Improving information gathering for IT experts.

Combining text summarization and individualized information recommendation.

Förbättra informationsinsamling för IT-experter.

Kombinationen av textsammanfattning och individuell informationsrekommendering.

Anton Bergenudd
antonbergenudd@hotmail.com

Faculty of Health, Science and Technology

Master thesis in Computer Science

Second Cycle, 30 hp (ECTS)

Supervisor: Assoc. Prof. Sebastian Herold, University of Karlstad, sebastian.herold@kau.se

Examiner: Assoc. Prof. Muhammad Ovais Ahmad, University of Karlstad, ovais.ahmad@kau.se

Karlstad, June 20th, 2022

Abstract

Information gathering and information overload is an ever growing topic of concern for Information Technology (IT) experts. The amount of information dealt with on an everyday basis is large enough to take up valuable time having to scatter through it all to find the relevant information. As for the application area of IT, time is directly related to money as having to waste valuable production time in information gathering and allocation of human resources is a direct loss of profits for any given company. Two issues are mainly addressed through this thesis: texts are too lengthy and the difficulty of finding relevant information. Through the use of Natural Language Processes (NLP) methods such as topic modelling and text summarization, a proposed solution is constructed in the form of a technical basis which can be implemented in most business areas. An experiment along with an evaluation session is setup in order to evaluate the performance of the technical basis and enforce the focus of this paper, namely "How effective is text summarization combined with individualized information recommendation in improving information gathering of IT experts?". Furthermore, the solution includes a construction of user profiles in an attempt to individualize content and theoretically present more relevant information. The results for this project are affected by the substandard quality and magnitude of data points, however positive trends are discovered. It is stated that the use of user profiles further enhances the amount of relevant articles presented by the model along with the increasing recall and precision values per iteration and accuracy per number of updates made per user. Not enough time is spent as for the extent of the evaluation process to confidently state the validity of the results more than them being inconsistent and insufficient in magnitude. However, the positive trends discovered creates further speculations on if the project is given enough time and resources to reach its full potential. Essentially, one can theoretically improve information gathering by summarizing texts combined with individualization.

Keywords

Text summarization, information gathering, individualization, topic modelling, natural language processes, profiling.

Sammanfattning

Informationsinsamling
och informationsöverbelastning är ett ständigt växande problem för IT-experter. Mängden information som hanteras på en daglig basis är tillräckligt stor för att ta upp värdefull tid på så sätt att berörda behöver sålla igenom mängder med information för att hitta relevant innehåll. När det gäller tillämpningsområdet IT är tid direkt relaterat till pengar eftersom att utvecklare behöver lägga värdefull produktionstid på informationsinsamling vilket är en direkt ekonomisk förlust för de flesta företag. Denna avhandling hanterar i huvudsak två problem: texter är ofta för långa och svårigheter med att hitta relevant information. Genom användning av NLP-metoder såsom ämnesmodellering och textsammanfattning kommer ett förslag på en lösning att konstrueras i form av ett tekniskt underlag som kan användas vid implementetation till nya användingsområden. Ett experiment tillsammans med en utvärderingssession sätts upp för att utvärdera prestationen för den tekniska basen och förstärka fokuset av denna uppsats, nämligen ”Hur effektiv är textsammanfattning kombinerad med individualiserad informationsrekommendering för att förbättra informationsuppsamling för IT-experter?”. Dessutom inkluderar lösningen en konstruktion av användarprofiler i ett försök att individualisera innehåll och teoretiskt presentera mer relevant information. Resultaten för detta projekt påverkas av den undermåliga kvaliteten och storleken på datapunkter, men positiva trender upptäcks. Det anges att användningen av användarprofiler ytterligare ökar mängden relevanta artiklar som presenteras av modellen tillsammans med de ökande återkallelse- och precisionsvärdena per iteration och noggrannhet per antal uppdateringar som görs per användare. Det ägnas inte tillräckligt mycket tid med avseende på utvärderingsprocessens omfattning för att med säkerhet ange giltigheten av resultaten mer än att de är inkonsekventa och otillräckliga i omfattning. De positiva trenderna som upptäckts skapar dock ytterligare spekulationer om projektet får tillräckligt med tid och resurser för att nå sin fulla potential. I huvudsak kan man teoretiskt förbättra informationsinsamlingen genom att sammanfatta texter i kombination med individualisering.

Nyckelord

Textsammanfattning, informationsinsamling, individualisering, ämnesmodellering, naturliga språkprocesser, profilering.

Acknowledgements

I would like to give my thanks to all the people who contributed to the progress of this paper. Firstly, a substantial thanks to Elvenite who provided me with knowledge as well as the topic of this thesis itself. Further, a special thanks to Niclas Lovsjö, an employee at Elvenite at the time who personally helped me a lot with analyzing data, problem solving and all over generating ideas. Thank you to Agnes Lindell, also an employee of Elvenite at the time, who helped me generate the topic of this thesis. I would also like to give a special thank you to Sebastian Herold who acted as a supervisor for this thesis. Sebastian contributed with a lot of valuable input in many different areas throughout the text and continuously helped with feedback to this paper.

Acronyms

LDA	Latent Dirichlet Allocation
NLP	Natural Language Processes
NLU	Natural Language Understanding
NLG	Natural Language Generation
DF	Data Frame
AI	Artificial Intelligence
IP	Internet Protocol
SQL	Structured Query Language
AJAX	Asynchronous JavaScript and XML
TF-IDF	Term Frequency Inverse Document Frequency
IDF	Inverse Document Frequency
ML	Machine Learning
LSA	Latent Semantic Analysis
HTML	Hyper Text Markup Language
DOM	Document Object Model
NER	Name Entity Recognizer
CSV	Comma-Separated Values
TF	Term Frequency
URL	Uniform Resource Locator
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
FPR	False Positive Rate
BS	Beautiful Soup
NLTK	Natural Language Toolkit
AWS	Amazon Web Services
IT	Information Technology
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment

Contents

Chapter 1

Introduction

Under this chapter there is a general introduction in order to get a quick overview over the project. The motivation, goals, structure and problem description will first be presented. Then ethical discussions, presentation of stakeholders, delimitations of the project as well as a brief presentation of the methodology are all presented. The final source code for the evaluation website can be found under https://github.com/antonbergenudd/thesis-survey-app and the final source code for the pipeline can be found under https://github.com/antonbergenudd/exjobb.

1.1 Motivation

Information gathering today is a process of filtering through multiple sources of information in an attempt to find what is relevant. There is no easy way to manually scatter through these massive amounts of data presented through different sources of information efficiently, instead it is easy to simply get lost in a jungle of data where a lot of unnecessary time is spent on barren work. An area where information gathering is key is the area of IT development, more specifically software developers within IT who constantly have to find new information to help progress within their line of work. The everyday struggle of IT developers involves information gathering processes where tasks like bug fixing, general information seeking, information about specific programming problems or just information about their business area are all present. All of these areas have a common denominator in the form of a problem where having to find the correct information in as little time as possible is prevalent [35]. In Brian Dorn, Adam Stankiewicz and Chris Roggis' paper they enforce the fact that programmers are facing this issue with information seeking all too often within their everyday work which confirms the presented hypothesis [10].

1.2 Problem Description

A lot of time is spent trying to find relevant information to specific problems, the topic of information gathering becomes more and more popular involving both financial losses as well as negative impacts to the distribution of human resources to any given IT company. From this, there are three different problems depicted where two of them are approached through this thesis.

1. There are tons of informational sources to choose from.
2. The texts are too lengthy.
3. Difficult to find relevant information.

These three problems often boil down to a matter of time wasted which in business terms translates to a waste of money and hence becomes relevant problems for most companies. In terms of the first and second issue, the difficulty of keeping track of all informational sources for any individual creates a problem which calls for a need to further improve information gathering processes in order to save time and efficiently progress everyday work. Furthermore, the information provided by found sources often have a lot of text along with them, making them hard to read and might consist of irrelevant content. These massive amounts of information calls for a need to be reduced down to smaller more concise pieces of text which contextually gives the reader the same information but in less words. As to the third issue; first there has to be a relevant concise text reducing time needed to read any information provided by any informational source, second there has to be some sort of filter to remove potentially irrelevant information, basically by individualizing the information gathering process. These two issues forms the core of the problem which is sought to be solved through this project.

1.3 Thesis Goal

As mentioned, this project focuses mainly on addressing two problems, the issues of informational text being too lengthy and the difficulty in finding relevant information. The combination of these two proposed problems emerge into a tool of filtering out irrelevant information as well as reducing time needed to read relevant information, which together forms a strong suggestion of improving information gathering as a concept. The processes of individualizing and summarizing information forms the basis for this project, where the central hypothesis to address is ”How effective is text summarization combined with individualized information recommendation in improving information gathering of IT experts?” which further acts as the backbone of this thesis. Similarly to how Brian Dorn et al. proposes a solution to information gathering issues in the shape of a model [10], this thesis will propose a solution in the shape of a technical basis in an attempt to further help improve information gathering processes in general. Specifically for this project, the information seeking will be targeted to specific domain area information. Through the use of text summarization, topic modelling, and user profiling; a prototype is constructed which then is evaluated by a selected group of participants. The prototype makes use of newsletters as informational feed input in which the user is presented an information feed of summarized articles from the domain area. User feedback on the perceived relevance and quality of these articles is used to adapt the information feed individually.

1.4 Ethics and Sustainability

The techniques used and developed approach fall into the area of Machine Learning (ML), ML is an area within Artificial Intelligence (AI) which essentially boil down to algorithms being trained with data in order to discover hidden rules or patterns to solve a specific task [36]. When using ML there is always a risk of potential biases, this project is no exception to the rule since an algorithm computes user profiles and thus essentially breaks down human interests and human judgement into digits, which in turn introduces human biases [51]. The bias becomes obvious when trying to analyze the results of any arbitrary user profile, as for example, if a user only is presented with articles within one topic for some reason, the conclusion can eventually be that the user may only find one topic interesting and hence the construction of a human bias arises. To condense information for the sake of time saving is a process which eventually progresses into laziness and comfort. All research contributing to further development of human comfort may be seen as destructive in the far run as comfort is often payed with mental or physical challenges. Not exposing our minds and bodies to mental or physical challenges can be harmful to anyone to an extent [40]. Therefore an ethical issue arises if it is really a necessity to further condense informational sources and ease the process of manual filtering, relieving developers of their mental stimuli. In other words, will the development of this project affect human manual informational seeking skills negatively?
Although, as it can be considered a warning to try to optimize information seeking with the motivation of laziness, it also contributes to something good. The research has the potential to make informational gathering processes more sustainable by reducing the resources needed to perform the same amount of work today but in less time and effort. This may lead to, as earlier mentioned, improved management of human resources or a decrease in money spent on the work hours of developers trying to find information. Since the development of technology in general is progressing in a rate never seen before it is hard for users to adapt to user profiling and recommendation systems. A lot of people see profiling as an intrusive act where a machine tries to define them or also abuse personal information for company profits. User profiling may lead to users feeling exposed and insecure depending on the quality of the profiles. This will continuously be an issue as long as science keep on progressing in the current speed not letting the common man to keep up. User profiling also involves handling sensitive user information as the profiling can lead to detailed information about the user in the long run and thus the area is automatically a relevant ethical issue of today where

internet privacy is becoming bigger and bigger [34][54].

1.5 Methodology

As previously mentioned, this project aims to build a prototype of text summarization and individualization of information in the form of text using different NLP techniques along with an evaluation of the performance for the prototype.

1.5.1 Building the prototype

Existing newspapers are to be harvested of articles within the domain area of food and beverage. The collected article texts are then shortened into concise and contextually rich paragraphs with a shorter length than the original text. The articles are also given a user relevance prediction according to computed user profiles before they are presented on an article evaluation website.

1.5.2 Evaluation

In order to evaluate the relevancy of articles and compute user profiles there has to be some sort of tool to present articles and collect answers from users. By the use of prior knowledge within the area web development a customized website is set up where articles are presented and evaluated. Furthermore, the evaluation of the entire prototype will be a comparison between the predictions of user article relevance and the actual user article relevance input to the evaluation website.

1.6 Stakeholders

The benefits from this project can be useful to anyone who faces the issue of gathering information in any way. Since a basic foundation to improve information gathering is concluded from this project, one can simply apply this technical basis to their own domain area and make use of its computations. The focus for this project however is specifically software consultancy and will hence target aspects of software informational gathering issues. The target company in the attempt to prove the viability of this project is Elvenite. Elvenite specializes in the food and beverage area where they provide sustainable solutions for larger companies. With four offices scattered around Scandinavia, Elvenite has around 80 employees which all strive for a better tomorrow. Elvenite benefits from this project directly since Elvenite's employees faces the challenges of information seeking in their everyday work, having to find the latest relevant information about their domain application [11].

1.7 Delimitations

This project focuses on individualized and summarized information gathering, it will not be a prototype of newsletter generation nor focus on the best way to present the results of the project. The main focus is the technical basis regarding text summarization as well as information individualization. The constructed data crawler is created for the purpose of crawling specific web pages which helped the cause of this project. Hence, the data used is also conducted in a way promoting the domain area relevant to the prototype concept. The evaluation website is tailored to fit the needs of this project's algorithm features and will only provide results relevant to the prototype's produced articles. Lastly, the computed user profiles are not final products of user interests and will only represent the participating test users' interests within the specific application domain of Elvenite.

1.8 Outline

Under chapter 2 the theoretical background needed for this project is presented, introducing the reader to concepts and keywords needed to comprehend the context of the thesis. Chapter 3 describes the process of how the problem description is answered. Theoretical descriptions of parts of the prototype that needs to be implemented along side with some explanation. Chapter 4 introduces the implementation theoretically described under chapter 3. Which scripts were created, what they do and why they came to place, in other words a more hands on perspective of chapter 3. Moving on to chapter 5 the results from the project will be presented. The final accuracy percentage from the predictions will be presented as well as anonymous user profiles and summarized texts. Chapter 6 discusses eventual problems and further work extracted from the project. The author's experience with the project will also be at hand. Finally, chapter 7 introduces a final conclusion to the entire project. Answers to if the thesis question is answered, if the project could be considered a success, and thoughts about the results are presented.

Chapter 2

Background

Under this section a brief presentation of background information is presented in order for the reader to be able to understand the context of this thesis.

2.1 Natural language processing

Quoted from Kai Jiang and Xi Lu's paper, the definition of NLP is the following, "Natural language processing is defined as a discipline that studies language issues in human-to-human and human-to-machine communication" [18]. Some example tasks typically found within the area of NLP with a varying range of complexities starting with basic tasks are email filtering and auto-correct used in most smartphones. Examples of more tasks are virtual assistants and voice recognition software as for example Google's Alexa and Apple's Siri [3]. Voice recognition is basically programmed to receive input in the form of speech, convert it to text, process the text and respond with a correct answer to whatever the input was. Combining rule based modelling of human languages, agent intent and sentiment, AI can make precise conclusions about conversations or texts fully automatic [18]. Furthermore, two of the tasks relevant for this project are topic modelling and text summarization. Topic modelling is used for identifying hidden topics through the use of patterns across sets of documents or more specifically through any arbitrary corpus as in this case, further explanation of what a corpus is is found under section 2.2.2. A task man would have taken a lot of time to perform manually, if possible at all for the same quality. More information about topic modelling can be found under section 2.3. The task of text summarization takes one or more documents and reduces them to shorter more concise versions of the same documents. Under section 2.2 more details can be found about text summarization.

2.2 Text summarization

All over the internet today there is more and more information presented in innumerable different forms. The need for text summarization is evident in order to efficiently grasp all of the information presented. Performing the summarization by hand would take too much unnecessary time, therefore automatic text summarization is becoming more and more popular. The process of automatic text summarization could be defined as automatically extracting essential context from a text of length X reduced down to a length smaller than X making it more concise and efficient to read, all without the need of a human moderator. What then is considered a good summary is left to be decided by the eyes of the reader, since a summary might be good enough for someone who has prior knowledge in the subject but may not be extensive enough for someone who does not know anything about the subject. There are two methods of text summarization of today, extractive and abstractive.

