation will reduce the impact of multiple term occurrences. Such approach will focus on the fact of a term being present in a document and the frequency itself will be less relevant, but still existent.

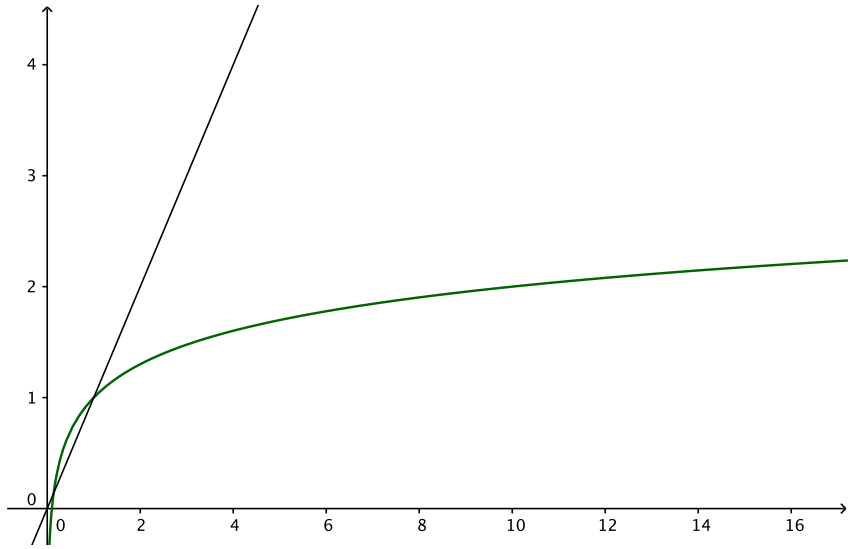

Figure 3.1: Comparison between linear and logarithmic representation of the term frequency

$$tf_{i,j} = \begin{cases} 1 + log(f_{i,j}) & for f_{i,j} > 0 \\ 0 & for f_{i,j} = 0 \end{cases} \tag{3.1}$$

Equation 3.1 demonstrates how the logarithmic term weight is set. Reason for using two functions for this task is that a logarithm can never reach value 0. 1+ in upper function is used because $log(1)$ equals 0 and would not represent terms occurring only once. Figure 3.1 presents comparison between a linear function and the logarithmic function used for term weighting.

Inverse Document Frequency

Document collections usually contain terms which are present in majority of the document. Using those terms will result in high recall of documents, but also in low precision. This is solved by giving higher weight to terms that are less frequent in the document collection. This weight is called *Inverse Document Frequency* (IDF) [Sparck Jones

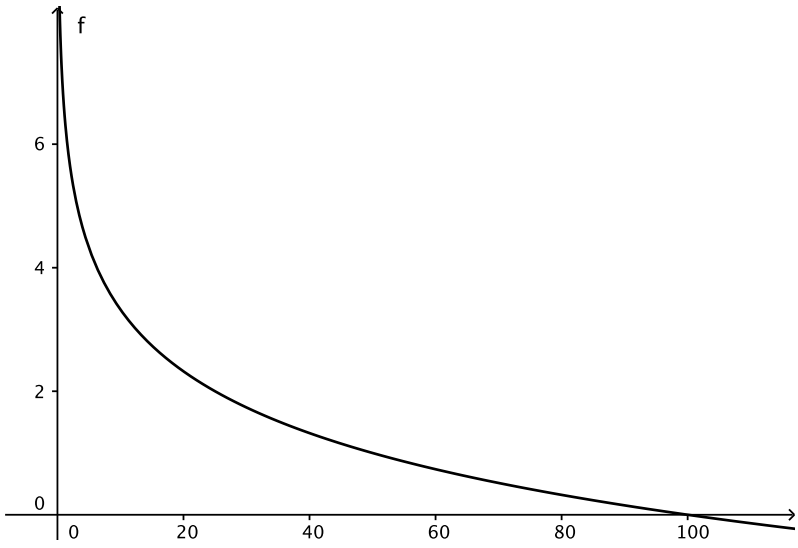

Figure 3.2: Graph presenting IDF with a corpus containing 100 documents.

(1972)].

$$idf(i) = log\left(\frac{N}{n_i}\right) \tag{3.2}$$

Equation 3.2 represents how IDF is calculated. It requires a number of all terms in the document collection N and a number of documents containing term i (n_i). The equation is valid only when term i is part of the document collection. As indicated by Figure 3.2 (using example of 100 terms in document collection), terms occurring less often get higher ranking then words occurring much more often. When a term exists in all documents, the IDF is set to 0.

TF-IDF

One of the best known approaches in term weighing is using *TF-IDF* (Salton and Yang (1973)), which is a combination of term frequency and inverse document frequency. Such combination results in increased weight for a term occurring multiple times within

Table 3.3: Sample of topics extracted from journal Science [Blei (2012)]

Topic 1	Topic 2	Topic 3	Topic 4
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	stains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

a document, as long as it is rare in the document collection.

$$tfidf(i,j) = tf_{i,j} \times idf(i) = \begin{cases} \left(1 + log(f_{i,j})\right) \times log\left(\frac{N}{n_i}\right) & for f_{i,j} > 0 \\ 0 & for f_{i,j} = 0 \end{cases} \tag{3.3}$$

3.2.4 Topic Modelling

Topic modelling allows to discover the hidden topics of the documents. Large document collections like Wikipedia, contain millions of articles, each of them containing information about multiple topics. A topic modelling tool will analyse each term in the vocabulary from the document collection and attach it to a specific topic. Each topic is then represented as a cluster of terms corresponding to it. Table 3.3 demonstrates an example of topics extracted from the journal Science by Blei (2012). The topic model is not able to label the category by itself, but it is clearly noticeable that the terms in each column belong to the same topic.

The approach used for creating topics is based on documents represented in a Vector Space. An advantage the topic modelling gives over VSM is using the extracted hidden topics for measuring document similarity, while the BOW representation of VSM can only rely on the occurrence rate of terms for each document. For example, when a

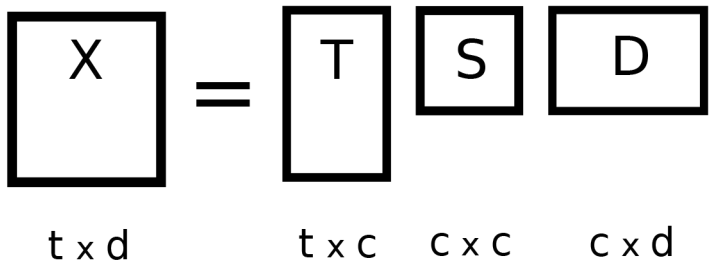

Figure 3.3: Graphical represantation of Singular Value Decomposition in LDA

search query contains the term "model", and a document contains the term "simulation", there is no relation in the VSM, but the topic model will find the relation with the topic 4 in table 3.3.

Two most popular algorithms used for this task are *Latent Semantic Analysis* [Deerwester et al. (1990)] and *Latent Dirichlet Allocation* [Blei et al. (2003)]. The common properties of those two approaches are that they are using documents represented as a VSM as input and they are designed to produce a predetermined number of topics. Both of them are also unsupervised, which means that they do not need any training data to learn their tasks. Differences lie in the computation methods of each algorithm, where LSA uses linear algebra and Singular Value Decomposition, while LDA uses a generative probabilistic approach.

Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a method proposed by Deerwester et al. (1990). The method is based on a document collection represented as a Vector Space Model (VSM) with weights being assigned using TF-IDF. LSA is a method based on linear algebra and uses a technique called Singular Value Decomposition (SVD).

$$X = TSD^T \tag{3.4}$$

Based on equation 3.4, matrix X is splited into a set of matrices T, S and D. Each matrix represents the following:

- X - Term-document matrix of original VSM, generated from the document collec-

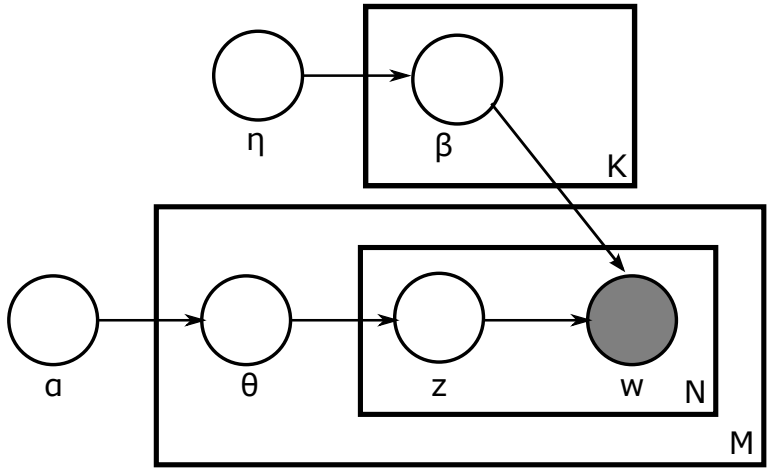

Figure 3.4: Graphical model represantation of LDA

tion.

- T - Terms over "concepts" matrix
- S - Variation of each "concept" sparse matrix
- D - Documents over "concepts" matrix

The "concepts" of SVD are responsible of setting the desired number of topics for a particular representation. This means that the matrix T is the representation of terms over topics, i.e. is the topic model of the document collection. In practice, only the matrix T is used in topic modelling, while other matrices can be disregarded.

Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a topic modelling method developed by Blei et al. (2003). The approach is a generative probabilistic model, where a topic is initialized randomly to each word of the document collection.

Figure 3.4 shows a graphical model of LDA. The upper box represents the topics of the document collection. The lower inner box represents words in a document and the outer box the documents within the corpus. Circles are representing following variables:

- α is the parameter of the Dirichlet prior on the per-document topic distributions
- η is the parameter of the Dirichlet prior on the per-topic word distribution
- β is the word distribution within a topic K
- θ is the topic distribution per document within corpus M
- z is the topic assigned to word w
- w is a word within document N

Variables marked in gray are the visible variables, which are the only ones that can be accessed directly. The white variables are defined as hidden, which need to be created by the LDA. This means that the only visible variables are the words, and the rest must be generated.

The general algorithm for LDA will first generate random topics for every word. The number of topics must be defined manually. The next step is to iterate through every word of every document in the corpus and reassign the word's topic. The entire process can be performed multiple times, based on a desired number of iterations.

An example of a approach used for reassigning the topics is Gibbs Sampling (A simplification presented by Mimno (2012)). It will use values stored by variables containing terms over topics (β) and topics over documents (θ). For each word iterated by LDA, the assigned topic will be removed from the word. Additionally occurrence of topics in θ and occurrence of words per topic in β will be updated. The selection of topic will be performed by using two values: the frequency of topics in current document and the frequency of current word for each topic. Topics will be selected based on the highest product of those values.

3.2.5 Evaluation methods

This section will explain evaluation methods used in the information retrieval.

Precision and Recall

Precision and *recall* are the fundamental evaluation types in IR. These are based on two parameters: the number of relevant documents in a corpus R and the number of

retrieved documents from corpus A. Precision is defined as the fraction of retrieved documents which is relevant (Equation 3.5) and recall is the fraction of the relevant documents which has been retrieved (Equation 3.6) [Ricardo Baeza-Yates (2011)].

$$precision = \frac{|R \cap A|}{|A|} \tag{3.5}$$

$$recall = \frac{|R \cap A|}{|R|} \tag{3.6}$$

The following corpus is used for demonstrating how precision and recall are calculated:

$$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}$$

A query is used for retrieving following documents:

$$A = d_3, d_6, d_7$$

While the following documents are relevant to the query:

$$R = d_2, d_6, d_7, d_9$$

The intersection between relevant and retrieved documents:

$$R \cap A = d_6, d_7$$

The size of each set are then used for calculating precision p and recall r:

$$p = \frac{|R \cap A|}{|A|} = \frac{2}{3} = 0.67$$

$$r = \frac{|R \cap A|}{|R|} = \frac{2}{4} = 0.50$$

Interpolated Average Precision

According to Manning et al. (2008), precision and recall are evaluations appropriate for unranked sets of documents. Modern search engines use ranked documents, where

Table 3.4: Example of a 11-point average interpolated precision

Recall	Interpolated Precision
0.0	0.73
0.1	0.67
0.2	0.61
0.3	0.58
0.4	0.49
0.5	0.46
0.6	0.42
0.7	0.38
0.8	0.35
0.9	0.33
1.0	0.30

the retrieved documents are sorted descending by the relevance. *Interpolated Average Precision* is a method for evaluating ranked document sets using precision at a number of recall levels (usually 11 points between 0-1), where the goal is to see how the precision changes when recall increases. The usual behaviour is a decrease of precision with increase of the recall (Table 3.4). These results are often shown as a graph (Figure 3.5).