2.2.1 Extractive vs abstractive

Extractive text summarization techniques extract the most relevant sentences from a text based on a calculated score and concatenates them together in order to form a more condensed version of the text, keeping the grammatical structure of the sentences. This way, the most important parts of a text, according to the scoring algorithm, no matter the size is included in the final summary.

+ Intuitively easy to evaluate and work with.
+ Easy to integrate.
+ Do not require any prior data or setup to be used.
– Relies on the original sentences being well structured.
– Excels in single document summarization though as the document number grows the longer the summary get.
– Hard to determine where to place sentences in relation to other extracted sentences.
– Hard to control the length of the summary.

The abstractive text summarization technique relies more heavily on AI to construct completely new sentences based on previously read or learned articles which the model has been fed. An AI is trained with an immense set of data in order for it to become familiar with the semantics and grammars of any existing language within the data set. Once it has been trained on the data set and eventually tweaked or re-learnt until it is considered good enough, the model can be fed a fresh article never seen before in order to construct its own paraphrased text summary.

+ Potentially optimal length of summaries.

+ Great at computing summaries over several different documents.
– Usually complex implementation and usage.
– Can take a lot of time to compute a readable text through the early stages of the model.
– Requires a large amount of data in order to train the model.

The abstractive summarization method is in general considered less accurate than the one of extractive summarization but has a bigger potential to create more condense texts as well as easier reads overall. Extractive text summarization tends to perform more consistent and overall better in most aspects of smaller projects [53]. Further, a presentation of both an extractive summarizer as well as an abstractive summarizer, taken from Ramsri Goutham's example summary in his paper about simple abstractive text summarization [45]. The original text is shown below.

The US has ”passed the peak” on new coronavirus cases, President Donald Trump said and predicted that some states would reopen this month.The US has over 637,000 confirmed Covid-19 cases and over 30,826 deaths, the highest for any country in the world.At the daily White House coronavirus briefing on Wednesday, Trump said new guidelines to reopen the country would be announced on Thursday after he speaks to governors.”We'll be the comeback kids, all of us,” he said. ”We want to get our country back.”The Trump administration has previously fixed May 1 as a possible date to reopen the world's largest economy, but the president said some states may be able to return to normalcy earlier than that.

The final result of using an extractive summarizer is shown in listing 2.1.

```
The US has "passed the "peak on new coronavirus cases, President Donald
    Trump said and predicted that some states would reopen this month.The
    US has over 637,000 confirmed Covid-19 cases and over 30,826 deaths,
    the highest for any country in the world.At the daily White House
    coronavirus briefing on Wednesday, Trump said new guidelines to reopen
    the country would be announced on Thursday after he speaks to governors
    ".'Well be the comeback kids, all of us", he said.
```

Listing 2.1: Example results of an extractive summarization technique.

The sentences that were not included for the summary are the following:

- ”We want to get our country back.”
- The Trump administration has previously fixed May 1 as a possible date to reopen the world's largest economy, but the president said some states may be able to return to normalcy earlier than that.

These sentences were left out due to the fact that a sentence scoring algorithm considered these sentences to be off less importance, compared to the total average sentence score.

Further, the mentioned abstractive summarization example from Ramsri Goutham's paper can be seen in listing 2.2 computed on the same original text.

```
The us has over 637,000 confirmed Covid-19 cases and over 30,826 deaths.
    President Donald Trump predicts some states will reopen the country in
    April, he said. "we'll be the comeback kids, all of us," the president
    says.".
```

Listing 2.2: Example results of an abstractive summarization technique.

The cornerstone of the extractive text summarization is the importance of determining a strategy of selecting sentences to be extracted. A common strategy to apply is simple sentence scoring, where sentences are given a score based on whatever the user finds important, and the top X scoring sentences are selected for the final summary. A concept called Term Frequency Inverse Document Frequency (TF-IDF) plays a key role in this sentence scoring for this project [53].

2.2.2 TF-IDF

To compile a concise version of a text using an extractive text summarizer, there needs to be some sort of scoring method to decide how informative a sentence is, a common scoring algorithm to use for this is called TF-IDF. TF-IDF targets to measure a word's relevance inside of a document which in turn is part of a collection of documents, a so called corpus [6]. Through this project a corpus is represented by a two dimensional array containing a collection of documents along with their respective word frequencies. A dictionary is all words occurring within a set of documents collected into one vector which can be described as a lexicon for all documents. To understand how the scoring calculation works within TF-IDF each term of TF-IDF are separately described starting with Term Frequency (TF). The words across one or more documents are gathered as well as the frequency of each word is counted, and later the words are saved into the form of a corpus of the documents, which then represents the term frequency (TF). Inverse Document Frequency (IDF) is then used to calculate how unique a word is across one or more documents which helps putting weight to more informative words. These two measurements TF (word frequency) and IDF (word weight) are then multiplied resulting in a final score for each word inside the entire corpus. Finally, each sentence within the documents can be given a score according to each word's TF-IDF score. The highest scoring sentences are most likely the sentences with the highest contextual information based off of the constructed corpus [7]. The mathematical definition of TF-IDF can be seen below where x represents a word, y represents a document where the word x can be found, tf represents the term frequency, df represents the document frequency, and N is the total number of documents used. The product of a specific term frequency and the logarithmic computation of the total number of documents over the number of documents containing a specific word results in the final TF-IDF score for the given word based on a set of documents.

$$w_{x,y} = tf_{x,y} * log(\frac{N}{df_x})$$

2.2.3 Multi- vs single document

Summarizing multiple documents focuses on extracting essential points over a number of documents all somewhat related to each other in terms of topic distribution as well as combining these points into a joint summary. Summarizing a single document simply means to reduce a longer text into a more concise text of smaller length than of the original version [48].

2.3 Topic modelling

Topic modelling is an ML approach used to identify hidden topics within a set of documents, where during the process, words are clustered from one or more documents to form the topics. As seen in figure 2.3.1 there are many application areas for a topic model, the figure in specific shows application areas for industrial usages. The topics are efficiently used when clustering related documents by simply

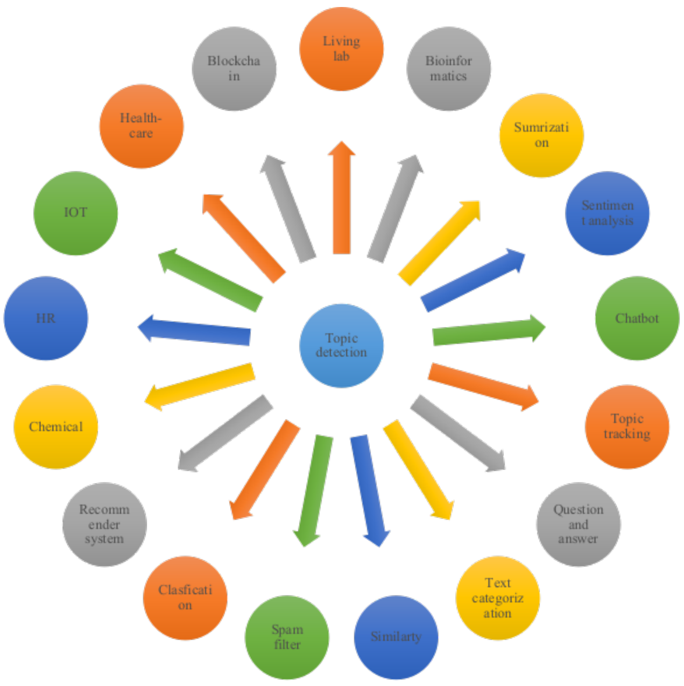


Figure 2.3.1: Industrial application of a topic model [13].

comparing the topics assigned by a topic model to the documents. There are several different models to use when speaking of topic modelling where the most notable ones are the Latent Dirichlet Allocation (LDA) model and the Latent Semantic Analysis (LSA) model [20]. Both LDA and LSA expects documents in the form of a bag-of-words format which represents word frequencies across all documents. Bag-of-words simply refer to the way each document is represented, which is an array of words along with their word frequencies inside of the document, which in turn infer that LDA and LSA both ignore syntactic composition of sentences [44]. A document can further be divided into k number of topics which are based on the grouping and frequencies of the words inside the document. Each topic has a set of words which per se has an importance score equivalent to the word's contribution to the topic itself. The words inside of a topic are represented as clusters where the outline of a cluster can differentiate a lot depending on the different setup of calculations used for computing topics. These topics are then used as a basis for the topic distribution given to each respective document input to the model where each topic is given a percentile as to how much the document correspond to that specific topic. This process is made possible by the use of topic modelling where this project will make use of an LDA topic model. Since LDA and LSA are fairly similar in performance as well as similar in usage the only key difference for this project is that LSA requires a larger data set whereas a smaller data set is used in this thesis [20].

2.3.1 Latent Dirichlet Allocation

LDA is an unsupervised probabilistic approach to labeling documents whereas a predetermined number of topics are given a percentage of relevance to each document fed to the model. The abbreviation LDA stands for Latent Dirichlet Allocation, where Latent indicate finding hidden topics by the use of patterns through a corpus, Dirichlet is a probability distribution algorithm which here indicates that the distribution of the words in the topics as well as the topics per se are distributed in a Dirichlet manner, the output is a vector of probability distributions, and Allocation refers to the distribution of topics within a document [6]. The core process of LDA begins with randomly assigning a topic to each word within a corpus of documents. Each topic for each word is then counted inside the entire corpus in order to summarize how many times a word is associated with a specific topic. Now there is a count of how many times a word is associated with a specific topic inside the corpus as well as how many times a topic appears in a specific document inside the corpus. After this random initialization of topics, the process begins to make an attempt of converging the words to a non-randomized topic. This is done by removing the currently assigned random topic one by one and re-process the word accordingly. The reassignment is done through the use of two calculations presented below.

1. How many times a word appears inside a topic based on all documents inside a corpus.
2. How many times a topic appears within a specific document.

Let us call a word from any arbitrary iteration X inside document Y. The first calculation is how many appearances X has in each topic based on other documents than Y. The second calculation calculates how many times a topic appears inside of Y. Lastly, the product between these two calculations results in a final assignment of the topic with the highest product, and so the iteration continues for every word in the corpus, giving each word a topic according to the highest possible product of the two calculations. How many times this iteration takes place is determined during the LDA setup where the number of iterations is called `passes` [6]. LDA expects a dictionary as well as a corpus which represents the scope of words that are to be used for the task. The words necessary for the model has to be processed in order to optimize the performance of the model. There are several optional methods of processing the data beforehand, whereas one necessary method is the so called stop word removal method. LDA expects the words delivered to the task to be of certain importance, meaning that words without any context such as stop words should be removed beforehand, otherwise the model will not be able to compute accurate topics [20]. ”a”, ”with” or ”can” is an example of what stop words could look like, intuitively it is possible to see that these words would not contribute to any context if part of a topic.

2.4 Evaluation techniques

Through this project’s technical approach there are two things in need of evaluation. The first is an evaluation of the topic model performance and the second is an evaluation of the constructed article relevance scoring approach.

2.4.1 Topic model

Because of the complexity of evaluating a topic model, one of the most simple ways of evaluating the performance of a topic model is to simply eye ball the results and manually judge if the topics makes any sense. This seemingly trivial approach of evaluation becomes an issue when wanting to confidently state the quality of generated topics from a NLP based model, as the results are presented in a text further obeying syntactic rules and grammars. As LDA has an unsupervised training approach which makes the process of evaluating the model hard. The evaluation comes down to human intuition. The evaluation can be divided into two approaches, using human judgement or using intrinsic evaluation metrics to try to prove the quality of generated topics. Typically one can identify the top few words from each topic to determine if the words correlate to one another. This, combined with looking at which documents are assigned to which topics, gives the user a good sense as to the performance of the model.

Furthermore, an intrinsic evaluation metric which measures relevance of words within a specific topic is called the coherence score. Topic coherence helps distinguish between interpretable topics and topics created from simple statistical inference. Coherence per se suggests that a set of statements or facts, in this case patterns, enforce one another. By optimizing the coherence score one can find more relevant and interpretable topics created by the LDA model. Lastly, one can use simple human judgement to determine what a topic is in order to judge whether or not the LDA model performs well or not, which of course leaves room for human errors [52].

2.4.2 Article relevance scoring

The evaluation of the approach for article relevance scoring boil down to the question of if the proposed approach is good enough for the constructed personalized recommendation system. One efficient way of setting up this evaluation is to focus on user evaluation feedback, where it is common to use a survey to collect data as it is a great tool to quickly assess and group user feedback from many different project types. As per definition a survey can be described as ".. to obtain the same kind of data from a large group of people (or series of events), in a standardized and systematic way."[39]. Furthermore, there are different ways to setup a survey where the choice between a quantitative and a qualitative survey has to be made. If a survey has a questionnaire with answers scaled in numbers or any other scalar measurement then it is a quantitative survey, as opposed to if it has elaborate answers then the most fitting type of survey is most likely a qualitative survey. One can also determine if a quantitative or qualitative survey is needed if the requested data collected from the surveys are in a numeric or textual format which most often distinguish the two, where the former is for quantitative surveys and the latter for qualitative [1].

2.5 Related Work

A number of papers were found with the focus of extending the knowledge and getting a greater understanding of information seeking problems and approaches. These articles are presented below.
Brian Dorn, Adam Stankiewicz and Chris Roggi conducts a study where they setup an experiment in a lab setting in order to answer two central questions "What foraging and search behaviors are employed by end-user programmers to overcome learning barriers?" and "To what extent do information seeking behaviors correlate with various measures of success in completing scripting tasks?". They mention the difference between information seeking for people without prior knowledge about programming opposed to people with prior knowledge. Their results came to be that both parties had some issues with the source of the information and not the actual act of information seeking itself. Furthermore, they found information about how the users found the information which essentially is the method of information seeking. The amount of testers were low with a bare 19 people attending the experiment and thus the results may vary a lot depending on the attending people. As the authors mentioned in the paper another limitation with the study was that they did not allow participants to communicate with friends or colleagues which limits a part of a general information seeking approach. Even though the amount of participants were low, the results are still useful for future research within the area of information seeking as it breaks down the basic methods of information seeking regardless of prior knowledge. This work is a more hands on approach of trying to find the essence of the issues with information seeking, the authors focus on research and expanding the knowledge about information seeking instead of trying to find any improvement to the subject

[10].
Yihan Lu, I-Han Hsiao and Qi Li uses a different approach of study where they explore popular sources of information specifically for programmers seeking information. In the work they first study how students search and explore the different online forums of programming. The purpose of the paper is to explore and analyze information seeking behaviour of programmers navigating programming discussion forums. A mentioned limitation of the study by the authors were for example searches made in Google which redirected users to a programming discussion forum could not be registered for this experiment. This paper however further helps the understanding of how information seeking is conducted in a more controlled environment within a specific domain area. This work is directly related to this thesis as the general knowledge about information gathering gained is applied to the same foundational thinking. As this thesis focus on constructing a technical basis for improving information seeking, the study conducted by Yihan Lu, I-Han Hsiao and Qi Li focus on gaining knowledge about the act of information seeking within the controlled environment earlier mentioned [32] .
A study to investigate the interaction effect of task difficulty and domain knowledge on user's search behaviors was conducted by Chang Liu et al. where the task of information gathering was thoroughly examined. The study conducted was essentially an experiment where people of different levels of domain knowledge was presented with different levels of tasks ranked easy or hard. The participants were then monitored as to how much dwell time was spent for each participant and which informational sources they entered. The study could have included more levels of difficulty for the tasks in order to retrieve a more nuanced result. The conclusion however was that users that had a higher domain knowledge, spent less time on pages users of less domain knowledge spent a lot of time on, showing a difference in information gathering regarding previous domain knowledge. This finding contributes to the understanding of information gathering as to how users navigate using previous experiences as well as it poses an example of how much time is spent on trying to find related content. This work is proof as to why the area of information gathering need more work in every aspect as much time is spent in dwell time trying to find the relevant information in for example different documentations [28].
Furthermore, there are papers who present a solution of improving information seeking in different areas. These articles are presented below.
Michael P. O'Brien and Jim Buckley discuss the problems of information seeking as well as provide a solution in the form of a model specifically for programmers in their paper named "Modelling the Information-Seeking Behaviour of Programmers – An Empirical Approach". The paper focus on reviewing existing models of information seeking in order to propose a better non-linear model as opposed to the existing model they present, along side this the authors explore the definitions and scope of information seeking in order to try to comprehend the entirety of the problem [38]. The limitation of this paper is the small amount of case studies proposed whereas the scenarios might be specifically beneficial for their case. The benefits from their paper however is that it proposes a model of thought processes to faster and better gain information and reach a solution faster than the compared models. As to how

this is related to the work of this thesis is that the main goal of improving information seeking is evident.
An attempt to improve information gathering in terms of source code information extraction is made by Paul Gross and Caitlin Kelleher in their paper about non-programmers attempt to understand source code and retrieve information. The paper presents a study performed where non-programmers were gathered and given a task of finding specific functionality in open-source code. The task involved barriers such as interpreting the code and formulating correct search queries to find the correct section in the source-code. Quoted from the paper the purpose is ".. explore non-programmers' natural search processes to guide the design of new tools that will explicitly support natural search processes". The authors further suggest design guidelines for software supporting non-programmers in order to contribute to the area of information gathering but specifically for the purpose of users with little to no competence about the area. Limitations of the study is the instrumentals used for the study, namely the Storytelling Alice software used. The findings from this paper is highly relevant as it emphasizes the gap found in information gathering for people without prior knowledge of IT processes. The issue of information gathering is not only relevant speaking of programmers, but may also be applied to non-programmers who has less experience in the matter [17].
Another paper implemented information ranking and selection in order to improve programming productivity. Mik Kersten and Gail C. Murphy identified a waste of productivity in programmers having to switch tasks whereas the authors present a task context model in order to prevent information overload. The proposed technical implementation is validated and proven to improve the productivity of the users. The identified problems by the authors are that often programmers work with more than one source repository which contradicts with the found assumptions of the Integrated Development Environment (IDE) they focus their work on, as well as the information gap between existing modules and different programmers. The authors propose limitations in their work such as a misinterpretation of input and issues with their implementation as for example when handling multiple related tasks. This work add technical depth to information gathering issues as well as the physical solution in the shape of software. It is an alternative approach to solving the same issue as presented in this thesis however specifically targeting a programmer's IDE [21].
To summarize the findings about related work, there are a lot of articles who focus on extending the knowledge about information gathering issues within the area of IT. A few articles found propose a solution to specific problems where technical implementations are described. However, no paper found provided a technical basis constructed from text summarization and individualization in an attempt to improve information gathering making this thesis project unique in contribution.

Chapter 3

Methodology

Under this chapter one can find a theoretical explanation of the entire project pipeline, explaining each step throughout the process. First a brief overview is presented in order to grasp the full context, the following sections explains each step within the pipeline more thoroughly.

3.1 Overview

In order to answer the question ”How effective is text summarization combined with individualized information recommendation in improving information gathering of IT experts?” introduced under section 1.2, the following scenario is setup. A simulation of a newsletter feed is constructed by the use of scraped articles from selected news sources. Then users of the tool are to evaluate the articles' context and understandability systematically and iteratively in order to analyze a progressive process of article selection. The articles are first scraped, processed, and topic distributed, then presented to the users, evaluated and lastly used to update the user profiles which eventually reflects the project results. The accuracy of the user profile represents the accuracy of the entire process. Figure 3.1.1 visualizes the steps of the

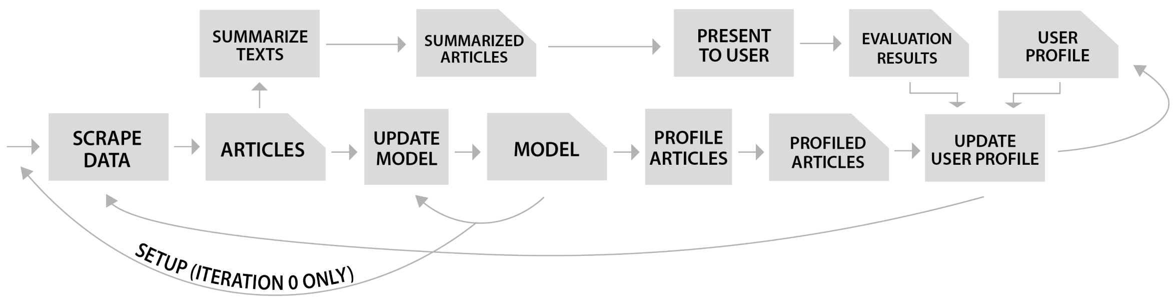


Figure 3.1.1: Visualization of the overall approach.

entire project pipeline. For this project there is a differentiation between the first iteration run, the so called iteration zero, and the coming iterations. The first iteration is run in order to setup and prepare data for the coming articles, as for example creating the topic model, generating user profiles and populating the database with an initial batch of articles. The coming iterations are run with the purpose of refining this topic model, improving the user profiles and further extend the database with data. The processed steps from figure 3.1.1 through the first iteration is limited to ”scrape data” and ”update model” only. All available articles up until a given date are scraped from two data sources. The topic model is then updated based on these scraped raw articles in order to continuously be able to distribute topics over articles with words never seen before. This makes the article topic distribution keep the same logical standard as the process progresses. Under section 3.3 this is further described. After the articles are scraped, they are run through a data pre-processing step where the data is refined in order to make the computed topics of higher relevance. This is described in more detail under section 3.2. As for the first iteration the scraping process differs as it is setup to scrape a larger amount of data instead. The data is collected from the same data sources as the coming iterations, though as much data as possible is collected instead of limiting the data size by a given date. Furthermore, the first set of articles for the coming iteration after iteration zero are not scraped but rather a careful selection of 15 articles are extracted from the larger data set. The purpose of this is to retrieve articles of minimal similarity in terms of the articles’ internally distributed topics. In this way the users’ initial profile creation process is enhanced, where greater adjustments can be made initially through the variation of selected articles.

Continuing on the path for the coming iterations, the articles are then used in both of two ways, one which is text summarization and one model updating. The first path described is the update model path where the LDA model is updated, introduced under section 2.3. As for iteration zero, the model is not yet created and therefore is here setup and saved for coming iterations. Using the LDA model, articles are profiled where they are given topic distributions based on the setup of the model. Further details are given in sections 3.3 and 3.4.

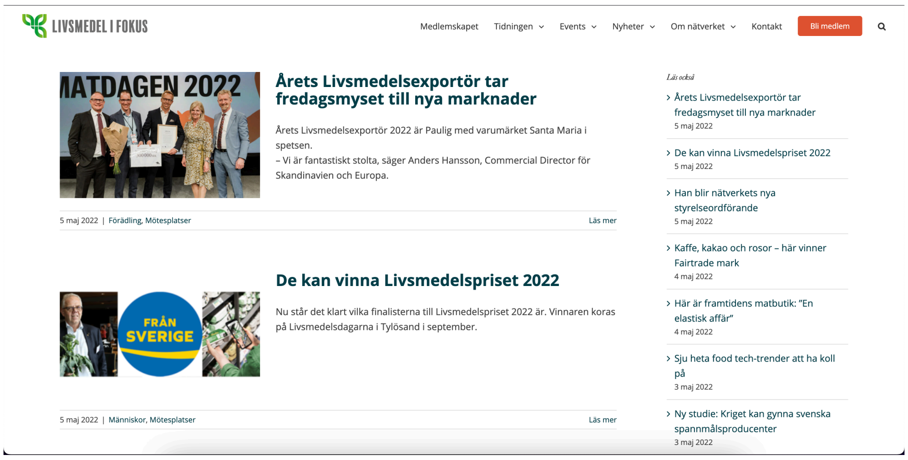
The results of the evaluation process combined with the profiled articles are used to update each respective user’s profile based on their respective answers. This is discussed more under 3.7. As to the second path, deviating from the path of updating the model, the text summarization process summarizes articles into more condensed versions of the texts. More information about this can be found under section 3.5. The summarized texts are further presented to the users who evaluate the articles via a website discussed under section 3.6. The website is essentially setup in order to generate evaluation results which are used for the update of the user profiles previously mentioned.

3.2 Scrape data

To simulate a newsletter feed, a scraping of article data from different newsletter organisations is completed to gain training data for the project. The information needed is found on websites containing domain specific news which frequently publishes new articles. In order to retrieve this data one needs to understand the concept of a Document Object Model (DOM) tree which will be used for this purpose. A DOM tree is a website's logical structure described with tags, elements within Hyper Text Markup Language (HTML). Inside the DOM tree, the entire website's content is located, as well as the article data which is to be scraped [47]. For this purpose, two Swedish data sources will be used ”www.livsmedelifokus.se” and ”www.livsmedelsnyheter.se”. The selection mainly originates from tips by colleagues at Elvenite combined with trivial research to see which data sources could be found useful followed by quick analyses over how many articles the data sources publish as well as the frequency and quality of the articles.

3.2.1 Data sources

Livsmedelifokus is essentially a network for people working with or are interested in the food and beverage industry. They have their own newsletter which consists of articles as the ones used for the mentioned data set. The network is headed by representatives across the entire food and beverage chain, giving the network a broad perspective on the topic [30]. The process of scraping was made possible by, as earlier mentioned, inspecting each website's DOM structure to identify key elements to extract. As can be seen in figure 3.2.1 the data which are of relevance in livsmedelifokus.se are the articles seen to the left, more specifically the article links are of importance in this matter.



Årets Livsmedelsexportör tar fredagsmyset till nya marknader

Årets Livsmedelsexportör 2022 är Paulig med varumärket Santa Maria i spetsen.
– Vi är fantastiskt stolta, säger Anders Hansson, Commercial Director för Skandinavien och Europa.

5 maj 2022 | Förädling, Mötesplatser Läs mer

De kan vinna Livsmedelspriset 2022

Nu står det klart vilka finalisterna till Livsmedelspriset 2022 är. Vinnaren koras på Livsmedelsdagarna i Tylösand i september.

5 maj 2022 | Människor, Mötesplatser Läs mer

Läs också

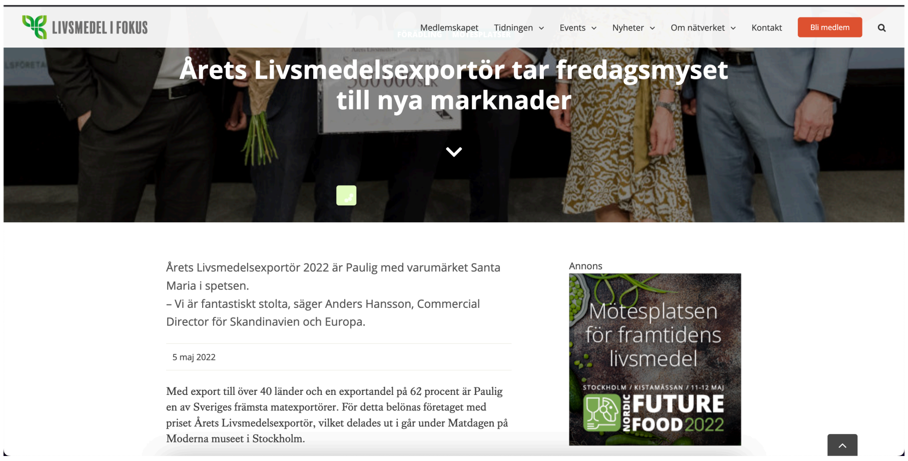
- Årets Livsmedelsexportör tar fredagsmyset till nya marknader
  5 maj 2022
- De kan vinna Livsmedelspriset 2022
  5 maj 2022
- Han blir nätverkets nya styrelseordförande
  5 maj 2022
- Kaffe, kakao och rosor – här vinner Fairtrade mark
  4 maj 2022
- Här är framtidens matbutik: "En elastisk affär"
  4 maj 2022
- Sju heta food tech-trender att ha koll på
  3 maj 2022
- Ny studie: Kriget kan gynna svenska spannmålsproducenter
  3 maj 2022

Figure 3.2.1: Image of livsmedelifokus.se/nyhetsarkiv/.

Once the article links have been collected, the article web page can be scraped. Seen in figure 3.2.2 the entire article information can be seen on this page.

Here the data of importance is the title found in bold on top of the page, the publish date found just beneath the title in a smaller greyish color and lastly the body of the article which are all paragraphs following the publish date.

Livsmedelsnyheter on the contrary is a daily news website also within the area of food



Årets Livsmedelsexportör tar fredagsmyset till nya marknader

Årets Livsmedelsexportör 2022 är Paulig med varumärket Santa Maria i spetsen.
– Vi är fantastiskt stolta, säger Anders Hansson, Commercial Director för Skandinavien och Europa.

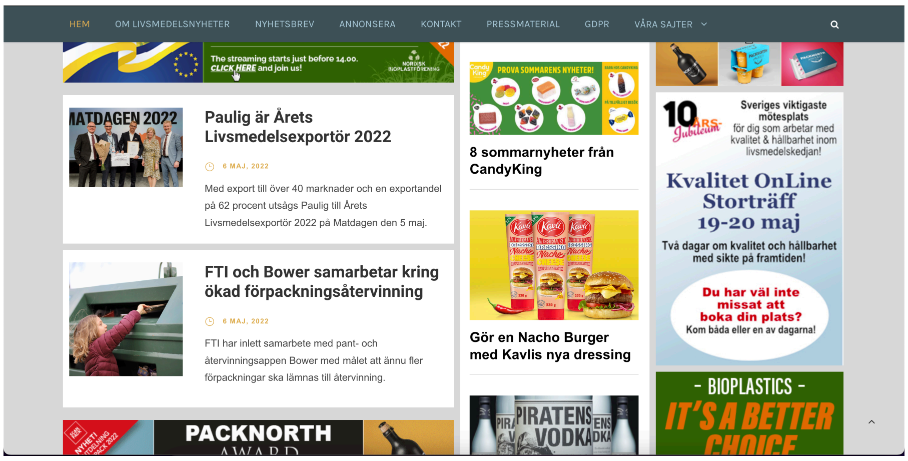
5 maj 2022

Med export till över 40 länder och en exportandel på 62 procent är Paulig en av Sveriges främsta matexportörer. För detta belönas företaget med priset Årets Livsmedelsexportör, vilket delades ut i går under Matdagen på Moderna museet i Stockholm.

Annons
Mötesplatsen för framtidens livsmedel
STOCKHOLM / KISTAMÄSSAN / 11-12 MAJ
NORDIC FUTURE FOOD 2022

Figure 3.2.2: An article from livsmedelifokus.se/nyhetsarkiv/.

and beverage which has its own newsletter as well as an online news feed [31]. Seen in figure 3.2.3 data of importance is the articles found to the left in the image, the website layout is similar to the one of livsmedelsnyheter and hence the same logic applies.



Paulig är Årets Livsmedelsexportör 2022

6 MAJ, 2022

Med export till över 40 marknader och en exportandel på 62 procent utsågs Paulig till Årets Livsmedelsexportör 2022 på Matdagen den 5 maj.

FTI och Bower samarbetar kring ökad förpackningsåtervinning

6 MAJ, 2022

FTI har inlett samarbete med pant- och återvinningsappen Bower med målet att ännu fler förpackningar ska lämnas till återvinning.

8 sommarnyheter från CandyKing

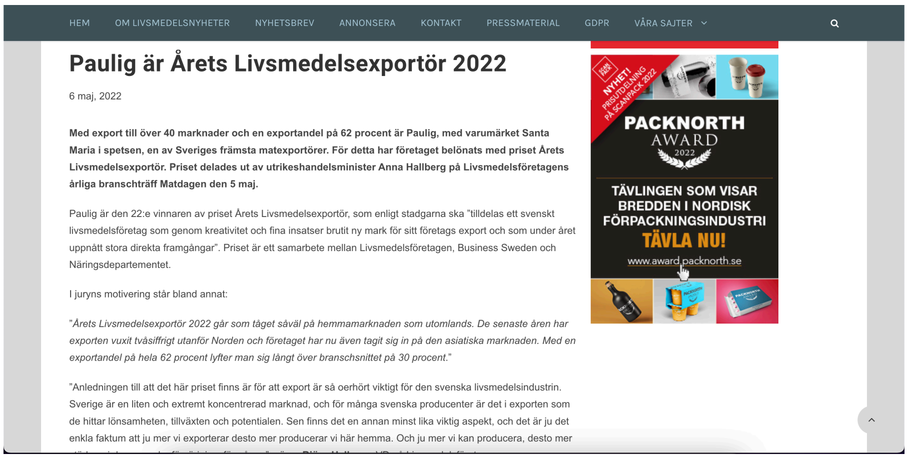
Gör en Nacho Burger med Kavlis nya dressing

PROVA SOMMARENS NYHETER!
10 ÅRS-Jubileum
Sveriges viktigaste mötesplats för dig som arbetar med kvalitet & hållbarhet inom livsmedelskedjan!
Kvalitet OnLine Storträff 19-20 maj
Två dagar om kvalitet och hållbarhet med sikte på framtiden!
Du har väl inte missat att boka din plats? Kom båda eller en av dagarna!
PACKNORTH AWARD
PIRATENS VODKA
- BIOPLASTICS - IT'S A BETTER CHOICE

Figure 3.2.3: Image of livsmedelsnyheter.