Mean Average Precision

The average interpolated precision produces an informative and visual representation of the model, but the modern evaluation methods are focused on evaluating the IR system using a single value. *Mean Average Precision* (MAP) is one of the most used evaluation methods [Manning et al. (2008)], where the average precisions are calculated for each query and retrieved document. The m_j in formula 3.7 represents the length of the set of relevant documents for query Q_j and R_{jk} is the k-th retrieved document.

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision(R_{jk}) \tag{3.7}$$

Based on example 3.4 which includes one query, the MAP is calculated as follows:

$$MAP = \frac{0.73 + 0.67 + 0.61 + 0.58 + 0.49 + 0.46 + 0.42 + 0.38 + 0.35 + 0.33 + 0.30}{11} = 0.48$$

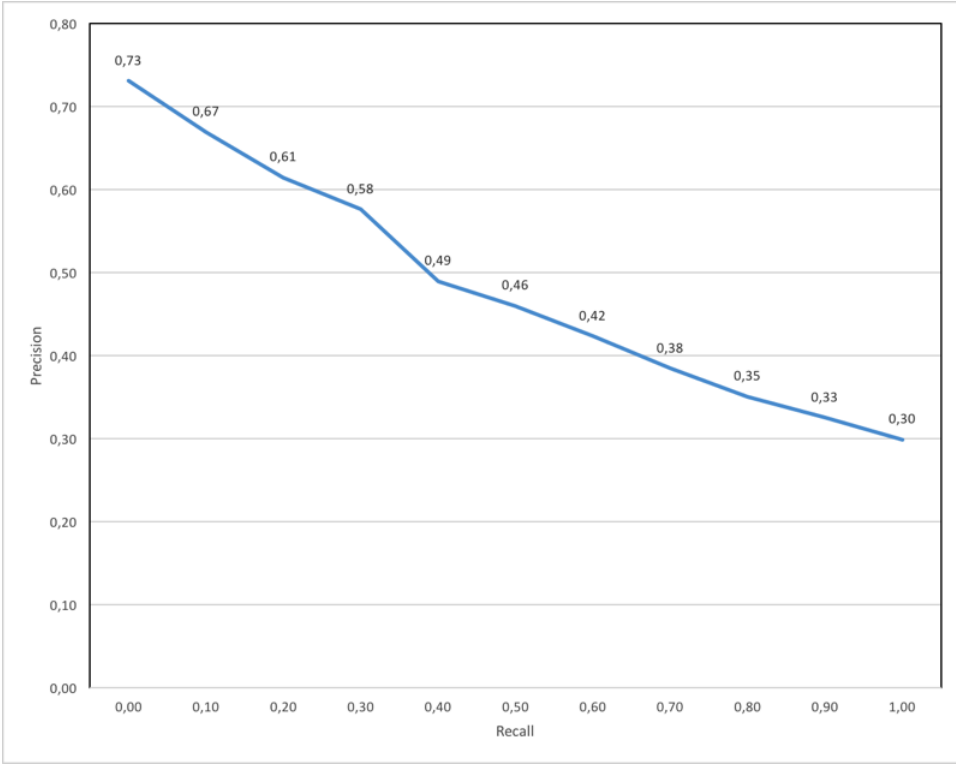

Figure 3.5: Graph of the average interpolated precision from table 3.4

Discounted Cumulative Gain

Discounted cumulative gain (DCG) is a evaluation method that works at the same principle as the MAP, but it is designed for non-binary relevance [Manning et al. (2008)]. A popular approach is to use the normalised version of DCG - NCDG, which sets the ranking between 0-1. The formula 3.8 shows the similarity to the MAP, with the difference of $R_{(j,m)}$ which is the relevance of document d_m in query Q_j, and Z_{kj} which is the normalisation factor. The relevance will be lower with a lower rank. This is achieved by the divisor $log_2(1+m)$ where m is the document's position.

$$NDCG(Q,k) = \frac{1}{|Q|}\sum_{j=1}^{|Q|} Z_{kj} \sum_{m=1}^{k} \frac{2^{R(j,m)}-1}{log_2(1+m)} \tag{3.8}$$

$$F(m,(R(m))) = \frac{2^{R(m)}-1}{log_2(1+m)} \tag{3.9}$$

Table 3.5 presents a simplified example of NDCG, where a single term query is used for retrieving documents. The column m is the index of the document, $R(m)$ is the relevance level and $F(m,R(m))$ is the part of DCG defined by formula 3.9. The decreasing score can be observed in the documents with relevance "2", where the difference at indexes 1 and 4 is 57%. The DCG score is 7.55. For calculating the NDCG, the ideal DCG score must be found. It is done by calculating the DCG where the documents are sorted by the relevance level. The IDCG has score of 8.76. The NDCG is calculated with equation 3.10 and has score of 0.86.

$$NDCG = \frac{DCG}{IDCG} = \frac{7.55}{8.76} = 0.86 \tag{3.10}$$

3.3 Implementation of topic models

Topic Modelling was performed by using Gensim [Řehůřek and Sojka (2010)], which is a tool based on Python and NumPy [2]. The topics were extracted using all of the models supported by Gensim, including: Latent Semantic Analysis, Latent Dirichlet Allocation, Random Projections and Hierarchical Dirichlet Process. The goal of using topic mod-

[2] Gensim `https://radimrehurek.com/gensim/`

Table 3.5: Example of NDCG evaluation with the retrieved documents on the left and the ideal order on the right

m	$R(m)$	$F(m,R(m))$
1	2	3.00
2	1	0.63
3	0	0.00
4	2	1.29
5	1	0.39
6	2	1.07
7	0	0.00
8	0	0.00
9	1	0.30
10	2	0.87
DCG		7.55

m	$R(m)$	$F(m,R(m))$
1	2	3.00
2	2	1.89
3	2	1.50
4	2	1.29
5	1	0.39
6	1	0.36
7	1	0.33
8	0	0.00
9	0	0.00
10	0	0.00
IDCG		8.76

elling in this application was to programmatically find a list of documents similar in topics to a smaller list of manually selected seed documents.

The process started with creating a Vector Space Model representation of the document collection. The process required traversing the entire document collection twice: first for generating the dictionary, second for creating the corpus. The source of documents used for the preprocessing was the plain text document collection extracted by WikiExtractor (section 2.3.1).

The dictionary was created by parsing every word from each article. For reducing the dictionary size, stop words and terms occurring only once were removed, also because these words are not informative in building topics. NLTK[3] is a natural language processing package in Python and it was used for removing the stop words.

The following step was to create the corpus of documents. It was done by counting the number of times a term occurs per document. Both dictionary and corpus are serialized and stored for later use. One of the desires was to test both bag-of-words and TFIDF vector space model, so an additional step was taken to compute both corpus representations.

The final step was generating the topic models. The input needed for the topic model generator is the corpus, dictionary and number of topics (except of HDP models). The recommended approach to transformation of topic models in Gensim in-

[3]NLTK `http://www.nltk.org`

cludes two steps, a so-called "double-wrapper" of the initial corpus. The initial transformation mentioned above creates a wrapper around the initial corpus, where the conversions are performed on-the-fly. In case of using large document collections, the transformation process can be a time consuming process, especially when there is a necessity of multiple iterations. The solution is the second wrap of the corpus, which serialises the topic model.

Multiple variations of topic models has been created for evaluating which model provides the best document filtering. The variations include the number of topics, format of the corpus and the multiple types of topic models. As mentioned earlier, Gensim supports four topic models, where LSA, LDA and RP require a selected number of topics and HDP can select the number of topics automatically. There are two variants of the corpus, one with bag-of-words representation and a second using TF-IDF. Furthermore, the number of topics ranged over 10, 20, 50, 100, 200 and 500. This gives a total of 38 combinations (Table 3.6).

The process of finding similar articles works by inputting the text of a selected article into a topic model. Gensim will then return a similarity score for every document between 0 and 1 – higher value means more relevant (Example showing the similarity rate in code 3.2).

3.4 Creation of gold standard

After the topic models were generated, it was required to verify which model provides the best representation of the anticipated results. The verification was performed by first creating a gold standard - a reference created by the domain expert, containing a set of documents assigned a graded level of relevance [Manning et al. (2008)].

The first step of the evaluation involved using a set of initial seed articles. A domain expert provided a list of 22 Wikipedia articles related to marine science:

> Algal bloom, Aquatic ecosystem, Biological pump, Biomass (ecology), Blue carbon, Carbon cycle, Carbon sink, Cyanobacteria, Dissolved organic carbon, Global warming, Iron fertilization, Limiting factor, Marine ecosystem, Ocean acidification, Ocean chemistry, Ocean fertilization, Oceanography, Oxygen minimum zone,

Table 3.6: List of all topic models combinations

Model	Corpus	Topics
LDA	BOW	10
LDA	BOW	20
LDA	BOW	50
LDA	BOW	100
LDA	BOW	200
LDA	BOW	500
LDA	TFIDF	10
LDA	TFIDF	20
LDA	TFIDF	50
LDA	TFIDF	100
LDA	TFIDF	200
LDA	TFIDF	500
LSA	BOW	10
LSA	BOW	20
LSA	BOW	50
LSA	BOW	100
LSA	BOW	200
LSA	BOW	500
LSA	TFIDF	10
LSA	TFIDF	20
LSA	TFIDF	50
LSA	TFIDF	100
LSA	TFIDF	200
LSA	TFIDF	500
RP	BOW	10
RP	BOW	20
RP	BOW	50
RP	BOW	100
RP	BOW	200
RP	BOW	500
RP	TFIDF	10
RP	TFIDF	20
RP	TFIDF	50
RP	TFIDF	100
RP	TFIDF	200
RP	TFIDF	500
HDP	BOW	-
HDP	TFIDF	-

Phytoplankton, Plankton, Trophic state index, Zooplankton

These articles were then used to find similar articles. An ideal way to generate the gold standard would be to evaluate each topic model and each article individually. Problem with such approach is the tremendous amount of time needed for verification. While selecting 200 similar articles per seed article, the verification would require 167200 ($22 \times 38 \times 200$) checks. To decrease the evaluation effort, a process of *pooling* has been applied to the similar articles, which selects the top-k retrieved documents from multiple IR systems [Manning et al. (2008)]. The initial step required merging together the results for all seed articles and topic models. The method (Algorithm 1) used for merging was based on the assumption that since the seed articles are related to each other, many articles will be present for multiple seed articles. The merging process first generated a list of all similar articles for each topic model. To compensate for the different score ratios for each model, the scores have been normalised by using feature scaling (equation 3.11) [Juszczak et al. (2002)]. Final step consisted of finding all instances for each article, and adding the scores for each instance. The result was a list containing unique similar articles, sorted descending by the summation score.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.11}$$

Algorithm 1 Pooling algorithm returning a list of n most relevant articles based on topic models and seed articles

```
1:  function POOLING(Models, SeedDocuments, limit=200)
2:      PoolDocs ← empty
3:      for Model in Models do
4:          SimilarityDocuments ← empty
5:          for Doc in SeedDocuments do
6:              SimilarityDocuments ← Model(Doc)
7:          SimilarityDocuments ← normalized(SimilarityDocuments)
8:          for SimDoc in SimilarityDocuments do
9:              PoolDocs[SimDoc].score ← PoolDocs[SimDoc].score + SimDoc.score
10:     return sorted(PoolDocuments).range(0, limit)
```

The verification of article relevance was achieved using Google Forms[4]. The form included 200 articles, each one presented with a title, a link to the Wikipedia page and

[4] Google Forms `https://www.google.com/forms/about/`

a short summary. The verification was performed by choosing a relevance level defined by three options: “Yes”, “Maybe” and “No”. Figures 3.6 and 3.7 display the forms intro page and examples of articles to be verified.

By using Apps Script[5] and other APIs provided by Google, it was possible to programmatically generate a form from existing topic models. The process involved two steps. The first step executed locally, consisting of creating spreadsheets containing the article similarities from topic models. The second uploads the spreadsheets to Google Drive and runs the Apps Script for generating the form.