The article links are retrieved and iterated in the same way where the layout of any arbitrary article page is displayed in figure 3.2.4.

The data of importance on this page are the title found on the top of the page, the publish date found between the initial paragraph and the content of the article, and lastly the content of the article.



Paulig är Årets Livsmedelsexportör 2022

6 maj, 2022

Med export till över 40 marknader och en exportandel på 62 procent är Paulig, med varumärket Santa Maria i spetsen, en av Sveriges främsta matexportörer. För detta har företaget belönats med priset Årets Livsmedelsexportör. Priset delades ut av utrikeshandelsminister Anna Hallberg på Livsmedelsföretagens årliga branschträff Matdagen den 5 maj.

Paulig är den 22:e vinnaren av priset Årets Livsmedelsexportör, som enligt stadgarna ska ”tilldelas ett svenskt livsmedelsföretag som genom kreativitet och fina insatser brutit ny mark för sitt företags export och som under året uppnått stora direkta framgångar”. Priset är ett samarbete mellan Livsmedelsföretagen, Business Sweden och Näringsdepartementet.

I juryns motivering står bland annat:

”Årets Livsmedelsexportör 2022 går som tåget såväl på hemmamarknaden som utomlands. De senaste åren har exporten vuxit tvåsiffrigt utanför Norden och företaget har nu även tagit sig in på den asiatiska marknaden. Med en exportandel på hela 62 procent lyfter man sig långt över branschsnittet på 30 procent.”

”Anledningen till att det här priset finns är för att export är så oerhört viktigt för den svenska livsmedelsindustrin. Sverige är en liten och extremt koncentrerad marknad, och för många svenska producenter är det i exporten som de hittar lönsamheten, tillväxten och potentialen. Sen finns det en annan minst lika viktig aspekt, och det är ju det enkla faktum att ju mer vi exporterar desto mer producerar vi här hemma. Och ju mer vi kan producera, desto mer

Figure 3.2.4: An article from livsmedelsnyheter.

3.2.2 Data pre-processing

In order to get the best results from the topic model computation, there is a need to clean the data before using it for the model. This process includes several steps such as removing punctuation, converting all words to lower case, removing stop words, tokenizing texts and lemmatizing words. The general motivation for pre-processing the data is to remove words that are considered non-informational words, making each word within a data set as informative as possible [52]. For this project the following steps are used for the pre-processing step.

- Removal of stop words, punctuations, and numbers
- Converting words to lower case.
- Identify and remove names of people.
- Finding the root of each word.
- Tokenize texts.
- Identify bigrams.

Removal of stop words, punctuation, and numbers

The removal of punctuation is the simple process of removing special characters such as ”!#/(=?...” since they do not contribute to any informational context in most cases but rather helps the reader to interpret the sentences correctly. Furthermore, numbers are removed since yet again they mostly did not contribute to any context in this scenario of NLP after analyzing the generated topics. All of the words are further converted to lower case in order to be able to compare capitalized words to non-capitalized words, specifically important during the process of stop word removal as the specified stop words are in lower capital letters. Lastly, the stop words are removed as it is known to not contribute to the context of documents but rather help in reading sentences efficiently. Examples of stop words are as following: ”bara, vara, vi, ni, ta, åtta, etc..", as can be seen, these words are intuitively not rich with information.

Identification and removal of names

In order to remove names from a text there first needs to be some kind of algorithm to identify and define what a name is within a text, a typical task to use for this is the NLP task called Name Entity Recognizer (NER). The essence of a NER task is to identify all key words or entities within a text, for example, names, places, times, company names etc. NER iterates a text where the output is a list of words with their corresponding entities. For this project a NER was solely used to identify names of people as this was considered to be of negative impact for the LDA model to consider for distributing topics [41].

Finding the root of words

This process is essentially performed in order to prevent conjugates of words to occur, as for example ”running” and ”run”. The process more or less identifies the dictionary form of a word, this is often referred to as lemmatizing. This will further improve the LDA model topic distribution as removing conjugate words will reduce repeated words to inflate a higher score during the topic computation [29].

Tokenizing

Tokenizing simply means to iterate sentences and represent the words in the shape of an array instead of its original text format. Essentially converting a text into its atomic elements. This way it is simpler to work with and manipulate programmatically [52].

Bigrams

Identifying bigrams is an optional step which is included due to its relatively good performance. Bigrams can be described as the word suggest: a pair of words, as for example ”Coca Cola”, if two words often occur together they should be handled as one word as for the topic computations of the LDA model [9].

3.3 Update model

As the LDA and LSA models are similar in performance and usage, LDA is chosen as topic model for this project due to the simple fact that LSA requires a larger data set [20]. During the setup phase mentioned under section 3.1 the initial LDA model is created which then is used for future iterations during the topic distribution computations. The LDA model is given a corpus and a dictionary setup from a number of articles as input, the output is then a cluster of x number of topics represented as a statistical distribution where each document is assumed to be a mix of topic distributions. Before using the LDA model to compute topic distributions for new articles, the LDA model first needs to be given a dictionary encapsulating the scope of the model's words. Essentially the model has to be taught which words should be included and which are of weight for the topic distributions for the articles. This model is then saved for future use where it later can be loaded and fed a new corpus representing a smaller amount of articles to be given topic distributions. To the left in figure 3.3.1 one can see the principal component representation of clustered topics which is computed by the LDA model. Principal component is used when visualizing data and helps presenting data sets of higher dimensions by reducing them to lower dimensions easier to comprehend. As for example, seen in figure 3.3.1 a two dimensional space is presented instead of the original dimension of number of topics-space [19]. Clusters that are closer together in the principal component presentation are seemed to be of closer relevance to one another. The numbers simply represents their unique id for this presentation and has nothing to do with any technical details. On the right side of the figure the top 30 most frequent words inside of the model's initial corpus are displayed along with the words per se and their frequency seen on the x-axis. After the model has distributed topics over each given article, the

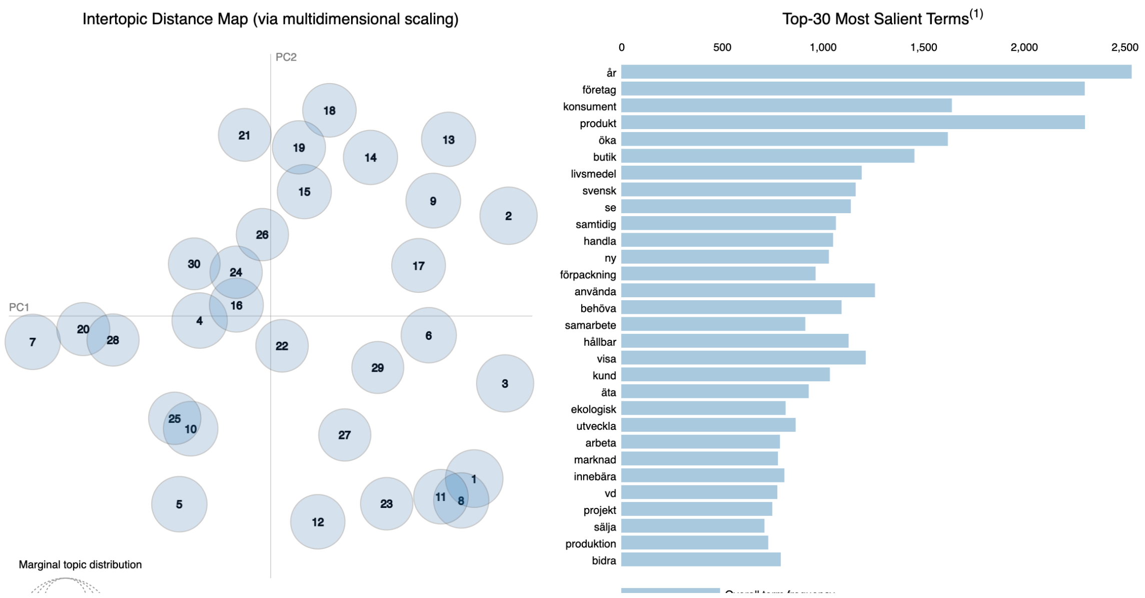


Figure 3.3.1: Visualization of the inferred topics from the LDA model.

words encapsulating the articles are added to the LDA model to be included in future computations. In this way the LDA model continuously grows and become more complex as more articles are given topic distributions, which makes the model more sustainable over time.

3.3.1 Enhancing the topics

As mentioned under chapter 2 it is difficult to evaluate the output of an LDA model. For this purpose an iteration of coherence score optimization is performed in an attempt to make the topics easier to interpret and more logical. This optimization is made through an iteration of the input variable representing the number of topics. By iterating the amount of topics input to the model a coherence score is calculated and saved for each respective iteration in order to compare coherence score along with the input number of topics. The coherence score simply projects how well the top contributing words within a topic relate to one another, which enforces the optimal consistency of topics. Since time is of the essence, together with the fact that the optimal number of topics is difficult to evaluate in real-time, the extent of the decision made for the chosen number of topics is not further examined in this thesis.

3.4 Profile articles

Once the LDA model has been created and saved, it is then used to give each article a topic distribution. An example of this distribution is shown in listing 3.1 where the numbers represent the distributions in a scale between one and zero, if multiplied by 100 it is equivalent to the percentage distribution for each topic. The listing displays a computation of 30 topics.

```
[0.014725511, 0.012386235, 0.020735823, 0.029861271, 0.017775781,
    0.01884111, 0.06323811, 0.04970541, 0.014441768, 0.01892052,
    0.02154401, 0.026241202, 0.0153603535, 0.036928583, 0.22159831,
    0.026318142, 0.015856918, 0.057931796, 0.032639958, 0.012127681,
    0.059777327, 0.012750805, 0.06431568, 0.018088603, 0.011657009,
    0.012662709, 0.011657009, 0.03770184, 0.03245206, 0.011758414]
```

Listing 3.1: Example of a computation of 30 topics distributed over an article.

Furthermore, each topic consists of a distribution of words where the top four most relevant words for the topics displayed in listing 3.1 are display in listing 3.2, for this example only ten topic word distributions are displayed.

```
0.382*"projekt" + 0.200*"bidra" + 0.194*"stärka" + 0.104*"styrelse"
0.442*"information" + 0.430*"exempel" + 0.036*"rapport" + 0.006*"europeisk"
0.152*"värld" + 0.151*"bygga" + 0.151*"utveckla" + 0.149*"teknik"
0.397*"tid" + 0.395*"odla" + 0.130*"direkt" + 0.014*"handla"
0.315*"effekt" + 0.315*"mjölk" + 0.162*"människa" + 0.160*"kalla"
0.357*"möjlighet" + 0.182*"genomföra" + 0.178*"förutsättning" + 0.178*"
    samhälle"
0.140*"använda" + 0.102*"minska" + 0.094*"jordbruk" + 0.075*"restprodukt"
0.622*"öka" + 0.151*"svår" + 0.151*"följd" + 0.011*"hög"
0.245*"butik" + 0.213*"märka" + 0.213*"matbutik" + 0.201*"ladda"
0.227*"livsmedel" + 0.225*"pris" + 0.179*"forskare" + 0.090*"undersöka"
```

Listing 3.2: Example of article topic word distributions.

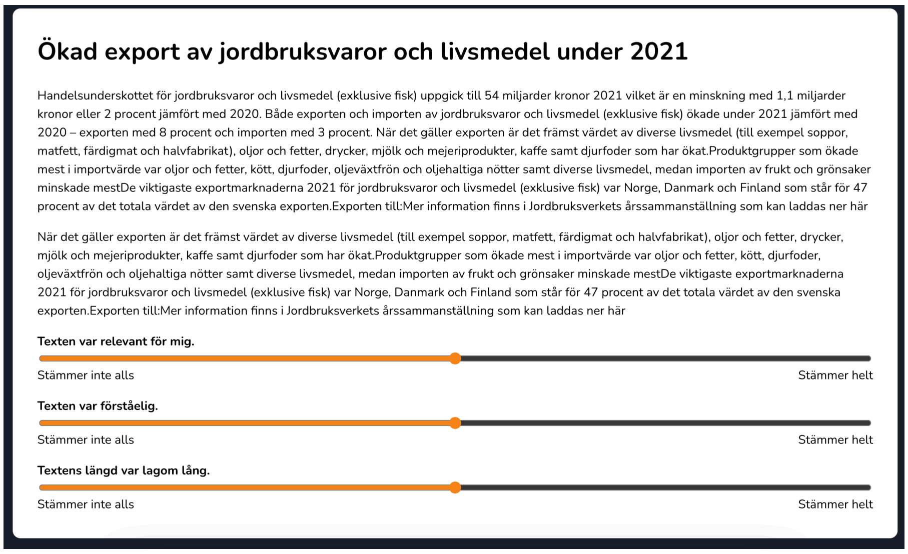
Each line represents a topic where the number displayed next to a word is the percentual contribution of the word, with the same logic as the topic distribution. Further, the topic distribution represents the article profile which per se is used to update the user profile in a later step. The user profile topic distribution is compared to each article topic distribution where the difference combined with the user evaluation relevance answer is the final factor of how to adjust the user profile.

3.5 Summarize texts

By summarizing a text or a set of documents one can easily conclude important key points inside the texts by having less words and sentences in total, making texts more efficient to read. In this way the reader can save time by easily filtering out what is important or relevant and what is not. The process of text summarization can essentially be done through either an extractive or abstractive summarization. The basic difference between the two is the way to extract important content. For this project the chosen extraction method is extractive summarization as it is considered easier to use and overall more consistent. By making use of simple TF-IDF scoring and scoring sentences according to the relevance score of the words they consist of, the top scoring sentences is chosen for the final summary.

3.6 Present to user

The evaluation for the entire project is made possible by setting up a website where users participating in the evaluation process can evaluate articles for each iteration. On this website the scraped articles for each iteration are presented along with their header and body. Users are then to choose a relevance score, as for the context of the article, a length score as to the length of the body, and an understandability score which essentially score the readability of the texts. All of these scores scale between one to five. An example of an article presentation is displayed in figure 3.6.1. Before the articles



Ökad export av jordbruksvaror och livsmedel under 2021

Handelsunderskottet för jordbruksvaror och livsmedel (exklusive fisk) uppgick till 54 miljarder kronor 2021 vilket är en minskning med 1,1 miljarder kronor eller 2 procent jämfört med 2020. Både exporten och importen av jordbruksvaror och livsmedel (exklusive fisk) ökade under 2021 jämfört med 2020 – exporten med 8 procent och importen med 3 procent. När det gäller exporten är det främst värdet av diverse livsmedel (till exempel soppor, matfett, färdigmat och halvfabrikat), oljor och fetter, drycker, mjölk och mejeriprodukter, kaffe samt djurfoder som har ökat.Produktgrupper som ökade mest i importvärde var oljor och fetter, kött, djurfoder, oljeväxtfrön och oljehaltiga nötter samt diverse livsmedel, medan importen av frukt och grönsaker minskade mestDe viktigaste exportmarknaderna 2021 för jordbruksvaror och livsmedel (exklusive fisk) var Norge, Danmark och Finland som står för 47 procent av det totala värdet av den svenska exporten.Exporten till:Mer information finns i Jordbruksverkets årssammanställning som kan laddas ner här

När det gäller exporten är det främst värdet av diverse livsmedel (till exempel soppor, matfett, färdigmat och halvfabrikat), oljor och fetter, drycker, mjölk och mejeriprodukter, kaffe samt djurfoder som har ökat.Produktgrupper som ökade mest i importvärde var oljor och fetter, kött, djurfoder, oljeväxtfrön och oljehaltiga nötter samt diverse livsmedel, medan importen av frukt och grönsaker minskade mestDe viktigaste exportmarknaderna 2021 för jordbruksvaror och livsmedel (exklusive fisk) var Norge, Danmark och Finland som står för 47 procent av det totala värdet av den svenska exporten.Exporten till:Mer information finns i Jordbruksverkets årssammanställning som kan laddas ner här

Texten var relevant för mig.