The results of the Google form were automatically collected in a Google spreadsheet, containing one respondent per row. The form had multiple responses, causing the need of generalisation of the results to create the gold standard. The chosen method was calculating the average response for each article. The answers where assigned to the following numeric values:

- Yes = 3
- Maybe = 2
- No = 1

The frequencies were calculated for each answer and the final relevance R score was calculated using equation 3.12. The relevance presents a decimal value, but the evaluation needed a natural number representation, which was achieved by rounding the relevance. The result was values in a range of 0-2.

$$R = round\left(\frac{3n_{Yes} + 2n_{Maybe} + n_{No}}{n_{Total}}\right) - 1 \tag{3.12}$$

For further analysis of the responses, a inter-annotator agreement needed to be measured. This kind of measure indicates the proportion of agreement per item (i.e. how much the judges agree for each evaluated article) and a value indicating agreement for the entire form. The method used was the Fleiss’ Kappa measure [Fleiss and Cohen (1973)], because it is designed for evaluating data sets having multiple judges and multiple categories.

[5]Apps Script `https://developers.google.com/apps-script/`

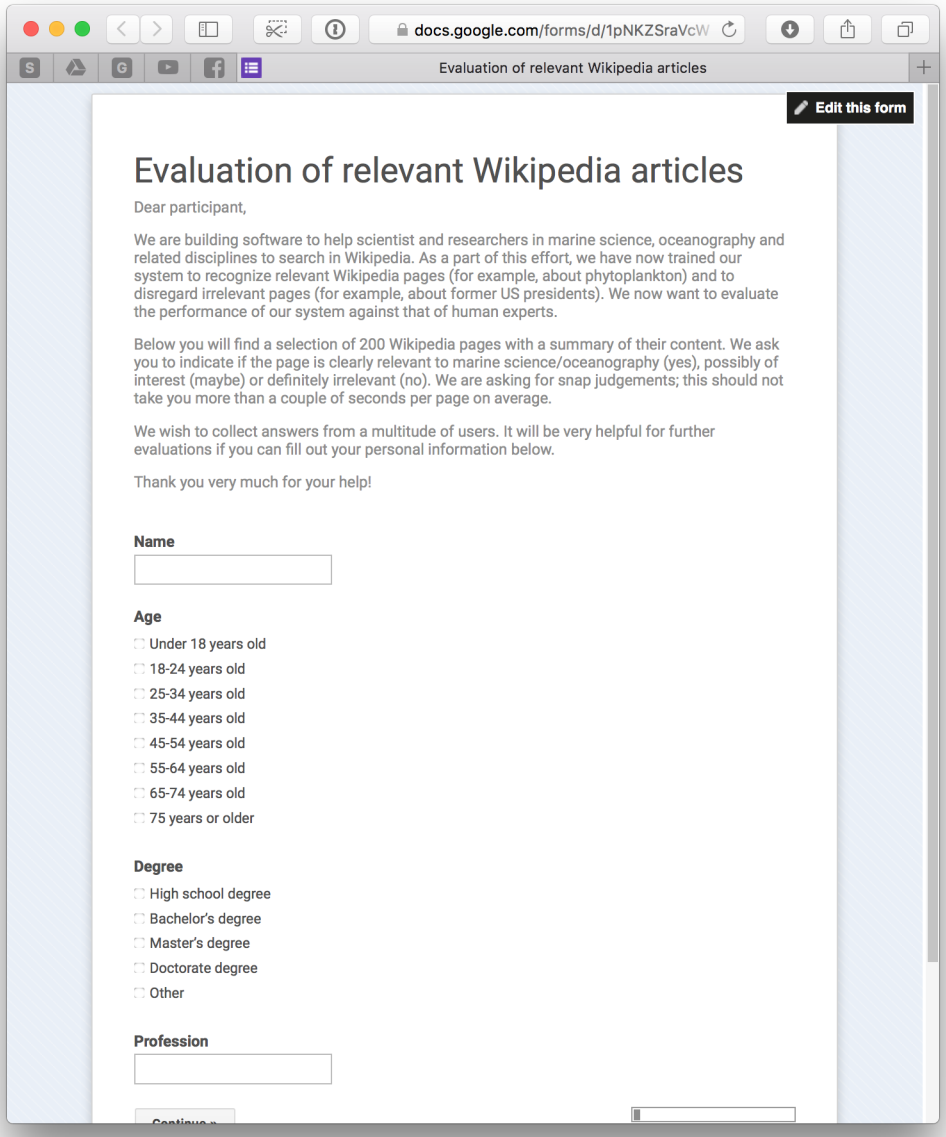

Figure 3.6: Introduction page of the Google form

Figure 3.7: Verification of articles in the Google form

Table 3.7: Results from the collected data

Relevance	Evaluation	Percentage
Yes	107	53,5 %
Maybe	67	33,5 %
No	26	13,0 %

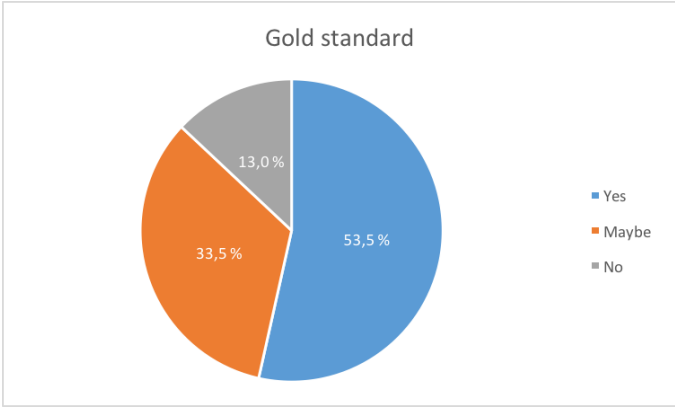

Figure 3.8: Pie chart presenting results from the collected data

3.4.1 Analysis of gold standard

The gold standard was created with the help of 11 participants: 10 of them having a doctorate degree and 4 having a profession related to the Marine Science. Based on the collected data (table 3.7 and figure 3.8) and the average relevance (equation 3.12): 107 articles are relevant, 67 are "maybe" relevant and 26 are not relevant. This results in 87% usable documents.

For evaluating the agreement, the Fleiss' Kappa has been calculated. It presents a inter-annotator agreement of 0.3130, meaning the form has many disagreements. Table 3.8 shows the number of votes for each level of relevance. From the 200 pages, 19 has been voted as relevant by all participants, and only 2 has been marked as totally irrelevant. For the opposite side, over half of the articles has never been voted as non relevant. This indicates that the most of the agreement is on the relevant pages.

Table 3.9 presents the top 30 relevant documents. The data indicates that there was a high agreement for selecting the documents as relevant with a agreement ratio between 1-0.8. The agreement of the non relevant documents (table 3.10) has the biggest ratio between 1-0.4, showing that only few articles are rated as absolutely non relevant.

Table 3.8: Relevance distribution for each number of votes per label

Votes	Relevant	Maybe relevant	Non relevant
11	19	0	2
10	29	0	6
9	19	1	7
8	12	1	4
7	19	14	7
6	16	12	5
5	13	25	11
4	16	27	6
3	10	26	7
2	12	29	11
1	12	43	33
0	23	22	101

Tables 3.11 and 3.12 present documents marked as "maybe" relevant and the list of the most disagreed documents. These indicate that the biggest group of the gold standard are documents having a low agreement ratio, which explains why the Kappa agreement score is low. The entire gold standard is presented in table A.1.

3.5 Evaluation of Topic Modelling

3.5.1 Implementing Trec Eval

After the gold standard was generated, the evaluation could be performed. The tool selected for this task was Trec Eval[6], the official tool used at the Text Retrieval Conferences. For running Trec Eval, a particular input format was required for both the gold standard (Example 3.1) and the retrieved documents (Example 3.2). The output produces multiple types of evaluation scores, including recall, precision, mean average precision (MAP) and normalized discounted cumulative gain (NDCG).

[6]Trec Eval `http://www-nlpir.nist.gov/projects/t01v/trecvid.tools/trec_eval_video/A.README`

Table 3.9: Highest agreement for relevant articles

Title	Agreement per article	Yes	Maybe	No
Plankton	1.0000	11	0	0
Phytoplankton	1.0000	11	0	0
Ocean acidification	1.0000	11	0	0
Nanophytoplankton	1.0000	11	0	0
Microbial loop	1.0000	11	0	0
Benthic zone	1.0000	11	0	0
Anoxic event	1.0000	11	0	0
Antarctic krill	1.0000	11	0	0
Redfield ratio	1.0000	11	0	0
Marine biology	1.0000	11	0	0
Filter feeder	1.0000	11	0	0
Ocean	1.0000	11	0	0
Diatom	1.0000	11	0	0
Colored dissolved organic matter	1.0000	11	0	0
Lysocline	1.0000	11	0	0
Mixed layer	1.0000	11	0	0
Foraminifera	1.0000	11	0	0
Downwelling	1.0000	11	0	0
Coral	1.0000	11	0	0
Detritus	0.8182	10	0	1
Iron fertilization	0.8182	10	1	0
Marine snow	0.8182	10	1	0
Ecosystem of the North Pacific Subtropical Gyre	0.8182	10	1	0
Eutrophication	0.8182	10	1	0
Emiliania huxleyi	0.8182	10	1	0
Dissolved organic carbon	0.8182	10	1	0
Coral reef	0.8182	10	1	0
Algal bloom	0.8182	10	1	0
Coccolithophore	0.8182	10	1	0
Seawater	0.8182	10	1	0

Table 3.10: Highest agreement for non relevant documents

Title	Agreement per article	Yes	Maybe	No
Thaumatin	1.0000	0	0	11
Samir Shihabi	1.0000	0	0	11
Mosquito control	0.8182	0	1	10
Cover crop	0.8182	0	1	10
Biological soil crust	0.8182	0	1	10
Organic lawn management	0.8182	0	1	10
Ralstonia solanacearum	0.8182	0	1	10
Olsenbanden Jr. og Sølvgruvens hemmelighet	0.8182	0	1	10
Soil biodiversity	0.6727	0	2	9
Soil food web	0.6727	0	2	9
Belgica antarctica	0.6727	0	2	9
Rock dove	0.6727	0	2	9
Drizzle	0.6727	0	2	9
Fungiculture	0.6545	1	1	9
Root	0.6545	1	1	9
Insect winter ecology	0.5636	0	3	8
Italian crested newt	0.5636	0	3	8
Plinthite	0.5636	0	3	8
Eichhornia crassipes	0.5636	0	3	8
Soil conservation	0.4909	0	4	7
Substrate (aquarium)	0.4364	1	3	7
Soil respiration	0.4364	1	3	7
Gleysol	0.4364	1	3	7
Desert	0.4364	1	3	7
Soil crust	0.4182	2	2	7
Pedosphere	0.4182	2	2	7
Climate change in Washington	0.4545	0	5	6
Marsh gas	0.4545	0	5	6
Soil erosion	0.3818	4	1	6
Peat	0.3818	1	4	6

Table 3.11: Highest agreement for "maybe" relevance

Title	Agreement per article	Yes	Maybe	No
Permafrost	0.6545	1	9	1
Atmospheric methane	0.5273	2	8	1
Plant litter	0.4909	0	7	4
Climate change feedback	0.4909	4	7	0
Isotope analysis	0.4909	4	7	0
Natural environment	0.4909	4	7	0
Permian–Triassic extinction event	0.4909	4	7	0
Environmental impact of pesticides	0.4909	4	7	0
Human impact on the environment	0.4909	4	7	0
Late Devonian extinction	0.4909	4	7	0
Lake ecosystem	0.4364	1	7	3
Salt marsh dieback	0.4364	3	7	1
Plant nutrition	0.4364	3	7	1
Erosion	0.4364	3	7	1
Runaway climate change	0.4364	3	7	1
Phosphorite	0.4182	2	7	2
Azolla	0.4545	0	6	5
Aphanizomenon	0.4545	0	6	5
Long-term effects of global warming	0.4545	5	6	0
Algaculture	0.4545	5	6	0
Integrated multi-trophic aquaculture	0.4545	5	6	0
Physical impacts of climate change	0.4545	5	6	0
Paleocene–Eocene Thermal Maximum	0.3818	4	6	1
Carbon-to-nitrogen ratio	0.3818	4	6	1
Permineralization	0.3455	2	6	3
Marine pharmacognosy	0.3455	3	6	2
Eocene	0.3455	3	6	2
Natural hazard	0.3455	3	6	2
Climate change in Washington	0.4545	0	5	6
Marsh gas	0.4545	0	5	6

Table 3.12: Highest disagreement among participants

Title	Agreement per article	Yes	Maybe	No
Ikaite	0.2727	4	3	4
Mudrock	0.2727	4	4	3
Aquarium	0.2909	3	3	5
Reef aquarium	0.2909	3	3	5
Blubber	0.2909	5	3	3
Black carbon	0.2909	5	3	3
Environmental impact of mining	0.2909	3	5	3
C4 carbon fixation	0.3091	4	2	5
Chlamydogobius	0.3091	4	2	5
Altitudinal zonation	0.3091	2	4	5
Arundo donax	0.3091	2	4	5
Carbon credit	0.3091	2	4	5
Marine aquarium	0.3091	2	5	4
Bitter Springs type preservation	0.3091	2	5	4
Organisms involved in water purification	0.3091	2	5	4
Raceway (aquaculture)	0.3091	2	5	4
Alcanivorax	0.3091	5	4	2
Planetary boundaries	0.3091	5	4	2
Renewable resource	0.3091	4	5	2
Permafrost carbon cycle	0.3091	4	5	2
Submarine eruption	0.3455	6	2	3
Permineralization	0.3455	2	6	3
Human impact on the nitrogen cycle	0.3455	6	3	2
Marine pharmacognosy	0.3455	3	6	2
Eocene	0.3455	3	6	2
Natural hazard	0.3455	3	6	2
Freshwater environmental quality parameters	0.3636	1	5	5
Berlin Method	0.3636	1	5	5
Microplastics	0.3636	5	5	1
Aquaculture of salmonids	0.3636	5	5	1

Code 3.1: Example of the gold standard format used in Trec Eval (zero indicates empty column)

```
Seed article ID, 0, Similar article ID, Relevance
129618, 0,  4209093,    0
129618, 0,  46374,      2
129618, 0,  1064680,    0
129618, 0,  9028799,    0
129618, 0,  563456,     1
129618, 0,  361028,     2
```

Code 3.2: Example of the gold standard format used in Trec Eval (zero indicates empty column)

```
Seed article ID, 0, Similar article ID, 0, Similarity rate, 0
129618, 0,  4209093,    0,  0.838495731354, 0
129618, 0,  46374,      0,  0.835850656033, 0
129618, 0,  1064680,    0,  0.834475517273, 0
129618, 0,  9028799,    0,  0.834436655045, 0
129618, 0,  563456,     0,  0.833110511303, 0
129618, 0,  361028,     0,  0.831434428692, 0
```

The methodology of the evaluation was to calculate MAP and NDCG for every topic model configuration, then compare the results. MAP and NDCG allows for evaluating a query containing multiple elements, which corresponds to the usage of multiple seek documents for each topic model. The difference between them is the evaluation of the relevance; MAP uses a binary relevance only, while NDCG supports a graded scale [Järvelin and Kekäläinen (2002)]. Difference between these two rankings will indicate the impact of the graded relevance scale.

3.5.2 Evaluation of Trec Eval results

The evaluation measures used in Trec Eval are MAP and NDCG. Three different comparisons will be presented: the number of topics, comparison between BOW and TFIDF corpora, and the model having highest rank.

Table 3.13: Best number of topics for each model variant

Model type	Corpus	Best nr of topics MAP	Best nr of topic NDCG
LDA	BOW	100	100
LDA	TFIDF	500	500
LSI	BOW	500	500
LSI	TFIDF	500	500
RP	BOW	200	200
RP	TFIDF	200	200

Number of topics

Figures 3.9 and 3.10 present the MAP and NDCG rankings for each selected number of topics. Table 3.13 summarises the best number of topics for each variant of the models. The only difference between these two evaluations are the scales, where the NDCG has higher scores, but the order is almost the same.

In most cases 500 topics gives best result, with exception of LDA model using BOW and RP models. Most of the models here have a parabolic behaviour, where they get a maximum score at a certain level and then start falling down. The TFIDF version of LDA is the only one having a sine curve, where there are local maximum values every second topic level. The LDA models start with the highest score at the lowest dimentionality, but is the first to have an decreasing value, while the RP models start at the bottom and reaches one of the highest scores towards the end. RP has also the biggest fall of all models, between the 200 and 500 topics. It could also indicate that the BOW version of RP could beat the LSI between 200 and 500 topics. The LSI models seem to be reach the maximum at 500, meaning that a higher number of topics might not improve the score.

BOW vs TFIDF

Figures 3.11 and 3.12 present the comparison between BOW and TFIDF corpora, ranked using MAP and NDCG. Table 3.14 presents the best corpus for each variant of the models. The results indicate that HDP and LDA have best results by using a bag-of-words corpus, LSI with a TFIDF corpus, RP by using TFIDF for 10 and 20 topics and BOW for 50 topics and higher.

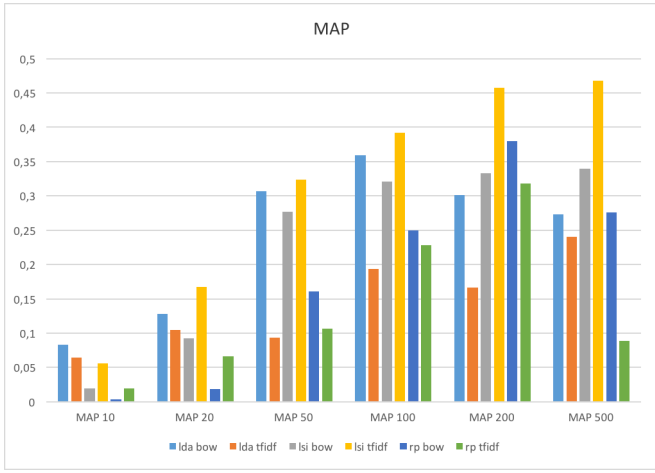

Figure 3.9: Bar chart presenting the MAP rank for all selected numbers of topics

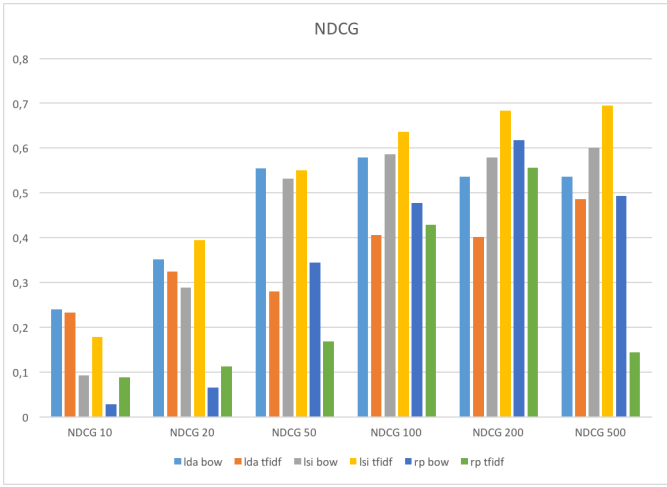

Figure 3.10: Bar chart presenting the NDCG rank for all selected numbers of topics

Table 3.14: Best corpus for each model variant

Model type	Topics	Best corpus MAP	Best corpus NDCG
HDP	-	BOW	BOW
LDA	10	BOW	BOW
LDA	20	BOW	BOW
LDA	50	BOW	BOW
LDA	100	BOW	BOW
LDA	200	BOW	BOW
LDA	500	BOW	BOW
LSI	10	TFIDF	TFIDF
LSI	20	TFIDF	TFIDF
LSI	50	TFIDF	TFIDF
LSI	100	TFIDF	TFIDF
LSI	200	TFIDF	TFIDF
LSI	500	TFIDF	TFIDF
RP	10	TFIDF	TFIDF
RP	20	TFIDF	TFIDF
RP	50	BOW	BOW
RP	100	BOW	BOW
RP	200	BOW	BOW
RP	500	BOW	BOW

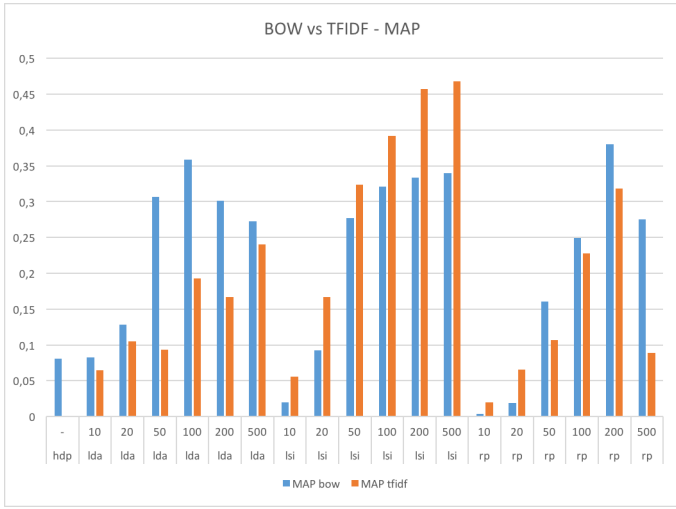

Figure 3.11: Bar chart presenting the MAP rank comparison between BOW and TFIDF corpuses

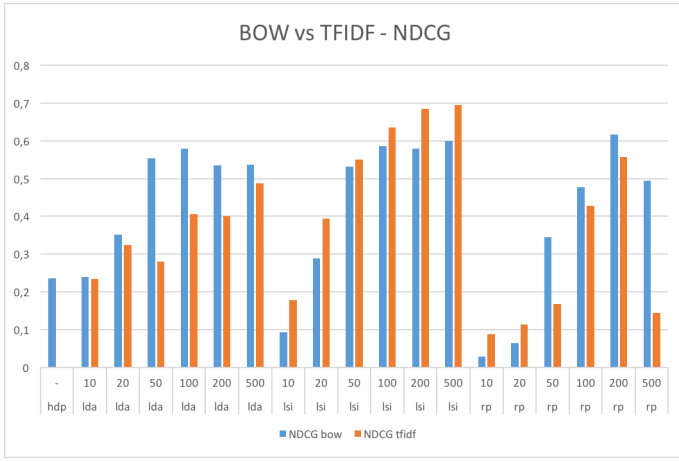

Figure 3.12: Bar chart presenting the NDCG rank comparison between BOW and TFIDF corpuses

Table 3.15: Top 5 topic models by MAP

Model type	Corpus type	Topics	MAP
LSI	TFIDF	500	0.468
LSI	TFIDF	200	0.4573
LSI	TFIDF	100	0.3923
RP	BOW	200	0.3801
LDA	BOW	100	0.3588

Best model

Figures 3.13 and 3.14 present the best model in terms of MAP and NCDG. Tables 3.15 and 3.16 present the top 5 models for either evaluation measure. Both evaluations arrive at the same results; the differences occur after the 4th position in the ranking. The best model is LSI, using 500 topics and TFIDF corpus.

Table 3.16: Top 5 topic models by NDCG

Model type	Corpus type	Topics	NDCG
LSI	TFIDF	500	0.6956
LSI	TFIDF	200	0.6842
LSI	TFIDF	100	0.6361
RP	BOW	200	0.617
LSI	BOW	500	0.6004

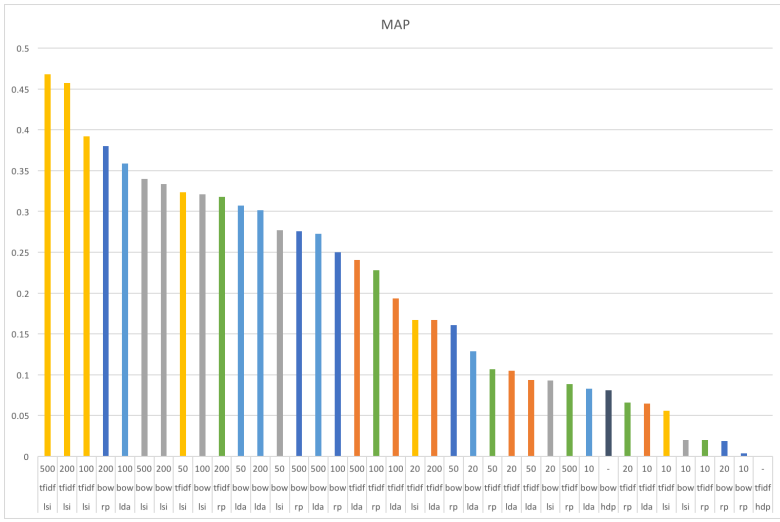

Figure 3.13: Bar chart presenting the MAP rank for all variants of topic models

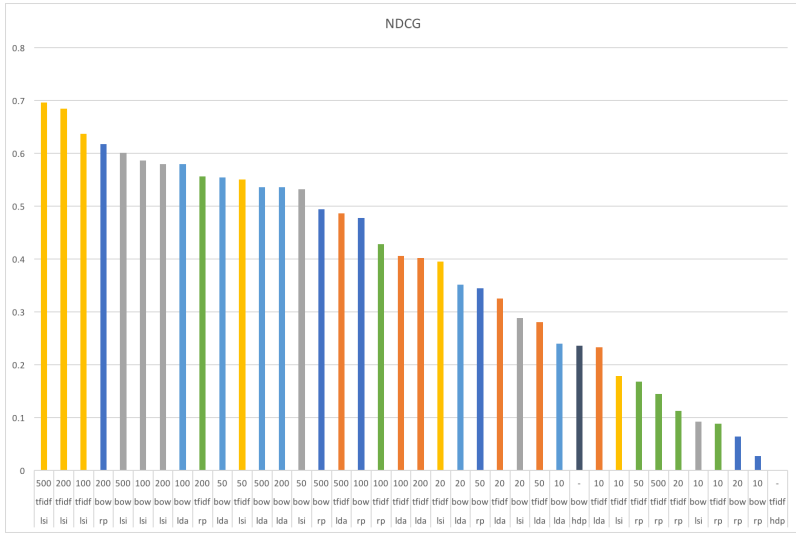

Figure 3.14: Bar chart presenting the NDCG rank for all variants of topic models

3.6 Document Filtering

Once the best topic model was established, it was used to select the most relevant pages from Wikipedia. This was accomplished by retrieving the similar articles to each of the seed pages and merging them together. Since the seed pages were marked with multiple levels of relevance, the available options were to select only the most relevant articles (marked as "Yes") as seed set or the articles with an intermediate relevance too (marked as "Yes" or "Maybe").