Stämmer inte alls — Stämmer helt

Texten var förståelig.

Stämmer inte alls — Stämmer helt

Textens längd var lagom lång.

Stämmer inte alls — Stämmer helt

Figure 3.6.1: Example of article presentation along with questions to answer from the website.

are presented, an algorithm predicts whether any given article will be of interest to the users based on their respective user profile compared to the articles' user profiles. The prediction of relevant answers for any user is computed through retrieving all articles for the respective iteration along with their topic profiles. Then collecting the current user profile computed from the previous iteration. The two profiles make up a one-dimensional array of distributions among the same set of topics in the same order, therefore these two arrays can be compared in similarity in an attempt to find which article will be relevant to the user in question. Through the use of angular distance one can compute angular similarities between profiles from which a similarity score is given showing the similarity between an article profile and a user profile. Then a threshold is set for this similarity score which will act as the difference between relevant and non-relevant articles. This threshold is altered according to the performance of different thresholds. From these relevancy predictions an accuracy measurement can be computed for each iteration which represents how well the algorithm predicted the relevance for each user. In this way it is possible to analyze the feedback of users by looking at the prediction accuracy value. The accuracy is a measurement between number of correctly predicted evaluation answers versus the total amount of answered evaluations, which results in a percentage. Each iteration should theoretically enhance the user profiles in order to increase the user relevancy prediction accuracy for each user and therefore eventually lead to a higher overall prediction score.

Furthermore, a basic database structure along side with a simple front end will be constructed in order for the project's evaluation to take place. In order to retrieve answers from the evaluation, a data extraction from a Uniform Resource Locator (URL) is setup where if entered, downloads all user evaluation answers for any given iteration. In order for users to identify themselves on the website they are assigned a unique id which will act as their personal login for the evaluation process.

3.7 Update user profile

As earlier mentioned, computing the user profile is made possible by first exporting the answers from any iteration of the evaluation, then retrieving the latest updated user profiles in order to update them based on the downloaded answers. The difference between each evaluated article's topic distribution and the current user profiles make up the magnitude of alteration where all differences are first summed up to later be added to an updated version of the user profiles. This way the user profiles are continuously updated as the evaluation process progresses and will theoretically tilt users toward more and more relevant content, or as for this project make the prediction algorithm predict which articles are relevant with higher accuracy. The user profile computation include processes such as normalizing distribution values, weighting article answers and reducing importance of each evaluated article the more articles are evaluated. In order to visualize the process, the sketch presented in figure 3.7.1 simplifies the process in a more educational manner. The figure represents the initial placement of a user profile in a theoretical two dimensional space of topics represented as C1 (topic 1) and C2 (topic 2). In practice the dimension is of k number of topics closer to 30, though for illustration purposes two topics are sufficient. This placement represents the initial user profile creation based on the initial batch of articles mentioned under section 3.1. The red crosses symbolize article profiles in this two dimensional room where the blue circle symbolize a user profile in the same room. Figure 3.7.1 illustrates the user profile initially placed in an arbitrary place in the room. After the first iteration is run, the user's answers are collected whereas each article

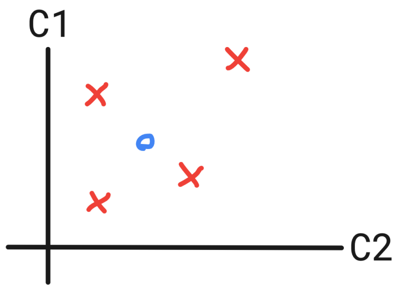


Figure 3.7.1: Visualization of user profile and article profiles in a two dimensional room.

profile difference to the user profile is calculated. A weighting and a scaling gradient is added to the difference based on the user relevance score and number of articles evaluated. This difference is then added to the user profile, which will shift the profile in the space towards a spot which better represents the user's interests as seen in figure 3.7.2. This way, the user's interest is continuously shifted towards its real position

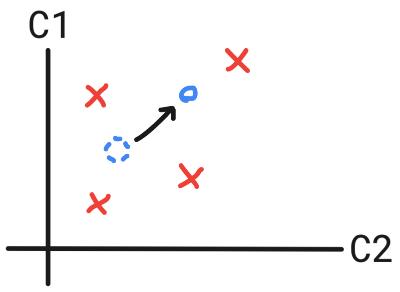


Figure 3.7.2: Visualization of updating the user profile in a two dimensional room.

inside of the space of topics where a combination of article topic distributions are of theoretical interest for the user.

3.7.1 Normalizing distribution values

The articles which are evaluated has been given topic distributions within the scale of 100% where if one sums all of the percentiles of the topic distributions up within an article topic distribution they will all add up to 100. Because of this structure, it will also be important to normalize the values of the user profile distributions in order to match the same scalar of values. This is necessary as the user profile will directly be compared to each article's individual topic distribution.

3.7.2 Answer weighting

Due to the psychological nature of human evaluations of different articles there has to be compensation to different aspects of evaluating articles. Firstly, extreme evaluation scores (as for this project, one and five) should be treated with higher impact as they should represent a more confident choice by the user. A five most likely means that the user finds the article highly interesting and should be of a greater weight than compared to a four. Secondly, the more articles a user evaluates, the lower the impact will be, since the longer an evaluation process goes on, the more accurate the user profile should become and thus should not be in need of much alteration.

3.7.3 Reducing importance progressively

As mentioned, the further into the evaluation process a user goes, the more defined the user profile should be. Taking this into regard there has to be some kind of implementation compensating the amount of time spent evaluating, or the amount of updates made for each user profile. This is measured through counting the amount of evaluated articles and applying a gradient accordingly, lowering the impact of each respective article past a certain threshold. This gradient is then of increasing impact the further into the evaluation process a user goes.

Chapter 4

Implementation

Under this section the actual implementation of the methodology introduced under chapter 3 will be presented. Following the same structure of the presented overview from the beginning of chapter 3. Here detailed examples and explanations are present showing how the implementation was carried out.

4.1 Running the pipeline

A script is run for each iteration which per se runs the four separate iteration scripts. The iteration number is first programmatically calculated by counting the past iterations' article data stored in a data folder, once complete the first script is executed including the calculated iteration number. The scripts are written in Python and are named as the following along with their functionality.

1. 1_scrape_iteration.py: Scrape data.
2. 2_compute_label_distributions.py: Profile articles & update model.
3. 3_summarize_texts.py: Summarize texts.
4. 4_predict_answers.py: Predict answers for the evaluation.

In comparison to figure 3.1.1 found in section 3.1 these scripts cover the entire pipeline displayed except for two activities which are ”present to user” and ”update user profile”. The ”present to user” activity is manually run through importing the scraped articles collected from the first script shown in the enumeration to the website database. The ”update user profile” activity is automatically run every time a user submits their evaluation answers and is handled from the website described under section 4.8.

4.2 Setup

As previously mentioned under section 3.1 the first iteration involved setting up an initial batch of articles in an attempt to boost the initial user profile creation.

These articles need to have their topic distributions of minimal similarity, optimally representing one topic each in order to help the user determine which topics are relevant to them and which are not, further described under section 3.1. The creation of the initial batch of articles is made possible by calculating an angular similarity between all article profiles collected from the initial scraping of the larger data set. The angular similarity is based on a cosine similarity which first needs to be calculated before the angular similarity. The cosine similarity represents, quoted from Selva Prabhakaran's paper on cosine similarity, "Mathematically, Cosine similarity measures the cosine of the angle between two vectors projected in a multi-dimensional space."[42]. A vector in this example is equivalent to a complete article profile. In cosine similarity, smaller angles have similar cosines, therefore further calculating the angular similarity on top of the cosine similarity will help in finding more accurate distances. The minimum angular similarity represent the articles with the maximal difference in topic distributions in comparison to one another. Listing 4.1 shows an example of an angular similarity calculation made between an article profile and a user profile.

```
cosine_distance = spatial.distance.cosine(article_dist, profile_dist)
cos_sim = 1 - cosine_distance
angular_distance = (2 * np.arccos(cos_sim)) / math.pi
angular_similarity = 1 - angular_distance
```

Listing 4.1: Example of angular similarity computation

By multiplying the arccosine of the cosine similarity divided by pi, the angular distance is retrieved. Then deducting the angular distance from one represents the distance in terms of similarity between zero to one where one represents 100% similarity. As for the initial batch of articles creation, first an array mapping the distances between all combination of data points inside the data set is created by constructing a Python Data Frame (DF) which from the official Python documentation is described as "Two-dimensional, size-mutable, potentially heterogeneous tabular data."[43]. The DF is created including the cells `from`, `to` and `distance`. The cell `from` contains an article id which represents the article profile where the distance is calculated from where the cell `to` is an article id representing the destination article profile which the former article profile is compared to. Lastly, the `distance` cell simply is the angular distance between the two article profiles. This data is then saved to further be iterated upon in order to find maximal distances between a set of `num_articles` size which is the initial article batch. In order to find the `num_articles` set of articles an arbitrary starting point inside the space of articles computed from the previous mentioned articles DF is chosen at random. The first node (article) with the largest angular distance to the initial node is selected as the first connection and added to the final article batch array as seen in listing 4.2 line 22.

```
1 num_articles = 15 - 2 # Remove initial articles
2 for i in range(0, num_articles):
3     other_points = []
4 
5     # Add all points connecting node and distance for each point added to
    final articles
```

```
6      for point in final_articles.copy():
7          all_points = df_points[df_points['from'] == point]
8          other_points.append([(t, d) for t, d in zip(all_points["to"],
    all_points["dist"])])
9
10     # Flatten list of tuples
11     all_values = [val for x in list(zip(*other_points)) for val in x]
12
13
14     # Add up all distances grouped on ids
15     summed_values = [(key, sum(num for _, num in value)) for key, value in
    itertools.groupby(all_values, lambda x: x[0])]
16
17     # Retrieve max distance from all summaries and get id
18     max_dist_point = max(summed_values, key=itemgetter(1))[0]
19     value = max(summed_values, key=itemgetter(1))[1]
20
21     # Add id to final article array
22     final_articles.append(max_dist_point)
23     test_values.append(value)
```

Listing 4.2: Selecting articles for initial batch.

This process is then repeated for `num_articles` amount of iterations. Each iteration step from selecting articles is further explained as the following:

1. For each node currently in the final article array: Add all possible node connections from current node to a temporary collective array, line 6-11.
2. Summarize all distances based on node id, line 15.
3. Retrieve the maximum distance from the summarized distances, line 18-19.
4. Add the retrieved maximum distance node id and add it to the final article batch array, line 22.
5. Repeat steps 1-4 for `num_articles` amount of times.

In this way articles with the largest summarized angular distance to a given set of nodes is iteratively selected until a batch of size equivalent to `num_articles` is selected and saved into a final initial article batch file which will be used to populate the evaluation website database as the first articles to evaluate.

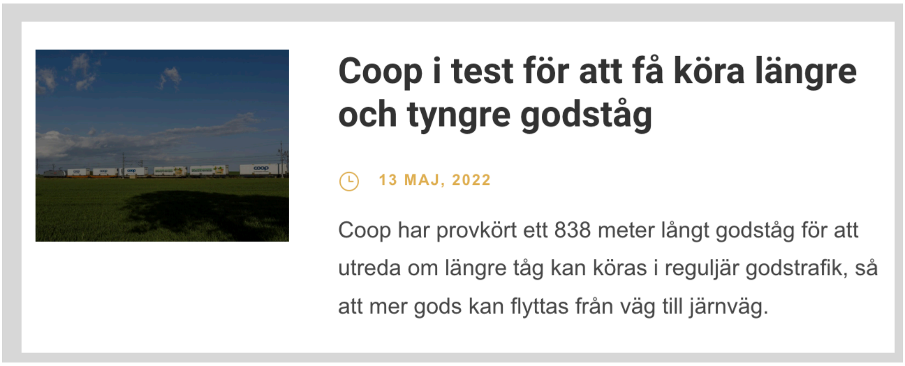
4.3 Scrape data

In order to limit the amount of articles scraped per iteration a saved date from the latest scraped iteration is automatically compared to each article's publish date. For each iteration run the date of which the scraping was completed is appended to a text file. The latest saved date to said file is fetched and used as comparison for the upcoming scraping session. During the setup iteration where the data used for the setup of the LDA model is gathered, this step is excluded. The saved date is used later in this process, if the iteration is not a setup iteration. Through the use of a Python package called Selenium a scraper could be implemented in order to extract articles from two different news sources called livsmedelifokus and livsmedelsnyheter [46]. Initially a web driver is setup using Selenium's web driver instance to simulate a web session where the first news source livsmedelsnyheter's website is opened under the URL https://www.livsmedelsnyheter.se/. As mentioned under section 3.2 the data is retrieved through targeting specific elements in the website's DOM tree. Selenium comes with several functionalities helping the user programmatically navigate a website by the use of DOM element selections. Initially before any article data can be loaded, the web session simulation first has to programmatically close all eventual popups and cookie banners initially encountered when entering the website in order to start navigating the main contents of the website. When entering livsmedelsnyheter one is presented with a popup followed by a cookie banner. The closing of these is made possible by the use of the chained web driver methods `find_element(selector)` followed by `click()` where the popup close button selector is given as input to the `find_element()` method, which here is the class `sgpb-popup-close-button-6`. As for the cookie banner close button the ID `cn-accept-cookie` is fed into the `find_element()` method. The data is now ready to be retrieved, this wraps up the process for the iterations, for the initial setup phase of the LDA model all available articles needs to be scraped. This is done through an iteration of continuously programmatically executing a click action on the ”load more” button found on the bottom of the page in order to load all of the available articles to the website.

```html
<div class="gdlr-core-load-more-wrap gdlr-core-js gdlr-core-center-align
   gdlr-core-item-pdlr" data-ajax="gdlr_core_post_ajax" data-settings="{
   TEMPORARILY-HIDDEN}" data-target="gdlr-core-blog-item-holder" data-
   target-action="append">
    <a href="#" class="gdlr-core-load-more gdlr-core-button-color" data-
   ajax-name="paged" data-ajax-value="2">Load More</a>
</div>
```

Listing 4.3: HTML code of load more button from livsmedelsnyheter.

The ”load more” button is selected in the DOM tree using the web driver and targeting the data property `data-ajax-name='paged'`, the DOM tree element can be seen in listing 4.3. After a ”load more” button click action has been executed, the process is put to sleep for five seconds to make sure the load animations has finished. A fail-safe functionality is implemented which waits until all of the animation classes `gdlr-core-animate` are invisible in the DOM tree, which means that the articles are all loaded into the website, before trying to press the ”load more” button once more. This fail-safe is implemented by the use of Selenium's `WebDriverWait` functionality which takes the web driver and a command to wait for as input. The command given as input here is a Selenium support condition called `invisibility_of_element_located` which does what the name suggests: waits until the given element is invisible, whereas `gdlr-core-animate` is given as input. However, this process is not included when scraping articles for an iteration. After either the code failed to perform another ”load more” button press during the initializing phase, or if it is a scraping iteration, the current state of the DOM tree is downloaded using an HTML parser called Beautiful Soup (BS) which is used to iterate the DOM tree programmatically and retrieve information, similar to the Selenium web driver logic. Read more about BS from their documentation [5]. All articles in the DOM tree are now to be selected and iterated to extract a URL to access more extensive information about the article, this is done by targeting the id `adplccount` with the scraper. Livsmedelsnyheter’s condensed article representation can be seen in figure 4.3.1. As seen in listing 4.4 a DOM tree is



Coop i test för att få köra längre och tyngre godståg

13 MAJ, 2022

Coop har provkört ett 838 meter långt godståg för att utreda om längre tåg kan köras i reguljär godstrafik, så att mer gods kan flyttas från väg till järnväg.