An additional filter method used is the thresholding of the similarity rate. The filter's task is to limit the number of retrieved articles. The articles having a similarity rate lower then the thresholds value will be removed from the document collection. The result of the thresholding is removing some of the irrelevant articles. The selected articles are then used as the data for the named entity recognition and the search interface.

Documents are retrieved using a LSI model (500 topics, TFIDF corpus), which was selected as the best model. The chosen seed articles are the ones marked as relevant. Each seed article retrieved 200 similar documents, which have been first filtered by a threshold value and then merged together. Table 3.17 presents the number of retrieved documents after a certain threshold level.

The data shows that only 2 retrieved articles have the rating lower then 0.6 and 21.9% have the rating lower then 0.7. The document collection with 0.7 threshold has been selected, because it has the closest rejection rate to the gold standard, which evaluates 13% as non relevant. As result, the non related documents being at the bottom were removed.

3.7 Discussion

3.7.1 Gold standard

One of the concerns of using the form for creating the gold standard was the amount of effort needed for completing it. The general idea was to quickly judge each article based on the summary, but judging 200 articles can be a tedious process and cause inaccurate evaluations towards the end. Another aspect affecting the quality of evaluation is the background of the participants. People having experience with marine

Table 3.17: Number of retrieved documents per selected threshold

Threshold	Number of articles	Rejection percentage
0.0	6059	0.0
0.1	6059	0.0
0.2	6059	0.0
0.3	6059	0.0
0.4	6059	0.0
0.5	6059	0.0
0.6	6057	0.0
0.7	4727	21.9
0.8	2108	65.2
0.9	83	98.6
1.0	0	100.0

science will certainly fill out the form more accurately then people having a different background (e.g. computer science).

The received data showed that a large part of the participants were claiming to have their profession related to marine science. Some of the participants stated that the evaluation was an educational experience, but also some mentioned that it was getting tedious towards the end. This can be seen in table 3.10, where one participant selected "Olsenbanden" as a "maybe"-relevant article, when it is absolutely unrelated. Because the "Olsenbanden" appears as one of the last articles in the form, it shows that some of the participants were not evaluating carefully.

3.7.2 Topic Modelling

Creating Topic Models

One of the most impressive aspects of topic modelling is its versatility. All input required for training of the system was the initial list of 22 articles related to marine science - resulting in over 200 times larger list of relevant documents. Exactly the same approach can be used for retrieving sets of documents related to other subjects, but it might also require finding new optimal configurations.

Gensim is a well documented tool, capable of processing large numbers of documents. The experienced limitations were the limited performance of HDP models, and computationally expensive indexing of topic model representations. The Gensim's

website mentions that HDP is an experimental method, explaining the low results.

While most of the parts of Gensim are operating in fixed memory, the indexing of similarities requires fitting all of the data into memory. There is a method allowing for doing this in fixed memory, but it turned out it was not working with the current configuration. As result, the memory usage was at 40 GB of RAM when computing a model having 500 topics, making it difficult to test with higher number of topics.

Evaluating Topic Models

The major difference of the evaluation measures used, is the scale of the scores, where the NDCG scores are almost twice as high the MAP. Because NDCG calculates relevance by squaring the relevance level (i.e. a document having relevance level 2 results in a score of 4), it suggests that the majority of the documents have the highest relevance, which is proven by the data from the gold standard. Since the MAP and NDCG present the models in almost the same order, it indicates that the ternary relevance level performs almost the same as the binary (MAP uses "Yes" and "Maybe" levels as one relevance level).

3.7.3 Document Filtering

The document filtering uses the same pooling method as used for generating the form. Only differences are a larger set of seed documents and a single topic model. The idea of using thresholding for filtering the number of retrieved pages originated from the way the form for gold standard was created. The articles were sorted by the similarity score, resulting in the beginning of the form being more relevant then the end. By setting a threshold, these non relevant documents having lowest score will be removed and make the retrieved document collection more relevant.

Chapter 4

Search Interface

4.1 Introduction

The purpose of a *search interface* is to present the filtered documents by using search term queries. For allowing additional filters, *faceted search* has been implemented by extracting named entities from the articles. These entities include geographical locations, marine species and variable changes. The interface also contains visualisation methods for the geographical locations.

The structure of this chapter presents first the theoretical background about information extraction, including *named entity recognition* and its applications. The second part describes the implementation of NER and faceted search. The last part includes discussion over the used implementation.

4.2 Background

4.2.1 Information Extraction

A disadvantage of methods using BOW approach for document representation is the partial loss of semantics, because the linguistic structure of sentences and phrases is lost due to the representation using a weighted term frequency. The goal of *Information Extraction* (IE) is to find entities and relations in unstructured text [Jiang (2012)]. The

relations can facilitate among other things, semantic search.

Entities

Entities are the smallest parts of data in an Information Extraction system. They represent "objects" which are described by the unstructured information. Examples of entities extracted by IE can include people, places, and organizations. In a sentence "Steve Jobs was the founder of the company Apple, located in California." - the entities extracted by IE are "Steve Jobs", "Apple" and "California".

The IE subtask responsible of extraction of entities is called Named Entity Recognition. In addition to recognizing entities, NER will also try to annotate the category of the entity. Referring to the example above, "Steve Jobs" would be annotated as a person, "Apple" as a company, and "California" as a place.

Relations

Relations in IE refer to relations between entities. The usual form of representing relations is by using a so-called triplet, having a "subject-predicate-object" notation. Using the same example as for entities, relations can be created based on terms "founder" and "located". The first term can create relation founder("Steve Jobs", "Apple"), and second term creates location("Apple", "California").

Events

Events are defined as a set of multiple relations, which are used for describing a situation using structured information. The events do not need to be binary but can be ternary and higher. As mentioned by Piskorski and Yangarber (2013), events should be ideally identifying *"who did what to whom, when, where, through what methods (instruments), and why"*. An example of an event is an article about a natural disaster (Table 4.1 from Ronen Feldman (2007)), where the extracted information contains the type of disaster, date and time it happened, location, damaged location and number of injuries.

Table 4.1: Table presenting event extraction from an article about tornado in Texas, USA

Event:	tornado
Date:	1997-4-3
Time:	19:15
Location:	Farmers Branch : "northwest of Dallas" : TX : USA
Damage:	mobile homes, Texaco station
Estimated Losses:	USD 350 000
Injuries	none

Approaches to IE

According to Appelt (1999), there are two approaches to Information Extraction. A knowledge engineering approach will require building an ontology by a "knowledge engineer" - a person having knowledge of building IE systems. The engineer will have access to a moderate amount of documents (an amount possible to examine by a single person in reasonable time). This person either has knowledge of the information to be extracted or collaborates with a domain expert. The algorithm will be created as set of rules, which are representing the possible relationships of the entities in text, i.e. will be a base for extracting the desired knowledge from unstructured text. This approach not only requires a skillful engineer, but also a tremendous amount of work, because all of the rules have to be created by hand.

Th second option is to use a data-driven approach, where a supervised machine learning learns to extract the information by itself, based on hand-labeled training data. The training process requires annotating manually the parts of text to be extracted by the system. Another training method is having a user validate the results of the system, to improve its precision. User validation can include indicating the correctness of the system's hypotheses - if not, the system will modify itself to create a better fit to the data.

DBPedia Information Extraction Framework

An example of a service using IE is DBPedia Information Extraction Framework (DIEF), presented by Bizer et al. (2009). It is a semantic representation of Wikipedia's content. Two main components of the system are the extraction of content and serialisation of the extracted data.

The extraction of data is based on the infoboxes of the articles. These boxes contain

a semi-structured table with parameters describing the article's subject, where each row contains the type of the parameter and its value (example code 4.1).

Code 4.1: Example of infobox about Norway in Wiki Markup notation

```
{{Infobox country
| common_name = Norway
| capital = [[Oslo]]
| area_km2 = 385178
| population = 5165802
}}
```

After extraction, the serialization process is initialized. It uses a Resource Description Framework (RDF) to create an ontology containing semantics of the data. The data are saved using triples, using a "subject-predicate-object" structure. The extraction process does also detect the datatypes of the input. In example code 4.2, the common name is detected as a string, the capital is detected as an object, because it contains a link pointing to another article. Finally the area and population are detected as integers.

Code 4.2: RDF representation of the infobox in Turtle notation

```
dbp:Norway  dbp:common_name "Norway" .
dbp:Norway  dbp:capital     dbp:Oslo .
dbp:Norway  dbp:area_km2    "385178"^xsd:int .
dbp:Norway  dbp:population  "5165802"^xsd:int .
```

A challenge of the extraction task is the lack of defined terminology in the infoboxes. For example the infoboxes of multiple people can present the persons birthday using different parameters like: "Born", "birthdate", or "birth_date". The way it is solved is by manually mapping 350 templates, which limits the amount of infoboxes possible to extract.

The reason why the consistency of the datatypes is so important lies in the possibility of accessing the data by queries. Because of the semantic structure of RDF, it is possible to treat the document collection as a database and create queries like "find all countries in Europe with population bigger then 50 million inhabitants".

4.2.2 Faceted search

Faceted search [Hahn et al. (2010)] is a way of improving the search experience for users. Usually a search is performed by writing a query containing a couple of words, and the system (e.g. Google) will return a list of documents related to the query. The limitation of this approach is the lack of specifying details of parts of the query. For example, while searching for rivers in Germany of length longer then 500 kilometers, a normal search will depend on documents in the corpus containing literal information from the query. Faceted search provides additional categories, based on faceted classification (Section 4.2.2). By adding categories to search, the results can be limited to documents which only correspond to selected filters.

E-commerce is one of the markets where faceted search is often used. By using faceted classification, it is possible for users to browse through products without a need for defining a precise search query. For example it makes it possible to search for a laptop without the need to specify a specific model, but only by setting up filters considering categories including screen size, battery capacity, and a price range.

Most of the faceted search interfaces use predefined facets, manually added into the databases e.g. parameters of a product in an e-commerce solution. Implementations extracting facets from free text are much rarer, mostly because of the difficulty of extracting entities and relations. Often can the lack of reliability be a major concern.

Faceted classification

Faceted classification allows categorising content in a set of groups, where the groups are describing all content of the document collection. Each facet is mutually exclusive and jointly exhaustive. This means that there cannot be any overlapping facets and at least one facet must occur in a document. For example, selection of brand X in facets will cause removing all other brands from the search results. Displaying empty brands as facets is impractical, because it will not give any useful information for the user.

DBPedia Faceted Browser

DBPedia created a project allowing for searching Wikipedia content using the faceted search paradigm[1]. The system was based on a semantic representation of Wikipedia, allowing for browsing data using queries (More detail in section 4.2.1).

The graphical user interface in figure 4.1 demonstrates how the faceted search works. The interface consists of following elements:

- Free Text Search
- Item Type Selection
- Facet Selection
- Result Navigation
- Selected Facets
- Search Results

The implementation allowed for using a classic search approach, where the content of the query in "Free Text Search" field was matched with content of the document. "Item Type Selection" made it possible to choose a specific category of interest. Each of the categories had defined parameters, which were possible to specify using the "Facet Selection" field. As in the example in figure 4.1, for the selected category of person, "Facet Selection" shows all available parameters of the category, which then allows for narrowing the search results. Selected filters are shown under "Selected Facets" and the results themself are shown under "Search Results". Since the results could be presented on multiple pages, "Result Navigation" allowed for browsing between them.