Figure 4.3.1: Condensed article from Livsmedelsnyheter.

represented by HTML code whereas the indentation represents the different levels of the tree.

```html
<div id="adplccount" class="gdlr-core-item-list gdlr-core-blog-medium clearfix gdlr-core-blog-left-thumbnail gdlr-core-item-pdlr">
 <div class="gdlr-core-blog-thumbnail-wrap clearfix">
     <div class="gdlr-core-blog-thumbnail gdlr-core-media-image  gdlr-core-opacity-on-hover gdlr-core-zoom-on-hover">
         <a href="https://www.livsmedelsnyheter.se/coop-i-test-for-att-fa-kora-langre-och-tyngre-godstag/">
             <img src="https://www.livsmedelsnyheter.se/wp-content/uploads/2022/05/cooptag.jpg" alt="" width="1024" height="778">
         </a>
     </div>
 </div>
 <div class="gdlr-core-blog-medium-content-wrapper clearfix">
     <h3 class="gdlr-core-blog-title gdlr-core-skin-title" style="font-size: 26px ;font-weight: 600 ;">
         <a href="https://www.livsmedelsnyheter.se/coop-i-test-for-att-fa-kora-langre-och-tyngre-godstag/">Coop i test för att få köra längre och tyngre godståg</a>
     </h3>
     <div class="gdlr-core-blog-info-wrapper gdlr-core-skin-divider">
         <span class="gdlr-core-blog-info gdlr-core-blog-info-font gdlr-core-skin-caption gdlr-core-blog-info-date">
             <span class="gdlr-core-head"><i class="icon_clock_alt"></i></span>
             <span class="date-article">13 maj, 2022</span>
         </span>
     </div>
```

```
        <div class="gdlr-core-blog-content">
            Coop har provkört ett 838 meter långt godståg för att utreda om längre tåg kan köras i reguljär godstrafik, så att mer gods kan flyttas från väg till järnväg.
            <div class="clear"></div>
        </div>
    </div>
</div>
```

Listing 4.4: DOM tree representation a condensed article from livsmedelsnyheter.

Seen in listing 4.4 the header is selected by targeting the classes `gdlr-core-blog-title` and `gdlr-core-skin-title` using the BS method `find(selector)`, then selecting the anchor tag with its respective hyperlink attribute enclosed by the selected target which contains the URL for the extended data of the article. Further extensive article data is then retrieved by making use of a basic Hypertext Transfer Protocol (HTTP) method called `GET`, which simply fetches the given URL and as a response returns the DOM tree, however as opposed to a Selenium instance it is not programmatically interactive. In figure 3.2.4 one can see an example of the extensive article information page. From the returned DOM tree a new BS instance, now of the extensive article information web page, is initialized and further iterated. The following elements are targeted using BS's `find(selector)` method along with their respective extracted contents.

- Title: Targeted class `infinite-single-article-title`.
- Publish date: Targeted class `date-article`.
- Full body: Targeted class `infinite-single-article-content` and then all paragraph elements encapsulated by the target class.

The DOM tree representing the title, date and body is found in listing 4.5 on line 5,6 and 9 respectively.

```
1  <div class="infinite-single-article">
2       {IMAGE}
3       <header class="infinite-single-article-head clearfix">
4          <div class="infinite-single-article-head-right">
5              <h1 class="infinite-single-article-title">Coop i test för att få köra längre och tyngre godståg</h1>
6              <span class="date-article">13 maj, 2022</span>
7          </div>
8       </header>
9       <div class="infinite-single-article-content">
10         <p><strong>{ARTICLE TEXT}</strong></p>
11         <p>{ARTICLE TEXT}</p>
12         {MORE PARAGRAPHS..}
13     </div>
14 </div>
```

Listing 4.5: DOM tree representation of an extensive article from livsmedelsnyheter.

The articles with a body length lower than 200 characters are skipped as they are considered to be of insufficient data for the project. As for each scraping for the iterations, thus not the setup iteration, the article publish date is directly compared to the date fetched mentioned in the beginning of this section. If the date is within the correct scope and the body length is of enough length, the article is added to a final Comma-Separated Values (CSV) data file. Once the iteration of livsmedelsnyheter is completed and all data is saved, the iteration of the next news source livsmedelifokus is started. The process is similar to the one recently described except for the selected DOM elements. Instead of repeating the description of the process, a simple presentation of the elements along with their targeted DOM selector is presented.

- Load more button: Target class `fusion-load-more-button`.
- Condensed articles: Target classes `fusion-post-medium`, `post`, and `status-publish`.
- Article link: Target classes `entry-title`, `fusion-post-title` and then retrieve the anchor tag encapsulated by the target classes.
- Title: Target class `title-heading-center`.
- Publish date: Target class `post-date` and `small`.
- Full body: Target classes `fusion-text`, `fusion-text-2`, `caslon` and then select all the paragraph tags encapsulated by the target classes.

These articles are appended to the same CSV files as the ones of livsmedelsnyheter. The results varied between 4-15 articles per iteration depending on the amount of published articles between the given dates from each news source.

4.3.1 Data pre-processing

As seen under section 3.2.2 there are a number of steps implemented for cleaning the data. The collected data from the previous scraping process is first loaded from the data file and stored inside of a Python DF.

Removal of punctuations and numbers

The removal of punctuations is done by iterating the data set and filtering out punctuations using regular expressions. Regular expressions are reassembling rules for specific symbols and when applied to texts will target specific pieces of the text. In order to apply the regular expression to each word, all of the words inside of an article has to be iterated. This is made possible by Python's built-in method `map()` which applied to a DF iterates each row one by one where a regular expression can be applied. The regular expression used to filter punctuations for this project is `[^\w]` which translates to ”match any character that is not an alphanumeric or underscore character”. Removing numbers is done in a similar way where the regular expression is written as `[0-9]+` where the translation is ”match one or more character in the range

0-9". The results after applying the above regular expressions are strings stripped off of punctuations and numbers.

Name entity recognizer

The NER used for this task is a pre-trained Swedish cased BERT based NER which can be downloaded and read more about on Hugging Face's website [12]. The implementation of the model is made possible by a Python module named `transformers` which allows for the use of general Natural Language Understanding (NLU) and Natural Language Generation (NLG) architectures where the method `pipeline()` is used to load the NER module. Each article text is run through the NER model where the output is collected in a two-dimensional array which is going to be handled later. Unfortunately, the proposed NER model can only handle texts of sizes less than or equal to 500 characters due to unknown reasons which were not explored for this thesis, and therefore only the first 500 characters of each text can be analyzed for name recognition. This does not pose a threat to the performance since this only improves the overall performance and does not effect any output negatively if not completely applied. The resulting output from the NER module is a tokenized version of each article along side with computed entities for each word. Each word is then iterated where a filter ignores all entities not matching the entity called `PER` which stands for Person in this context. The NER module also applies an accuracy as to how confident the entity matching is to the given word, words that has at least a 95% entity confidence and has the `PER` entity are added to an array of all unique names found for all texts. This array of retrieved names will later be used during the lemmatisation process.

Stop words

First, all words are transformed to lower case by using Python's built in method called `lower()` which is applied to each text. After the lower case transformation is completed, the removal of stop words is executed. The regular expression applied for this filtering is the following `\b(all|bara|borta|ge|ha...)\b` where the list of stop words simply is a string with the words separated by a vertical bar character `|`, which means "or". The full list of stop words are retrieved from a text file originally written by Peter M. Dahlgren[8] and can be found under appendix A.

Lemmatizing

As for the implementation of a lemmatizer, a module called Stanza is imported. Stanza offers a multilingual default pipeline which is setup to use a Swedish configuration in order to apply a lemmatizer over Swedish texts [49]. The pipeline's default processors is set to be a tokenizer, part of speech tagger, lemmatizer and a depparser. This project is only going to make use of the tokenizer and the lemmatizer but since the other two processors do not interfere with the ones to be used, they are left as they are. Once the Stanza pipeline is initiated it is applied to the article texts. Each article is iterated and run through the pipeline, which in term returns a tokenized and lemmatized format of the text. Each word is then matched with the names from the NER computation where, as earlier mentioned under section 4.3.1, if matching any name from the array, the word is removed. Otherwise, the lemmatized version of the word is appended to the final output of the lemmatized article text.

Bigrams

Bigrams are constructed from analyzing all of the article bodies by the use of a Gensim model called `Phrases` which identifies common sentences throughout a corpus [15]. The model setup can be seen in listing 4.6.

```python
Phrases(df_copy['parsed_content'], min_count=5, threshold=10)
```

Listing 4.6: Python code of the setup of a `Phrases` model.

`df_copy` represents the corpus, `min_count` is the minimum word count to include as a common sentence and lastly the threshold is a simple arbitrary scoring number which is not put much thought into. From applying this model, bigrams are identified and added to the final version of the processed data which completes the pre-processing steps of the article data.

4.4 Update model

The data set has now been processed and is ready to be handed to an LDA model to compute its topic distributions. In order to create the initial LDA model, there first has to be a dictionary which will be put into the LDA model. A dictionary is created based on all of the article texts from the processed data set mentioned under section 4.3.1. By the use of a module called Gensim which in turn has a class called `Dictionary` the dictionary could easily be initialized as can be seen in listing 4.7

```python
dict = Dictionary(df['content'])
```

Listing 4.7: Initialization of a Gensim dictionary.

`df['content']` is the DF of the article data where the `content` is a cell in the DF representing the article bodies. The dictionary is then filtered to remove extreme words where words frequently occurring or not occurring frequent enough are removed to further improve the quality of the data. This is done by the use of a `Dictionary` class function called `filter_extremes()` which automatically handles the process of filtering out extreme words. Once filtered, the dictionary is saved for future iteration computation in a text file. After the dictionary is created a corpus based on the dictionary is constructed using another Gensim Dictionary class function called `doc2bow()` which converts the input into a `bag-of-words` format. Lastly, both the corpus and dictionary is added as input to an LDA model also provided by the Gensim module which can be seen in listing 4.8

```
lda_model = LdaModel(
    corpus = corpus,
    id2word = dict,
    minimum_probability = 0.0,
    num_topics = 30,
    random_state = 100,
    alpha = 50/num_topics,
    eta = 200/len(dict),
    chunksize = 500,
    passes = 100,
    per_word_topics=True
)
```

Listing 4.8: Initialization of a Gensim LDA model.

The first two parameters represent the corpus respectively the dictionary which is the scraped data. `minimum_probability` represent the minimum probability to include when generating topic distributions. If this number is not set to zero, the resulting amount of computed topics may be less than of the maximal amount of topics possible to obtain, which further causes inconsistencies. `random_state` is set in order to be able to reproduce the same computation over several iterations. The `alpha` and `eta` values are both important to find a good balance between a low and high value of each respective parameter. Hence, the following computation of 50/{number of topics} is suggested by Thomas L. Griffiths and Mark Steyvers in their study regarding LDA. However, the `eta` value is based on an experimental number found to be fit [16]. The `chunksize` and `passes` refers to the iterations of the LDA model computations, how big chunks of the data set should be tested into how many passes. Lastly, the `per_word_topic` refers to the output type of the data where the set value simply is easier to work with for this project. After the LDA model has processed the given input variables, included the dictionary data and corpus data, the model is saved for further use. Through each iteration the previously constructed LDA model is loaded as well as the saved dictionary which forms a basis for the topic distributions. Once the topic distribution has been computed for the articles, the LDA model is updated to include the new article contents and then once again gets saved. This way the model is continuously updated through every iteration. The update is made possible by Gensim's default LDA model method `update(corpus)` which simply is the corpus data scraped for the current iteration.

4.5 Profile articles

All articles from the processed article data set are given their respective topic distributions taken from the computation of the constructed LDA model from section 4.4. Article profiles are created based on the LDA model computation where the output from the model is topic distributions for all articles, seen in listing 4.9.

```
for i in df_copy.index:
    df_copy.at[i, 'label_dist'] = str([j[1] for j in lda_model[new_corpus][
   i][0]])
```

Listing 4.9: Python code of creating article profiles using LDA.

The code loops all articles inside of the DF named `df_copy` and assigns the computed topic distributions computed by the LDA model to a new column called `label_dist` inside of the DF.

4.6 Update user profile

The first action to take when calculating a user profile is to either fetch an existing profile or, if no existing profile is found, create a new standardized profile. The standardized profile is created by filling an array with a percentage reflecting an equal distribution of every topic as seen in listing 4.10.

```
if (! isset($user_profile))
    $user_profile = array_fill(0, $num_topics, (1/$num_topics));
else
    $user_profile = json_decode($user_profile->profile);
```

Listing 4.10: Python code of the creation of a user profile.

After the user profile is fetched, each article evaluated by the user in question is iterated in order to weigh each respective article distribution according to the user's evaluation data. All evaluations are first iterated before any adjustment is made to the user profile in order to ignore the order of which the articles are iterated through. The weighting of each article topic distribution is done through iterating the articles topic distribution values one by one in order to simply subtract the user profile topic distribution value which corresponds to the current value with the current value, see listing 4.11 line 5.

```
1  $total_diff = [];
2  foreach($article_dist as $j => $single_topic_dist) {
3
4      // Get abs. difference between profiles
5      $diff = $single_topic_dist - $user_profile[$j];
6      $total_diff[] = $diff;
7
8      // Decrease impact of articles the more you answer
9      // Roughly every other iteration (30 articles)
10     $gradient = $num_answers >= 30 ? (1 / ($num_answers/$num_topics)) : 1;
11
12     // Add weighted difference to user profile. This weight decides the
       impact of a single article.
13     $scale = [-0.75, -0.25, 0, 0.25, 0.75];
14
15     // Add weighted difference (Is deducted if negative)
16     $sum_weighted_diff[$j] += $diff * $scale[$article->pivot->relevance -
       1] * $gradient;
17 }
```

Listing 4.11: PHP code of the iteration of article topic distribution.

A gradient is then calculated based on the amount of articles evaluated in order to make sure that the further the user progresses through the evaluation process, the smaller steps the user profile will compute for each iteration of articles. The gradient is programmed to apply when the total number of evaluations for a specific user are above 30. The final difference then gets decrementally lower the more evaluations are completed, see listing 4.11 line 10. Furthermore, the weighting factor is one value of a set between -0.75, -0.25, 0, 0.25 and 0.75 and depends on the user relevance score as for the evaluation, the higher the relevance the larger the value. The reason behind the non-linear scaling is that of the extreme answers of one and five are considered to be more reliant than of those being closer to the middle of the scale where the user is considered to be more uncertain of the answer. Thus technical implementation of the total difference with applied gradient and weightage comes down to line 16 seen in listing 4.11. This line translates to: The difference is calculated by multiplying the above mentioned scaling of the answer with the original difference and the calculated gradient. After all of the differences has been calculated for every article topic distribution and summarized to a final value, the values has to be normalized in order to properly fit the user profile. Firstly, the values are scaled down to be values between one and zero by iterating each topic value and perform the following computation seen in listing 4.12 line 2.

```
1 $max = max($user_profile);
2 $min = min($user_profile);
3 foreach($user_profile as $j => $val) {
4     $user_profile[$j] = ($val - $min) / ($max - $min);
5 }
```

Listing 4.12: PHP code of normalizing a user profile topic distribution.

The code in listing 4.12 translates to: adjust the current topic distribution value to be the same value deducting the lowest topic value found, then dividing by the maximum topic value found deducted by the minimum topic value found. The values are now scaled down to be values between one and zero, though they do not summarize to be a total of 100%, this is done through a new computation found in listing 4.13.

```
$sum = array_sum($user_profile);
foreach($user_profile as $j => $val) {
    $user_profile[$j] = $val / $sum;
}
```

Listing 4.13: PHP code of adjusting profile values up to a total of 100%.

The total summarized value of the current user profile is calculated, then each value is iterated and adjusted to be a result of the division between the current value and the total summarized value. The effect of this computation is all of the values adding up to a maximum of one or in other words, all topics percentiles adding up to 100%. Lastly, the updated user profile is saved and used for the coming iteration.

4.7 Summarize texts

Once the data is collected and processed, the next step is to summarize the texts. The summarization is setup as an extractive summarizer using text tokenizer functions imported from a Natural Language Toolkit (NLTK) tokenizer module called `word_tokenize()` and `sent_tokenize()` [37]. A word count computation is performed by manually counting the word frequencies from one article at a time and appending the count to an array which essentially represents the articles in a `bag-of-words` format, see listing 4.14.

```python
words = word_tokenize(df_copy.loc[row, 'content'], language="swedish")
    freqTable = dict()
    for word in words:
        word = word.lower()
        if word in stopwords:
            continue

        if word in freqTable:
            freqTable[word] += 1
        else:
            freqTable[word] = 1
```

Listing 4.14: Python code of calculating word frequency used for text summarization.

The sentence tokenizer is further executed in order to keep track of which sentence contain which words, in addition to the previous word tokenizer where essentially the sentence structure is lost. The sentences collected from the sentence tokenizer are scored based on the sentence word frequencies calculated in listing 4.14, the more frequent the words inside a sentence are, the higher the score for the sentence. In listing 4.15 one can see the computation of the sentence scoring using the word frequencies as a basis.

```python
sentences = sent_tokenize(df_copy.loc[row, 'content'], language="swedish")
    sentenceValue = dict()
    for sentence in sentences:
        for word, freq in freqTable.items():
            if word in sentence.lower():
                if sentence in sentenceValue:
                    sentenceValue[sentence] += freq
                else:
                    sentenceValue[sentence] = freq
```

Listing 4.15: Python code of calculating sentence scores used for text summarization.

Since only one article is iterated at a time, the word frequency and sentence scoring will be based on the article's scope of words and contextual meaning and thus provide a content rich summary for each individual article. Lastly, all of the scored sentences in an article are compared to a given threshold which is the basis for determining if a sentence is included in the final summary seen in listing 4.16.

```python
sumValues = 0
```

```
2  for sentence in sentenceValue:
3      sumValues += sentenceValue[sentence]
4
5  average = int(sumValues / len(sentenceValue))
6  summary = ''
7  tmp_sentence_score_threshold = sentence_score_threshold
8  while summary == '':
9      for sentence in sentences:
10         if (sentence in sentenceValue) and (sentenceValue[sentence] > (
       tmp_sentence_score_threshold * average)):
11             summary += " " + sentence
12
13     tmp_sentence_score_threshold -= 0.05 # Lower with 0.05 because we want
       longer summaries
```

Listing 4.16: Python code of the computation of a sentence score threshold.

Line 1-3 in listing 4.16 calculates all of the sentence scores and summarize them in order to find the average sentence score, found on line 5, which further is used for comparing to the final threshold. Line 8-13 is the iteration where the sentence is decided if it is going to be added or not, the original logic is to add a sentence if the sentence value is above the average value multiplied by a predetermined threshold value. The threshold value is set to be 1.2 which roughly translates to a score 20% higher than the average value. A fail-safe loop wraps the entire iteration checking if no sentence in the article is above the threshold. Then the threshold is decrementally lowered by 0.05 until at least one sentence is added. The threshold is then reset for the next summary computation, and so it continues. Each article text summary is then saved to the same article data set in a new column called `sum_content`.

4.8 Present to user

The construction of the evaluation website is made possible by the use of a PHP platform called Laravel [22]. The website is hosted using Laravel Forge[26] which takes care of the server distribution and update as well as database hosting by the use of Amazon Web Services (AWS) [4].

4.8.1 Login

When entering the website one is first prompted to login in order to start their individual article evaluation session. Each user is handed a unique login token connected to their respective account which their user profile and article evaluations are connected to. The login code can be entered from the login page which will, if correct, redirect the user to the home page consisting of the articles to be evaluated. Otherwise the website will simply prompt the user that the login code is invalid. The login session is setup by the use of a URL header parameter representing the token which is an implementation made for efficiency and not security. The user token is visible in the URL and if removed from the URL, it will redirect the user back to the login page. Users can not enter any website without having a valid token present in the header as the article page is protected by a middleware called `EnsureTokenIsValid` where a token validation is performed prior to entering the site, see listing 4.17.

```php
if (! in_array($request->token, User::all()->pluck('token')->toArray())) {
    return redirect('login');
}
```

Listing 4.17: PHP code of the EnsureTokenIsValid middleware.

The code simply checks if the passed URL parameter called `token` does not exist among any user stored in the database and in that case redirects the user back to the login page, otherwise the user is redirected to the home page. Other URLs setup for the website are protected by a `ValidateAdminIP` middleware which only allows specific Internet Protocol (IP) addresses to enter, these pages can not be reached by any user if they do not originate from specific IP addresses, see listing 4.18.

```php
$ip_addresses = [
    {IP ADDRESSES}
];

if(! in_array($request->ip(), $ip_addresses)) {
    abort(403, 'Tough luck buddy!');
}
```

Listing 4.18: PHP code of the ValidateAdminIP middleware.

4.8.2 Home page

Once logged in with a correct user token, the user is redirected to the home page. Here the user is presented with articles from all iterations which has not yet been evaluated, optimally there should only be one set of articles from one iteration if the user has answered articles for every past iteration on time. The articles are fetched via a Laravel model called `Article` along with eloquent syntax code which behind the scenes construct a Structured Query Language (SQL) query based on the given arguments [25]. As can be seen in listing 4.19 line 2, the `Article` model is setup to use the query condition `whereDoesntHave` along with a nested `where` condition, which translates to: find all articles where no users with the current user token are attached.

```php
1 // Retrieve user's non-answered articles
2 $articles = Article::whereDoesntHave('users', function($q) use ($request) {
3     $q->where('user_id', User::where('token', $request->token)->first()->id
     );
4 })->get();
5
6 $user = User::where('token', $request->token)->first();
7 $user->last_accessed = Carbon::now();
8 $user->save();
9
```

```
return view('welcome')->with('articles', $articles);
```

Listing 4.19: PHP code of the home page data retrieval.

Lastly, through line 6-8 a ”last accessed” date is saved to the current user in order to simply keep track of user activity on the website. Each article presented on the website has three different questions to evaluate where each question is a scalar of five steps ranging from zero ”completely disagree” to five ”completely agree”. The first question is about relevance, second is text length and third is understandability. The input gets stored as a number and is then used as a weight towards the update of the user profile, as mentioned under section 4.6. Once the user has chosen a scale for every presented article, the user can press submit where the answers are submitted to the profile computation algorithm discussed under section 4.6 using a simple Asynchronous JavaScript and XML (AJAX) `post` query seen in listing 4.20 [2].

```
$.post('/submit-answer', {"data": articleData, "token": token}).then(() =>
    {
    window.location.reload();
});
```

Listing 4.20: Javascript code of the action after pressing submit from the home page.

The `articleData` variable is all the answers from the home page and the token is the current user token. After the request is complete the website is reloaded which should then display a feedback message thanking the user for its contribution as no articles are left to be evaluated.

4.8.3 Predictions

Predictions can be made as to the users' evaluations based on their computed user profiles from previous iterations. Each user profile is collected from an exported CSV file from the evaluation website database, namely the URL ”/export/answers?iteration=X” which conveniently returns a CSV file with all the evaluations from all users for the given iteration. Each user is then iterated one at a time in order to compute predictions for the coming iteration. The prediction algorithm is displayed in listing 4.21.

```
cosine_distance = spatial.distance.cosine(article_dist, profile_dist)
cos_sim = 1 - cosine_distance
angular_distance = (2 * np.arccos(cos_sim)) / math.pi
angular_similarity = 1 - angular_distance
prediction = angular_similarity >= threshold
```

Listing 4.21: Python code of the relevancy prediction for an arbitrary user and article.

The `spatial.distance.cosine()` method calculates the cosine distance between two arrays which here are arrays of topic distributions for both the article profile and the user profile. Deducting the cosine distance from one represents the output in a form of cosine similarity which then is used for the angular distance calculation. The calculation of the angular distance can be seen in listing 4.21 on line 3. As well as for the cosine distance, the angular distance is deducted from one where the result is the angular similarity. This similarity score is then compared to a prediction threshold value which makes up the foundation of the prediction algorithm. The final result is a binary answer of one or zero whether the user will find the article relevant or not. The predictions are saved for each user in a separate file. After all of the predictions are computed for the given iteration and the users has evaluated all articles, an accuracy score can be calculated as to how many correct predictions the algorithm made based on the users' evaluations. All user predictions are iterated and compared to their corresponding evaluation downloaded from the previously mentioned URL "export/answers?iteration=X" in order to determine how many correct predictions the algorithm made. A correct prediction is both if the user find an article interesting or if the user has not found the article interesting. The summary of all correct predictions made can be seen in listing 4.22.

```python
correct_pred = 0
answers = df_user_answers["Rel"][0]
for idx, prediction in enumerate(article_predictions["relevant"]):
    prediction = prediction >= predict_rel_threshold
    answer = answers[idx] >= answer_rel_threshold
    if prediction == answer:
        correct_pred += 1
```

Listing 4.22: Python code of the accuracy calculation summary.

The evaluation answer has a relevance threshold of four taken from the scalar of one to five through the evaluation process. A four or higher is considered to be of relevance. If the prediction matches the evaluation made, regardless of it being positive or false, the `correct_pred` variable is incremented by one which keep track of all correct predictions. `correct_pred` is then divided by the total amount of predictions in order to retrieve a final accuracy score for the prediction model from the iteration in question. The score is saved for future analysis.

4.8.4 Explore

In order to analyze data in real-time in an easy and controlled way, two explorative pages are setup, protected behind the `ValidateAdminIP` middleware explained under section 4.8.1. The two URLs are "/explore" and "/explore/articles". For each URL one can provide a URL parameter called `iteration` which filters results on the URL web page based on the given iteration parameter. If no iteration parameter is found then results from all iterations are presented, see listing 4.23.

```php
$articles = isset($request->iteration) ? $user->answers()->get()->where('
    iteration_id', $request->iteration) : $user->answers;
```

Listing 4.23: PHP code of the filtering of data based on an iteration parameter.

Under the URL "/explore" one can analyze each user's current user profile distribution visualized on a graph as well as average length scoring and average understandability scoring. The graph is constructed by the use of a Laravel package class called `LabelDistChart` provided by `Laravel Charts`[24]. The graph setup can be seen in listing 4.24.

```
$labelDistChart = new LabelDistChart;
$labelDistChart->labels(array_keys($profile));
$labelDistChart->options([
        'scales' => [
            'yAxes' => [
                [
                    'ticks' => [
                        'max' => 1.0
                    ],
                ],
            ],
        ],
    ]);
$labelDistChart->dataset('Labels by distribution', 'bar', $profile);
```

Listing 4.24: PHP code of the setup of user profile graph visualization.

Under the URL "/explore/articles" one can analyze the articles topic distributions in the same way using the same graph implementation as mentioned in listing 4.24.

4.8.5 Database

The database setup for the website contains four tables called `article_user`, `articles`, `user_profiles` and `users`. Under this section all of the tables are described along with their respective data columns.

`article_user`: All the user answers stored as a `one-to-many` relationship. The columns are `article_id`, `user_id`, `relevance`, `understandability`, `length` and `difference`. The article and user id are foreign keys connecting article answers to users. `Relevance`, `understandability` and `length` are the scoring of each article as input by the user and the `difference` column is the computed change made to the user profile based on the profile computation saved for safety measurements.

`articles`: All articles from each iteration are stored here. The columns are `id`, `publish_date`, `title`, `content`, `sum_content`, `label_dist` and `iteration_id`. The `publish_date`, `title` and `content` are simply information for each article. The `sum_content` is the final summary of the original content of the article once processed through the summarization script. `label_dist` is the label distribution computed by the LDA model and finally the `iteration_id` is which iteration the given article is related to in order to filter out and analyze articles for each iteration.

`users`: The `users` table is the table where all users are stored, the columns here are id, token, email, last_accessed, and created_at and updated_at. The token is the unique token used for the user to login with which is manually created for each user from a random number between 1 and 99999. last_accessed is a date stored every time the user enters the article home page in order to somewhat track users.

user_profiles: Finally, all computed user profiles are stored here with their belonging iteration. The columns are id, created/updated_at, user_id, token, profile and iteration_id. The user_id is the foreign key relating the profile to a user and the token simply is the user token redundantly saved for easier access. profile simply is the user profile computed from the previously mentioned user profile update scripts and lastly, the iteration_id is the current iteration the user profile has been computed from.

4.8.6 Notifying users

Lastly, a Laravel command is constructed in order to conveniently automatically notify all users of newly imported articles. Each user's email is stored along with the user in the database where the email is used for the notification. The code for the user notification command is displayed in listing 4.25.

```
1  $iteration = Article::orderBy('iteration_id', 'desc')->first()->
       iteration_id;
2
3  $notAnsweredUsers = User::whereDoesntHave('answers', function($q) use (
       $iteration) {
4      $q->where('iteration_id', $iteration);
5  })->get();
6
7  foreach($notAnsweredUsers as $user) {
8      if(isset($user->email) && !$user->email == '') {
9          $details = [
10             'title' => 'Det har nu anlänt nya artiklar på hemsidan.',
11             'body' => 'Din kod är: '.$user->token,
12             'link' => env('APP_URL').'?token='.$user->token
13         ];
14
15         Mail::to($user->email)->send(new NotifyMail($details));
16
17         dump("Notified ".$user->email);
18     }
19 }
```

Listing 4.25: PHP code of the notify users command.

First, all users which has articles not yet answered are fetched seen in line 3-5 and then iterated through seen on line 7 where each user is sent an email using Laravel's Mail facade, line 15 [27]. The email is sent from a Gmail account specifically created for this purpose, which is used as a dummy account. The command is executed using Laravel artisan[23] through the following line entered to a terminal: php artisan `survey-app:notify-users`.

Final layout

After the implementation and some alterations to the evaluation website the final layout is displayed in this coming section. First, the login page is shown in figure 4.8.1. Once logged in, the homepage design ended up as shown in figure 4.8.2. The smaller

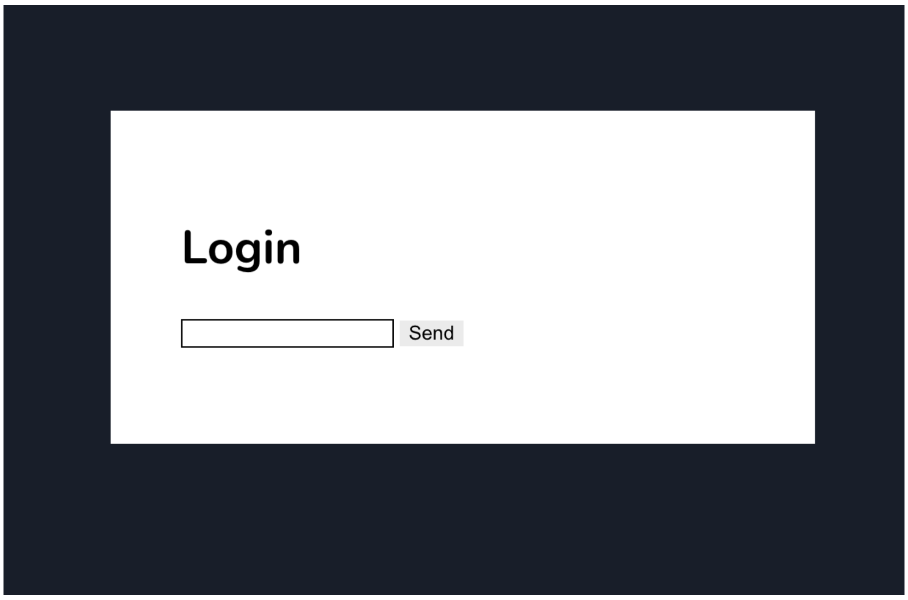
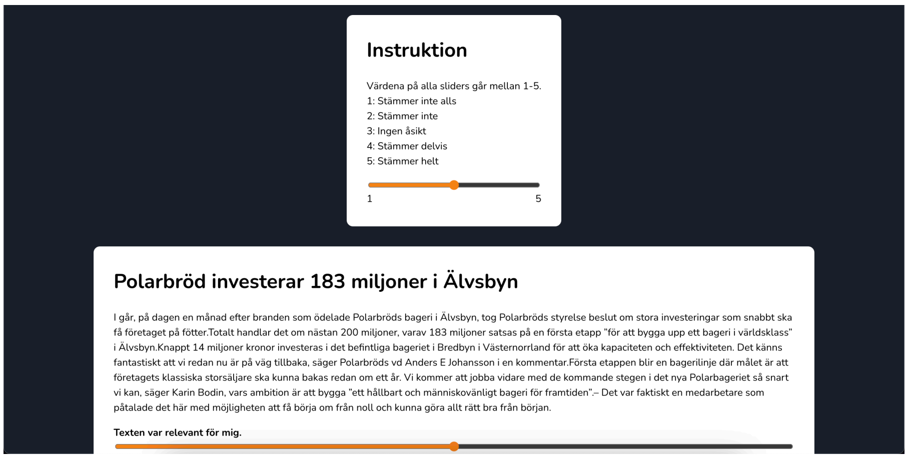


Figure 4.8.1: Login page.

box seen on the top is an instruction box helping users faster understand what they see and how to perform the evaluation.