Faceted search and Information Extraction

Faceted Search relies on well-formulated facets in order to provide a good expressiveness and precise search results. While a manual categorization of each product for an

[1]Description of DBPedia Faceted Browser project (discontinued in 2012) `http://dbpedia.org/projects/faceted-wikipedia-search`

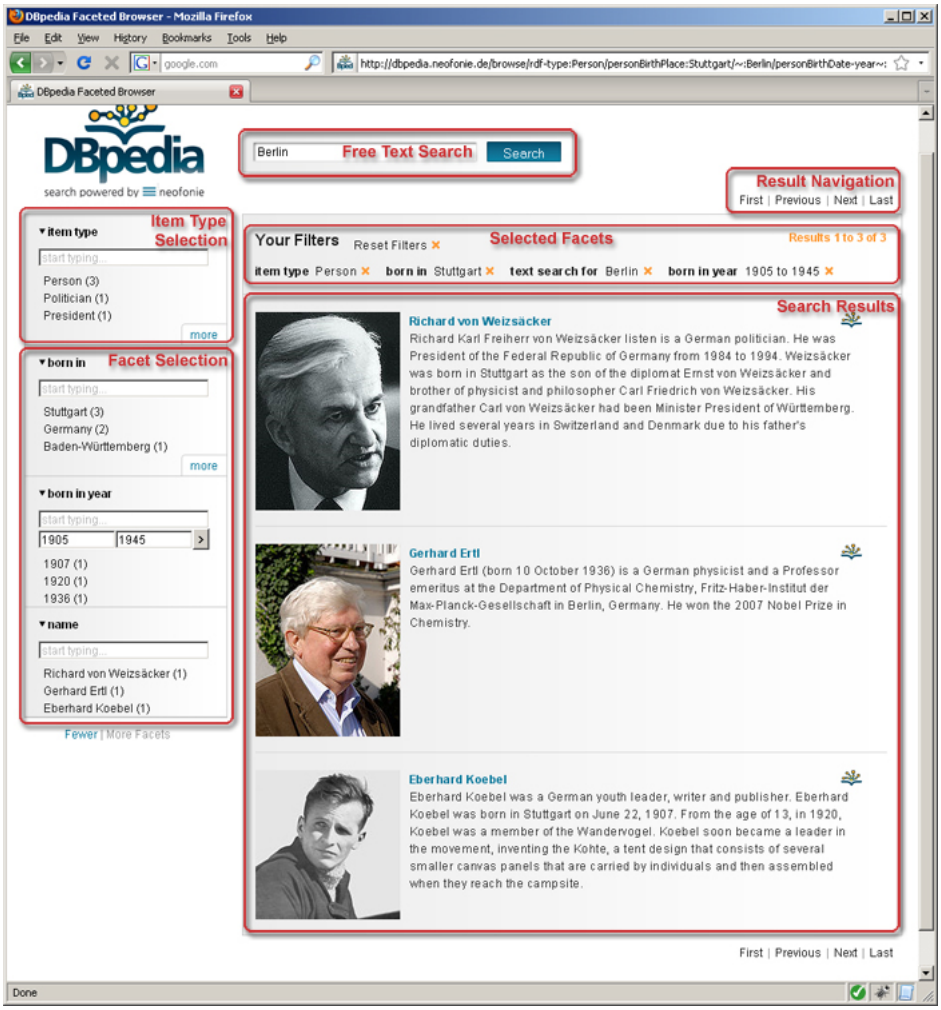

Figure 4.1: Graphical user interface of DBPedia Faceted Browser

e-commerce application might be a sufficient solution, manually categorising each entity in the document collection would take too long. As shown in the DBPedia Faceted Search, the entities and their relations extracted by a IE system, can be used as facets.

Another example of using the IE for the Faceted Search was introduced by Atanassova and Bertin (2014). This system focuses on the Information Retrieval for scientific publications, based on extracting facets from the document collections. The extraction is based on the *Contextual Exploration* (CE) - a tool used for automatic annotation of parts of text, including titles, words and sentences Desclés (2006). CE is used for annotating each sentence with a category. The categories include result, summarize, conclusion and opinion. The system allowed for searching the sentences filtered by a selected category. Additionally the system also presents the previous and the next sentence of the same paragraph and points to the document and position of retrieved sentence.

4.3 Implementation of named entitity recognition

The next goal of the thesis was to use *named entity recognition* to extract data of selected categories. The chosen categories include geographical locations and marine species. The systems used for this task were MRNER[2] and WoRMSNER[3], created by Erwin Marsi. For extraction of the entities, these systems require input files that contain the string of the entity and other metadata including the IDs, language of the entity, and a category. An example of the Marine regions data is shown in table 4.2 and an example for WoRMS data in table 4.3. The NER works by tokenizing the input documents and finding matches between tokens and the entities. The matches are then extracted with the entity's metadata and additional location of tokens in the text.

The source of the data are Marine Regions[4] (MR) for the geographical locations [Claus et al. (2014)] and World Register of Marine Species[5] (WoRMS) [Costello et al. (2013)]. Input consists of one text document per file. The documents used for NER come from the list of relevant articles generated in the previous step.
Following list presents 5 sentences where geographical locations were found:

[2]MRNER `https://github.com/OC-NTNU/mrner`
[3]WoRMSNER `https://github.com/OC-NTNU/wormsner`
[4]Marine Regions `http://www.marineregions.org`
[5]World Register of Marine Species `http://www.marinespecies.org`

"Paradoxically, oceanic areas adjacent to unproductive, arid land thus typically have abundant phytoplankton (e.g., the eastern **Atlantic Ocean**, where trade winds bring dust from the **Sahara** Desert in north Africa)." - Plankton

"In oligotrophic oceanic regions such as the **Sargasso Sea** or the South Pacific Gyre, phytoplankton is dominated by the small sized cells, called picoplankton, mostly composed of cyanobacteria ("Prochlorococcus", "Synechococcus") and picoeucaryotes such as "Micromonas"." - Phytoplankton

"This particular gyre covers most of the **Pacific Ocean** and comprises four prevailing ocean currents: the North Pacific Current to the north, the **California Current** to the east, the North Equatorial Current to the south, and the **Kuroshio Current** to the west." - Ecosystem of the North Pacific Subtropical Gyre

"Some further south is the Sula Reef, located on the Sula Ridge, west of **Trondheim** on the mid-Norwegian Shelf" - Deep-water coral

"The first ever live video of a large deep-water coral reef was obtained in July, 1982, when Statoil surveyed a tall and wide reef perched at water depth near Fugløy Island, north of the Polar Circle, off northern **Norway**." - Deep-water coral

Example of 5 sentences containing the extracted marine species entities:

"They are an informal grouping within the infraorder **Cetacea**, usually excluding dolphins and porpoises." - Whale

"Sawfishes, also known as carpenter sharks, are an order **(Pristiformes)** of rays characterized by a long, narrow, flattened rostrum, or nose extension, lined with sharp transverse teeth, arranged so as to resemble a saw." - Sawfish

"**Cyanobacteria** played an important role in the evolution of ocean processes, enabling the development of stromatolites and oxygen in the atmosphere." - Seawater

"However it may benefit some species, for example increasing the growth rate of the sea star, **Pisaster ochraceus**, while shelled plankton species may flourish in altered oceans." - Ocean acidification

"The Galapagos shark (Carcharhinus galapagensis) is a species of requiem shark, in the family **Carcharhinidae**, found worldwide." - Galapagos shark

An additional information extraction system used in this thesis included extraction of *variable changes* [Marsi and Oztürk (2015)]. This system finds text fragments describing a change by matching tree patterns onto the syntax trees of the source text. The variables are labeled an increasing, decreasing or changing. The output of this system included the string of the extracted variable, the variables label and the indexes of start and end positions of the variable in the text (Example 4.3).

Example of 5 extracted variable changes. The variable direction is indicated by colour: red for increase, blue for decrease and green for change:

"**Deep-water corals** grow more slowly than tropical corals because there are no zooxanthellae to feed them." - Deep-water coral

"**Compounds such as salts and ammonia dissolved in water** lower its freezing point, so that water might exist in large quantities in extraterrestrial environments as brine or convecting ice." - Ocean

"Ocean acidification is the ongoing decrease in **the pH of the Earth's oceans, caused by the uptake of carbon dioxide** () from the atmosphere." - Ocean acidification

"Changes in the speed of sound are primarily caused by changes **in the temperature of the ocean** , hence the measurement of the travel times is equivalent to a measurement of temperature." - Ocean acoustic tomography

"Anticipated effects include warming global temperature, rising **sea levels**, changing precipitation, and expansion of deserts in the subtropics." - Global warming

Table 4.2: Example of input data for MRNER

MRGID	GeoName	Language	Placetype
14	België	Dutch	Nation
14	Belgique	French	Nation
14	Belgium	English	Nation
14	Belgica	Spanish	Nation
14	Belgien	German	Nation
14	Belgio	Italian	Nation

Table 4.3: Example of input data for WoRMSNER

AphiaID	RankName	ScientificName
410608	Species	Kerguelenella macra
410609	Species	Kergueleniola macra
410610	Genus	Kerguelenicola
410611	Genus	Maghrebidiella
410612	Species	Maghrebidiella maroccana
410613	Genus	Marigidiella

Code 4.3: Example of JSON data from the extracted variable changes

```
{
"nodeNumber": 3,
"subStr": "deep-water coral",
"extractName": "SUBJ_grow",
"treeNumber": 92,
"filename": "16730504 - Deep-water coral#scnlp_v3.6.0.parse",
"charOffsetEnd": 10372,
"subTree": "(NP (JJ deep-water) (NNS coral))",
"charOffsetBegin": 10355,
"label": "increase",
"key": "16730504 - Deep-water coral#scnlp_v3.6.0.parse:92:3:SUBJ_grow"
},
```

4.4 Results of named entity recognition

4.4.1 Extraction of geolocation entities

The result of the MRNER is a total of 27435 extracted entities, found in 2341 documents. This gives an average of 11.7 entities per document. Table 4.4 shows the 10 most frequent geolocations in the document collection.

Table 4.4: Most frequent geographical locations in the document collection

Location	Number of documents
United States	685
Europe	342
World	319
Australia	285
North America	269
China	268
California	229
Canada	228
United Kingdom	227
Pacific	221
Japan	218
Africa	202
India	197
Asia	179
New Zealand	144
Arctic	142
River	139
Germany	137
South America	125
Pulau Air	119
Russia	117
England	113
Mexico	113
Florida	110
Texas	110
Antarctica	108
France	107
Western	107
South Africa	105
Antarctic	102

4.4.2 Extraction of marine species entities

The marine species were extracted by using WoRMSNER based on the World Register of Marine Species. The total of 12769 species entities were extracted from 1878 pages. The average is 6.8 species entities per document. Table 4.6 shows the most frequent Marine species in the document collection.

4.4.3 Extraction of variable changes

The extraction of variable changes was performed by using a tool created by Marsi and Oztürk (2015). The tool found a total of 66695 variable changes in 4210 documents, including:

- 18141 change variables
- 27564 increase variables
- 20990 decrease variables

This gives an average of 15.8 variables per document, including:

- 4.3 change variables per document
- 6.5 increase variables per document
- 5.0 decrease variables per document

Table 4.6 presents the most frequent variables in the corpus.

Table 4.5: Most frequent Marine species in the document collection

Marine species	Number of documents
Russia	163
Mars	125
Bacteria	100
fragile	77
Virginia	66
Escherichia coli	59
Venus	58
Argentina	57
Victoria	55
Pseudomonas	51
Ammonia	49
Cancer	48
Colombia	43
Fungi	40
Bacillus	36
Archaea	35
Cyanobacteria	35
Clostridium	34
Saccharomyces cerevisiae	34
Costa	30
Salmonella	30
Scotia	29
Johnson	28
Staphylococcus	28
Streptococcus	28
Le	26
Mariana	26
Caenorhabditis elegans	24
Cryptosporidium	20
Daphnia	20

Table 4.6: Most frequent variables in the document collection

Variable	Number of documents
NP	653
climate	176
temperature	172
water	114
plant	95
pressure	63
species	62
population	55
fish	52
environment	43
sea level	38
ph	35
water level	35
habitat	34
heat	34
oxygen	33
salinity	32
color	31
cell	30
bacterium	29
agent	28
number	27
size	25
biodiversity	24
water temperature	24
condition	23
nutrient	23
production	22
organism	21
process	21

4.5 Implementation of user interface

4.5.1 Graphical User Interface

For visualising results, a graphical user interface has been implemented. The GUI presented in figure 4.2 is a website containing following elements:

1. **Search bar** for typing in queries. The query is sent after 1 second of inactivity or after pressing the enter.

2. **Map** displays all the available location filters using blue pins. When a location is selected (by either using filter list or clicking the pins) the pin's colour changes to red.

3. **Retrieved document** displays the title with a link to the Wikipedia page and a text description of the article. The text description highlights the search terms from the search bar and the selected filters.

4. **Extracted variable changes** are displayed under the text description. It contains the part of text where the variable is located and highlights the variable using a colour that corresponds to the variable's direction.

5. **Selected filters** displays the filters selected from the filter list underneath. The filters can be cancelled by clicking the "x" button.

6. **Filters** displays all the available facets for filtering the search results. The list of facets contains geographical locations, place type, species, taxonomy, variable text and variable label.

4.5.2 Search engine architecture

The topic modelling system has extracted a couple of thousand articles related to marine science. An efficient way of opening up these articles is through a search engine system, where a user can find documents related to marine science. Our search is based on Solr for the back-end and a single page application running JavaScript called SolrStrap[6].

[6] SolrStrap `http://fergiemcdowall.github.io/solrstrap/`

Figure 4.2: Graphical user interface of the Ocean Wiki search engine, the elements are explained in section 4.5.1.

1. Search bar
2. Map
3. Retrieved document
4. Extracted variable changes
5. Selected filters
6. Filters

The search engine implementation consists of processes on both server and client side. The server side is running Solr with the indexed data (section 2.3.2). The main index contains the extracted marine science articles with corresponding named entities (geolocations, species and variables). Additional indexes were created for the geographical locations and variable changes. The reason of having extra indexes is to access the additional metadata of the entities, e.q. the coordinates for geolocations and the character offsets for the extracted variables. Along with the retrieval of the documents, the server side does also generate the so-called *text snippets*, which contain parts of the document text that contain the selected search terms. The last task of the Solr is to produce the lists of facets, based on the named entities of the retrieved articles. The facets are used for filtering the results.

The SolrStrap is running at the client side in a web browser. The front-end's task is to display the graphical elements on the website and send requests to the server. Main features of the SolrStrap includes a *real-time search* - where the documents are retrieved as the query is typed in, and *infinite scrolling* - allowing a continuous loading of retrieved documents when the bottom of the page is reached. Extra elements include the lists of the selected and available facets (section 4.5.1). An optional setting allows also for displaying the text-snippets for the retrieved documents. The communication to the server is performed by sending queries containing the search terms and the selected facets. The response from the server is the lists of retrieved documents, facets and text-snippets.

The modifications made to the front-end involved adding the JavaScript version of *Google Maps*[7] and a solution to display the extracted variable changes. This allows using the locations on the map as facets (section 4.5.4) and to display parts of the text containing the variables changes (section 4.5.5).

4.5.3 Faceted search

Faceted search is used as filters, where users can select specific categories of interest. The purpose of faceted search in this context is to demonstrate how the extracted entities can be used as facets.

[7] JavaScript Google Maps `https://developers.google.com/maps/documentation/javascript/`

For using the faceted search, the entities for each article were added to the Solr index by using a multivalued field, i.e. an array of entities of a specific category. The added facets include data from the name entity recognition: geographical location (location name and place type) and marine species (species name and genus). Additionally, the variable changes are also added as facets, including the text of the variable, and its value.

Faceted search is also one of the built in features of SolrStrap, which allows for easy implementation, by setting up the names of facet fields. A small modification has been added for highlighting the faceted terms in the text snippets. This is achieved by modifying a parameter called highlight query, which extends the original search term with the facets. This modification forces Solr's highlighting tool to find and highlight snippets containing both the original query and the facet. This can be seen in the "Anoxic event" article (figure 4.2), where the query contains "ocean" and the selected facet is "Europe".

4.5.4 Map

An interesting way of visualizing the geo-location facets is to present them on a map. The map was placed right under the search bar, so it can inherit its fixed position on the screen. The metadata of recognized geo-location entities does not contain the coordinates. The Marine Regions API was used for retrieving the coordinates corresponding to the locations ID value. For retrieving the location data in the website, a dedicated Solr index was created. The content of the index included metadata and the corresponding coordinates.

The locations are placed on the map during the loading of the facets. The name of a facet (i.e the name of geo-location) is used as a search term in the location index, and the retrieving its coordinates, which can be pinpointed on the map. The locations are placed in the form of blue pins, which turn red when the location facet is selected. To make the map interactive, the pins have an additional click listener, which toggles the facet. This step required another modification of the front-end, because the built in function was bound to the existing HTML structure.

4.5.5 Extracted variable changes

The final part of the website included displaying the extracted variables for each retrieved article. The selected method of displaying the variables was to create separate text snippets for each variable. This is because SolrStrap does not support hierarchical facets, which does not allow using the server side for expanding the text description (more detail in discussion 4.6.2). A Solr index has been used, in a similar manner to the map implementation, for retrieving the start and end char offsets of the variable. From these offsets, a window could be generated, where the variable was surrounded by 10 terms on each side.

Colour was used to highlight to the variable according to its type: red for increasing, blue for decreasing and green for changing. If the snippet contained the search term too, it was rendered in bold font. Additional functionality allowed for using facets to filter variables based on their type.

4.6 Discussion

4.6.1 Extraction of named entities

The extraction of named entity recognition works by tokenizing the text of a document and comparing tokens to the entities from Marine Regions and WoRMS data. The comparison is case sensitive, causing failed matches when the entities in the text use different case then the entities in the references (i.e. "ocean acidification" will not match "Ocean acidification"). Another recorded issue in NER involves ambiguity of results. Based on the top results from WoRMS NER (table 4.6), one of the most occurring entities are Russia and Virginia. They are extracted because the WoRMS database defines them as marine species entities, while in most cases these entities are related to the geographical locations instead. This happens because the NERs does not implement entity disambiguation.

4.6.2 Search interface

The Solrstrap used in the front-end was performing well for the tasks it was designed for: connecting to a Solr index, retrieving documents by queries, generating and high-

lighting text snippets and filtration by faceted search. Implementing the Google Maps into the website by modifying the source code was a trivial task, because these systems were working asynchronously.

The problems started when the results from variable changes were implemented. The planned functionality was to retrieve articles containing variables specified by the variable label facet, but do not retrieve the article if the variable of is too far away from the search term. The source of the problem was the lack of support for hierarchical facets in SolrStrap, resulting in the lack of relation between the label and the text of a variable in the index. Initial implementation included a two step approach, where the documents are retrieved first, and then the additional variable snippets are rendered while loading them on screen. In case the retrieved article had no variables satisfying the criteria, the article was still present in the results. One possibility would be to remove the article from the result, but then the number of facets would be indicating an incorrect number of documents. Because the rendering of variable changes was performed at the client side, the performance is noticeably reduced for larger articles.

Another minor issue with the user interface was the lack of support for smaller screens/mobile devices. Specially the map which has a static height makes it difficult to browse on a smart-phone. The desirable feature would be a hiding map and sidebar menu for the facets.

Chapter 5

Conclusion

5.1 Summary of results

The result of this thesis is a knowledge discovery support system, capable of retrieving Wikipedia articles relevant to marine science, automatically extracting facets from free text and providing a visualisation of the data. This summary concludes the research goals defined in chapter 1. The search interface[1] and the source code[2] of the project are published online.

5.1.1 How to access the Wikipedia articles?

The Wikipedia articles are accessed by using Wikipedia dump files i.e. regular backup files of the entire Wikipedia content, containing the article content in a WikiMarkup format with additional metadata related to the article. The content is stored locally for allowing fast access to the data (more details in section 2.2).

5.1.2 How to extract article text?

The extraction of the article text is performed by WikiExtractor (section 2.4). This tool allows the extraction of the article's plain text from the WikiMarkup format by removing

[1] Ocean Wiki `http://folk.ntnu.no/mateuss/ocean-wiki/`
[2] Source code `https://github.com/mateusz-siniarski/master-thesis-source-code`

all markup tags. The plain text of the articles is serialised into a hierarchy of text files, including the article's metadata.

5.1.3 How to find articles relevant to marine scientists?

Topic Modelling (section 3.3) is used for finding documents relevant to marine science. The process is divided in multiple parts: creating set of seed articles, creating a set of topic models, creating the gold standard for defining relevant documents evaluating the topic models, and applying the topic models. The topic model with the highest score in the evaluation is the Latent Semantic Analysis (LSA), configured to 500 topics and a TFIDF transformed corpus.

5.1.4 How to extract terms for implementation of the faceted search?

The extraction of named entities is implemented by using two Named Entity Recognition systems and a system responsible of finding variable changes in text (section 4.3). The NERs extracted the entities from the relevant documents by using the Marine Regions and WoRMS databases. The results are sets of entities for geographical locations and marine species. Geographical entities contain data of name of the geographical location, the type of location and coordinates of the location. Marine species entities contain the name of the species and its rank. Variable changes are also extracted, where the output is the text of the variable and its direction of change. These entities and variables are used as facets in the search interface.

5.1.5 How to present the results in a user interface?

Implementation of the user interface requires creating back-end and front-end solutions for storing and accessing the data (section 4.5). Solr is used as the back-end, for indexing and storing the relevant articles, the extracted named entities and the variable changes. A modified version of SolrStrap is used for the front-end, facilitating retrieval of indexed documents by search term. The interface provides a map displaying the locations extracted from the retrieved articles as well as facets for filtering the search results. The retrieved documents are presented in a form of text snippets containing the search terms. Additional text snippets show the extracted variables.

5.2 Future work

This section contains recommendations for future work which is considered as most promising for further improvement of the system developed so far.

5.2.1 Testing more topic modelling configurations

The evaluation of topic modelling in chapter 3 suggests trying out different configurations of the models. An example is the sudden score decrease of RP models from 200 to 500 topics (section 3.5), suggesting a potential maximum value between these number of topics. A higher number of topics can also yield improvements, but for achieving this part, the high memory usage issue has to be solved (section 3.7).

5.2.2 Better matching of the extracted named entities and word sense disambiguation

A problem found in the extraction of the named entities is that some of the marine species share the same names as geographical locations (more detail in section 4.6). The result in most cases is a geographical location being extracted as a marine species. A solution to this issue could be implementing an entity disambiguation system, which uses the context of the entity and to determine its correct category.

5.2.3 Use a front-end that supports hierarchical facets

The biggest disadvantage of the SolrStrap front-end is the lack of support for hierarchical facets. The lack of this feature causes problems when attempting to remove retrieved articles if the distance between a search term and a variable is too large (Problem described in more detail in section 4.6). Hierarchical faceted search can create a relation between the text of the variable and its direction, allowing for filtering the documents before retrieving. A solution could be using a different front-end with support for hierarchical facets.

5.2.4 Testing the system with targeted users

For the final verification of the knowledge discovery support system, it should be tested with the marine scientists for verifying if the system meets their expectations and if it can optimise their research. By noting the documents they find relevant/irrelevant, the system could be retrained to better match their expectations.

Bibliography

Andreas Hotho, Andreas Nürnberger, G. P. (2005). A brief survey of text mining.

Appelt, D. E. (1999). Introduction to information extraction. *Aicommunications*, 12(3):161–172.

Atanassova, I. and Bertin, M. (2014). Semantic facets for scientific information retrieval. In *Semantic Web Evaluation Challenge*, pages 108–113. Springer.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009). Dbpedia-a crystallization point for the web of data. *Web Semantics: science, services and agents on the world wide web*, 7(3):154–165.

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.

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

Bornmann, L. and Mutz, R. (2015). Growth rates of modern science: A bibliometric analysis based on the number of publications and cited references. *Journal of the Association for Information Science and Technology*.

Claus, S., De Hauwere, N., Vanhoorne, B., Deckers, P., Souza Dias, F., Hernandez, F., and Mees, J. (2014). Marine regions: towards a global standard for georeferenced marine names and boundaries. *Marine Geodesy*, 37(2):99–125.

Costello, M. J., Bouchet, P., Boxshall, G., Fauchald, K., Gordon, D., Hoeksema, B. W., Poore, G. C., van Soest, R. W., Stöhr, S., Walter, T. C., et al. (2013). Global coordina-

tion and standardisation in marine biodiversity through the world register of marine species (worms) and related databases. *PloS one*, 8(1):e51629.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. (1990). Indexing by latent semantic analysis. *JAsIs*, 41(6):391–407.

Desclés, J.-P. (2006). Contextual exploration processing for discourse and automatic annotations of texts. In *FLAIRS Conference*, volume 281, page 284.

Fleiss, J. L. and Cohen, J. (1973). The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*.

Hahn, R., Bizer, C., Sahnwaldt, C., Herta, C., Robinson, S., Bürgle, M., Düwiger, H., and Scheel, U. (2010). Faceted wikipedia search. In *Business Information Systems*, pages 1–11. Springer.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Jiang, J. (2012). Information extraction from text. In *Mining text data*, pages 11–41. Springer.

Juszczak, P., Tax, D., and Duin, R. (2002). Feature scaling in support vector data description. In *Proc. ASCI*, pages 95–102. Citeseer.

Manning, C. D., Raghavan, P., Schütze, H., et al. (2008). *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Marsi, E. and Oztürk, P. (2015). Extraction and generalisation of variables from scientific publications.

Marsi, E., Oztürk, P., Aamot, E., Sizov, G., and Ardelan, M. V. (2014). Towards text mining in climate science: Extraction of quantitative variables and their relations. In *Proceedings of the Fourth Workshop on Building and Evaluating Resources for Health and Biomedical Text Processing*.

Mimno, D. (2012). The details: Training and validating big models on big data. `https://vimeo.com/53080123`.

Piskorski, J. and Yangarber, R. (2013). Information extraction: Past, present and future. In *Multi-source, multilingual information extraction and summarization*, pages 23–49. Springer.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Ricardo Baeza-Yates, B. R.-N. (2011). *Modern Information Retrieval*. Pearson, Harlow, Essex, 2nd edition.

Ronen Feldman, J. S. (2007). *The Text Mining Handbook*. Cambridge, New York City, NY.

Salton, G. and Yang, C.-S. (1973). On the specification of term values in automatic indexing. *Journal of documentation*, 29(4):351–372.

Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Swanson, D. R. (1986). Fish oil, raynaud's syndrome, and undiscovered public knowledge. *Perspectives in biology and medicine*, 30(1):7–18.

Tran, T., Cimiano, P., Rudolph, S., and Studer, R. (2007). *Ontology-based interpretation of keywords for semantic search*. Springer.

Appendix A

Gold standard

Table A.1: Gold standard with an answer frequency for each relevance level, agreement level and the relevance rank between 0-2

Title	Agreement	Yes	Maybe	No	Relevance
Plankton	1.0000	11	0	0	2
Phytoplankton	1.0000	11	0	0	2
Ocean acidification	1.0000	11	0	0	2
Nanophytoplankton	1.0000	11	0	0	2
Microbial loop	1.0000	11	0	0	2
Benthic zone	1.0000	11	0	0	2
Anoxic event	1.0000	11	0	0	2
Antarctic krill	1.0000	11	0	0	2
Redfield ratio	1.0000	11	0	0	2
Marine biology	1.0000	11	0	0	2
Filter feeder	1.0000	11	0	0	2
Ocean	1.0000	11	0	0	2
Diatom	1.0000	11	0	0	2
Colored dissolved organic matter	1.0000	11	0	0	2
Lysocline	1.0000	11	0	0	2

Mixed layer	1.0000	11	0	0	2
Foraminifera	1.0000	11	0	0	2
Downwelling	1.0000	11	0	0	2
Coral	1.0000	11	0	0	2
Thaumatin	1.0000	0	0	11	0
Samir Shihabi	1.0000	0	0	11	0
Detritus	0.8182	10	0	1	2
Iron fertilization	0.8182	10	1	0	2
Marine snow	0.8182	10	1	0	2
Ecosystem of the North Pacific Subtropical Gyre	0.8182	10	1	0	2
Eutrophication	0.8182	10	1	0	2
Emiliania huxleyi	0.8182	10	1	0	2
Dissolved organic carbon	0.8182	10	1	0	2
Coral reef	0.8182	10	1	0	2
Algal bloom	0.8182	10	1	0	2
Coccolithophore	0.8182	10	1	0	2
Seawater	0.8182	10	1	0	2
Trichodesmium	0.8182	10	1	0	2
Pycnocline	0.8182	10	1	0	2
Primary production	0.8182	10	1	0	2
Nitrogen cycle	0.8182	10	1	0	2
Coral bleaching	0.8182	10	1	0	2
Ocean chemistry	0.8182	10	1	0	2
Hydrothermal vent	0.8182	10	1	0	2
Sea	0.8182	10	1	0	2
Cold seep	0.8182	10	1	0	2
Deep sea fish	0.8182	10	1	0	2
Abyssal plain	0.8182	10	1	0	2
Common octopus	0.8182	10	1	0	2
Deep sea communities	0.8182	10	1	0	2

Nitrification	0.8182	10	1	0	2
Cnidaria	0.8182	10	1	0	2
Total organic carbon	0.8182	10	1	0	2
Marine bacteriophage	0.8182	10	1	0	2
Wild fisheries	0.8182	10	1	0	2
Mosquito control	0.8182	0	1	10	0
Cover crop	0.8182	0	1	10	0
Biological soil crust	0.8182	0	1	10	0
Organic lawn management	0.8182	0	1	10	0
Ralstonia solanacearum	0.8182	0	1	10	0
Olsenbanden Jr. og Sølvgruvens hemmelighet	0.8182	0	1	10	0
Marine pollution	0.6727	9	2	0	2
Biological pump	0.6727	9	2	0	2
Anoxic waters	0.6727	9	2	0	2
Solubility pump	0.6727	9	2	0	2
Spring bloom	0.6727	9	2	0	2
Fisheries and climate change	0.6727	9	2	0	2
Cyanobacteria	0.6727	9	2	0	2
High-Nutrient, low-chlorophyll	0.6727	9	2	0	2
Ocean color	0.6727	9	2	0	2
Phosphorus cycle	0.6727	9	2	0	2
Paleoceanography	0.6727	9	2	0	2
F-ratio	0.6727	9	2	0	2
Periphyton	0.6727	9	2	0	2
Oligotroph	0.6727	9	2	0	2
Aquatic mammal	0.6727	9	2	0	2
Soil biodiversity	0.6727	0	2	9	0
Soil food web	0.6727	0	2	9	0
Belgica antarctica	0.6727	0	2	9	0
Rock dove	0.6727	0	2	9	0

Drizzle	0.6727	0	2	9	0
Ocean fertilization	0.6545	9	1	1	2
Bioirrigation	0.6545	9	1	1	2
Outwelling	0.6545	9	1	1	2
Submarine landslide	0.6545	9	1	1	2
Permafrost	0.6545	1	9	1	1
Fungiculture	0.6545	1	1	9	0
Root	0.6545	1	1	9	0
Deep sea	0.5636	8	3	0	2
Carbon cycle	0.5636	8	3	0	2
Oxygen minimum zone	0.5636	8	3	0	2
Benthic boundary layer	0.5636	8	3	0	2
Aquatic ecosystem	0.5636	8	3	0	2
Carbon	0.5636	8	3	0	2
Insect winter ecology	0.5636	0	3	8	0
Italian crested newt	0.5636	0	3	8	0
Plinthite	0.5636	0	3	8	0
Eichhornia crassipes	0.5636	0	3	8	0
Giant tube worm	0.5273	8	1	2	2
Deep sea creature	0.5273	8	1	2	2
Sea surface microlayer	0.5273	8	2	1	2
Baltic Sea hypoxia	0.5273	8	2	1	2
Microbial mat	0.5273	8	2	1	2
Euryhaline	0.5273	8	2	1	2
Atmospheric methane	0.5273	2	8	1	1
Ecosystem	0.4909	7	4	0	2
Effects of global warming on marine mammals	0.4909	7	4	0	2
Algal mat	0.4909	7	4	0	2
Climate change	0.4909	7	4	0	2
Carbon dioxide in Earth's atmosphere	0.4909	7	4	0	2

Carbon dioxide	0.4909	7	4	0	2
Biomass (ecology)	0.4909	7	4	0	2
Global warming	0.4909	7	4	0	2
Effects of global warming	0.4909	7	4	0	2
Ecological resilience	0.4909	7	4	0	2
Ice-ice	0.4909	7	4	0	2
Porites	0.4909	7	4	0	2
Mariculture	0.4909	7	4	0	2
Climate change feedback	0.4909	4	7	0	1
Isotope analysis	0.4909	4	7	0	1
Natural environment	0.4909	4	7	0	1
Permian–Triassic extinction event	0.4909	4	7	0	1
Environmental impact of pesticides	0.4909	4	7	0	1
Human impact on the environment	0.4909	4	7	0	1
Late Devonian extinction	0.4909	4	7	0	1
Plant litter	0.4909	0	7	4	1
Soil conservation	0.4909	0	4	7	0
Carbon sink	0.4545	6	5	0	2
Biogenic silica	0.4545	6	5	0	2
Carbon sequestration	0.4545	6	5	0	2
Fish farming	0.4545	6	5	0	2
Ecophysiology	0.4545	6	5	0	2
Bioindicator	0.4545	6	5	0	2
Trophic state index	0.4545	6	5	0	2
Ecosystem services	0.4545	6	5	0	2
Thermal pollution	0.4545	6	5	0	2
Long-term effects of global warming	0.4545	5	6	0	1
Algaculture	0.4545	5	6	0	1
Integrated multi-trophic aquaculture	0.4545	5	6	0	1
Physical impacts of climate change	0.4545	5	6	0	1

Azolla	0.4545	0	6	5	1
Aphanizomenon	0.4545	0	6	5	1
Climate change in Washington	0.4545	0	5	6	0
Marsh gas	0.4545	0	5	6	0
Fish kill	0.4364	7	3	1	2
Paleosalinity	0.4364	7	3	1	2
Methane clathrate	0.4364	7	3	1	2
Swim bladder	0.4364	7	3	1	2
Photorespiration	0.4364	7	3	1	2
Hydrogen sulfide	0.4364	7	3	1	2
Salt marsh dieback	0.4364	3	7	1	1
Plant nutrition	0.4364	3	7	1	1
Erosion	0.4364	3	7	1	1
Runaway climate change	0.4364	3	7	1	1
Lake ecosystem	0.4364	1	7	3	1
Substrate (aquarium)	0.4364	1	3	7	0
Soil respiration	0.4364	1	3	7	0
Gleysol	0.4364	1	3	7	0
Desert	0.4364	1	3	7	0
Soil crust	0.4182	2	2	7	1
Pedosphere	0.4182	2	2	7	1
Phosphorite	0.4182	2	7	2	1
Sulfur cycle	0.3818	6	4	1	1
Microecosystem	0.3818	6	4	1	1
Anaerobic oxidation of methane	0.3818	6	4	1	1
Anguillicoloides crassus	0.3818	6	4	1	1
Oil spill	0.3818	6	4	1	1
Soil erosion	0.3818	4	1	6	1
Paleocene–Eocene Thermal Maximum	0.3818	4	6	1	1
Carbon-to-nitrogen ratio	0.3818	4	6	1	1

Peat	0.3818	1	4	6	1
Constructed wetland	0.3818	1	4	6	1
Microplastics	0.3636	5	5	1	1
Aquaculture of salmonids	0.3636	5	5	1	1
Brevetoxin	0.3636	5	5	1	1
Methanogen	0.3636	5	5	1	1
Sapropel	0.3636	5	5	1	1
Freshwater environmental quality parameters	0.3636	1	5	5	1
Berlin Method	0.3636	1	5	5	1
Submarine eruption	0.3455	6	2	3	1
Human impact on the nitrogen cycle	0.3455	6	3	2	1
Marine pharmacognosy	0.3455	3	6	2	1
Eocene	0.3455	3	6	2	1
Natural hazard	0.3455	3	6	2	1
Permineralization	0.3455	2	6	3	1
Alcanivorax	0.3091	5	4	2	1
Planetary boundaries	0.3091	5	4	2	1
C4 carbon fixation	0.3091	4	2	5	1
Chlamydogobius	0.3091	4	2	5	1
Renewable resource	0.3091	4	5	2	1
Permafrost carbon cycle	0.3091	4	5	2	1
Altitudinal zonation	0.3091	2	4	5	1
Arundo donax	0.3091	2	4	5	1
Carbon credit	0.3091	2	4	5	1
Marine aquarium	0.3091	2	5	4	1
Bitter Springs type preservation	0.3091	2	5	4	1
Organisms involved in water purification	0.3091	2	5	4	1
Raceway (aquaculture)	0.3091	2	5	4	1
Blubber	0.2909	5	3	3	1
Black carbon	0.2909	5	3	3	1

Aquarium	0.2909	3	3	5	1
Reef aquarium	0.2909	3	3	5	1
Environmental impact of mining	0.2909	3	5	3	1
Ikaite	0.2727	4	3	4	1
Mudrock	0.2727	4	4	3	1