Instruktion

Värdena på alla sliders går mellan 1-5.
1: Stämmer inte alls
2: Stämmer inte
3: Ingen åsikt
4: Stämmer delvis
5: Stämmer helt

Polarbröd investerar 183 miljoner i Älvsbyn

I går, på dagen en månad efter branden som ödelade Polarbröds bageri i Älvsbyn, tog Polarbröds styrelse beslut om stora investeringar som snabbt ska få företaget på fötter.Totalt handlar det om nästan 200 miljoner, varav 183 miljoner satsas på en första etapp ”för att bygga upp ett bageri i världsklass” i Älvsbyn.Knappt 14 miljoner kronor investeras i det befintliga bageriet i Bredbyn i Västernorrland för att öka kapaciteten och effektiviteten. Det känns fantastiskt att vi redan nu är på väg tillbaka, säger Polarbröds vd Anders E Johansson i en kommentar.Första etappen blir en bagerilinje där målet är att företagets klassiska storsäljare ska kunna bakas redan om ett år. Vi kommer att jobba vidare med de kommande stegen i det nya Polarbageriet så snart vi kan, säger Karin Bodin, vars ambition är att bygga ”ett hållbart och människovänligt bageri för framtiden”.– Det var faktiskt en medarbetare som påtalade det här med möjligheten att få börja om från noll och kunna göra allt rätt bra från början.

Texten var relevant för mig.

Figure 4.8.2: Home page.

After the evaluations are submitted by a user a feedback message is displayed, if a user tries to enter the web page once more before any new articles have been imported, the same message is displayed. The explore user profile page is a simple page only displaying boxes as shown in figure 5.2.3, thus there is no need to present it here. Lastly, the explore article page is a replica of the explore user profile page, however it included a summarized profile for all articles on the top of the page for further analysis of the article profiles. This page is shown in figure 4.8.3.

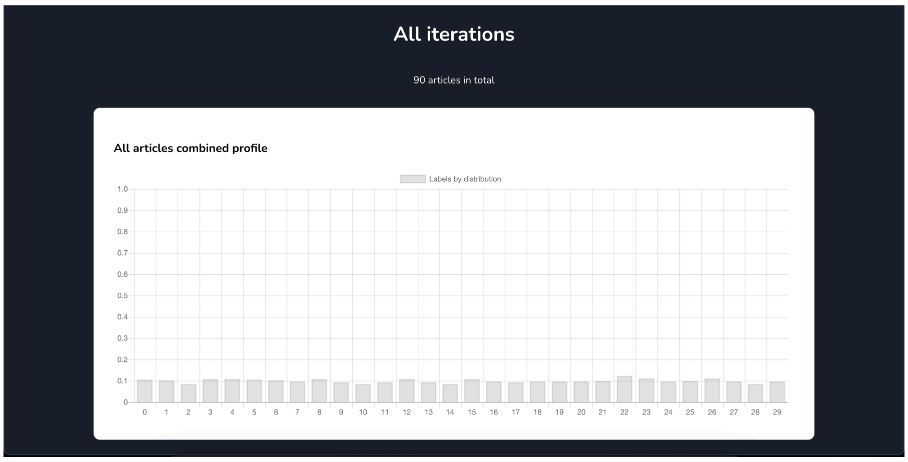


Figure 4.8.3: All articles exploration page.

Chapter 5

Evaluation and Results

Under this section, results from the project is presented. The final model used for the computation of topics, presentation of the topics along with examples of user profiles, article profiles and a summarized article are here presented. The study protocol is explained. Finally, the project results are presented as to the predictions and progression of the user profiles.

5.1 Study protocol

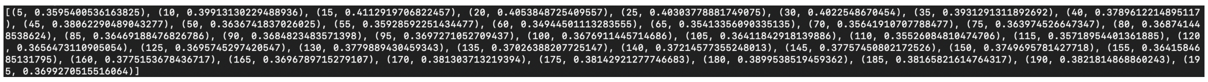
Under this section the evaluation study protocol is presented. The study is setup to run for ten iterations, updating the website with new articles every Monday and Thursday followed by a notification in an attempt to activate the users. The evaluation is setup in order to evaluate the performance of the prediction model as well as the text summarizer where users will be able to score both relevancy of articles presented and length/understandability of the texts. Before the actual evaluation begins, a test phase is setup where a number of testers tried out the evaluation website in order to find some bugs and errors to be eliminated before the real evaluation could begin. This process is only run for one week as the consequence of a live bug is not of great impact since the evaluation is run in a controlled closed environment. Furthermore, the real evaluation consist of 20 employees from Elvenite at the time participating all somewhat related to the domain area the articles are scraped from since their workplace specializes in the area. For each iteration, each participant entered the website and scored a number of articles, depending on the amount scraped from the news sources. The score ratings are between one and five and as mentioned concerned the article length, understandability and relevance. The evaluations are submitted by each respective user and saved to a database where it could later be analyzed. Prior to this, the evaluations are used in order to update each specific user's profile. The data collected is analyzed through a prediction algorithm which compares each user's evaluation submissions based on the pre-computed prediction of the relevancy score related to their respective user profiles in order to compute an accuracy for the prediction model. After iteration ten is completed, the evaluation will come to a stop and the data is collected, saved and further analyzed.

5.2 Evaluation

Here a presentation of the final outcomes from the implementation is presented. First, a presentation of the final LDA model, second, the final topics, and last, examples of an actual user profile, article profile and text summarization taken from the evaluation is presented.

5.2.1 LDA model

The data set ended up being around 6200 articles scraped from the two news sources mentioned under section 4.3 which further was used to train the original LDA model according to section 4.3. The final LDA model setup can be seen in code 4.8 under section 4.4. The final number of topics chosen for the LDA model is 30 after running a coherence optimization script in order to find the optimal number of topics reflecting the highest coherence score combined with a tangible number of topics. The optimization results can be seen in figure 5.2.1. The highest coherence number found



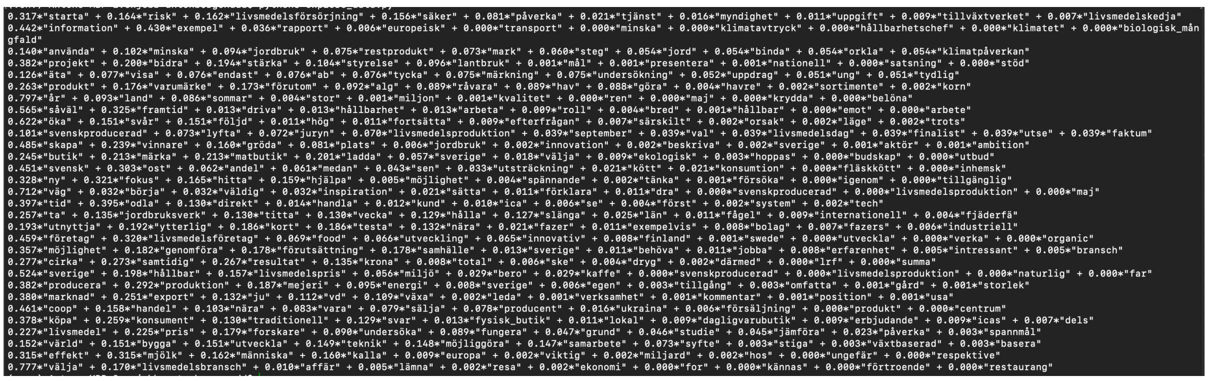
```
[(5, 0.3595400536163825), (10, 0.39913130229488936), (15, 0.4112919706822457), (20, 0.4053848725409557), (25, 0.40303778881749075), (30, 0.4022548670454), (35, 0.3931291311892692), (40, 0.3789612214895117
), (45, 0.38062290489043277), (50, 0.3636741837026025), (55, 0.35928592251434477), (60, 0.34944501113283555), (65, 0.35413356090335135), (70, 0.35641910707788477), (75, 0.363974526647347), (80, 0.36874144
8538624), (85, 0.36469188476826786), (90, 0.3684823483571398), (95, 0.3697271052709437), (100, 0.36769114457146 86), (105, 0.36411842918139886), (110, 0.35526084810474706), (115, 0.35718954401361885), (120
, 0.3656473110905054), (125, 0.3695745297420547), (130, 0.3779889430459343), (135, 0.37026388207725147), (140, 0.37214577355248013), (145, 0.37757450802172526), (150, 0.3749695781427718), (155, 0.36415846
85131795), (160, 0.3775153678436717), (165, 0.3696789715279107), (170, 0.381303713219394), (175, 0.38142921277746683), (180, 0.3899538519459362), (185, 0.38165821614764317), (190, 0.3821814868860243), (19
5, 0.3699270515516064)]
```

Figure 5.2.1: Coherence scores after altering different number of topics put into the LDA model.

is when using 15 topics, however more topics is needed in order to properly compute a solid user profile. Therefore, 30 was chosen as number of topics as it is a high enough number of coherence score with a difference of only 0.9 and a high enough number of topics to properly compute a user profile.

5.2.2 Topics

From the final LDA model the computed topics used for the topic distribution among the articles are presented in figure 5.2.2. The figure displays the top ten most contributing words to each respective topic along with the percentual contribution next to the word.



```
0.317*"starta" + 0.164*"risk" + 0.162*"livsmedelsförsörjning" + 0.156*"säker" + 0.081*"påverka" + 0.021*"tjänst" + 0.016*"myndighet" + 0.011*"uppgift" + 0.009*"tillväxtverket" + 0.007*"livsmedelskedja"
0.442*"information" + 0.430*"exempel" + 0.036*"rapport" + 0.006*"europeisk" + 0.000*"transport" + 0.000*"minska" + 0.000*"klimatavtryck" + 0.000*"hållbarhetschef" + 0.000*"klimatet" + 0.000*"biologisk_mån
gfald"
0.140*"använda" + 0.102*"minska" + 0.094*"jordbruk" + 0.075*"restprodukt" + 0.073*"mark" + 0.060*"steg" + 0.054*"jord" + 0.054*"binda" + 0.054*"orkla" + 0.054*"klimatpåverkan"
0.382*"projekt" + 0.200*"bidra" + 0.194*"stärka" + 0.104*"styrelse" + 0.096*"lantbruk" + 0.001*"mål" + 0.001*"presentera" + 0.001*"nationell" + 0.000*"satsning" + 0.000*"stöd"
0.126*"äta" + 0.077*"visa" + 0.076*"endast" + 0.076*"ab" + 0.076*"tycka" + 0.075*"märkning" + 0.075*"undersökning" + 0.052*"uppdrag" + 0.051*"ung" + 0.051*"tydlig"
0.263*"produkt" + 0.176*"varumärke" + 0.173*"förutom" + 0.092*"alg" + 0.089*"råvara" + 0.089*"hav" + 0.088*"göra" + 0.004*"havre" + 0.002*"sortimente" + 0.002*"korn"
0.797*"år" + 0.093*"land" + 0.086*"sommar" + 0.004*"stor" + 0.001*"miljon" + 0.001*"kvalitet" + 0.000*"ren" + 0.000*"maj" + 0.000*"krydda" + 0.000*"belöna"
0.565*"såväl" + 0.325*"framtid" + 0.013*"driva" + 0.013*"hållbarhet" + 0.013*"arbeta" + 0.009*"roll" + 0.004*"bred" + 0.001*"hållbar" + 0.000*"emot" + 0.000*"arbete"
0.622*"öka" + 0.151*"svår" + 0.151*"följd" + 0.011*"hög" + 0.011*"fortsätta" + 0.009*"efterfrågan" + 0.007*"särskilt" + 0.002*"orsak" + 0.002*"läge" + 0.002*"trots"
0.101*"svenskproducerad" + 0.073*"lyfta" + 0.072*"juryn" + 0.070*"livsmedelsproduktion" + 0.039*"september" + 0.039*"val" + 0.039*"livsmedelsdag" + 0.039*"finalist" + 0.039*"utse" + 0.039*"faktum"
0.485*"skapa" + 0.239*"vinnare" + 0.160*"gröda" + 0.081*"plats" + 0.006*"jordbruk" + 0.002*"innovation" + 0.002*"beskriva" + 0.002*"sverige" + 0.001*"aktör" + 0.001*"ambition"
0.245*"butik" + 0.213*"märka" + 0.213*"matbutik" + 0.201*"ladda" + 0.057*"sverige" + 0.018*"välja" + 0.009*"ekologisk" + 0.003*"hoppas" + 0.000*"budskap" + 0.000*"utbud"
0.451*"svensk" + 0.303*"ost" + 0.062*"andel" + 0.061*"medan" + 0.043*"sen" + 0.033*"utsträckning" + 0.021*"kött" + 0.021*"konsumtion" + 0.000*"fläskkött" + 0.000*"inhemsk"
0.328*"ny" + 0.321*"fokus" + 0.165*"hitta" + 0.159*"hjälpa" + 0.005*"möjlighet" + 0.004*"spännande" + 0.002*"tänka" + 0.001*"försöka" + 0.000*"igenom" + 0.000*"tillgänglig"
0.712*"väg" + 0.032*"börja" + 0.032*"väldig" + 0.032*"inspiration" + 0.021*"sätta" + 0.011*"förklara" + 0.011*"dra" + 0.000*"svenskproducerad" + 0.000*"livsmedelsproduktion" + 0.000*"maj"
0.397*"tid" + 0.395*"odla" + 0.130*"direkt" + 0.014*"handla" + 0.012*"kund" + 0.010*"ica" + 0.006*"se" + 0.004*"först" + 0.002*"system" + 0.002*"tech"
0.257*"ta" + 0.135*"jordbruksverk" + 0.130*"titta" + 0.130*"vecka" + 0.129*"hålla" + 0.127*"slänga" + 0.025*"län" + 0.011*"fågel" + 0.009*"internationell" + 0.004*"fjäderfä"
0.193*"utnyttja" + 0.192*"ytterlig" + 0.186*"kort" + 0.186*"testa" + 0.132*"nära" + 0.021*"fazer" + 0.011*"exempelvis" + 0.008*"bolag" + 0.007*"fazers" + 0.006*"industriell"
0.459*"företag" + 0.320*"livsmedelsföretag" + 0.069*"food" + 0.066*"utveckling" + 0.065*"innovativ" + 0.008*"finland" + 0.001*"swede" + 0.000*"utveckla" + 0.000*"verka" + 0.000*"organic"
0.357*"möjlighet" + 0.182*"genomföra" + 0.178*"förutsättning" + 0.178*"samhälle" + 0.013*"sverige" + 0.011*"behöva" + 0.011*"jobba" + 0.008*"erfarenhet" + 0.005*"intressant" + 0.005*"bransch"
0.277*"cirka" + 0.273*"samtidig" + 0.267*"resultat" + 0.135*"krona" + 0.008*"total" + 0.006*"ske" + 0.004*"dryg" + 0.002*"därmed" + 0.000*"lrf" + 0.000*"summa"
0.524*"sverige" + 0.198*"hållbar" + 0.157*"livsmedelspris" + 0.056*"miljö" + 0.029*"bero" + 0.029*"kaffe" + 0.000*"svenskproducerad" + 0.000*"livsmedelsproduktion" + 0.000*"naturlig" + 0.000*"far"
0.382*"producera" + 0.292*"produktion" + 0.187*"mejeri" + 0.095*"energi" + 0.008*"sverige" + 0.006*"egen" + 0.003*"tillgång" + 0.003*"omfatta" + 0.001*"gård" + 0.001*"storlek"
0.380*"marknad" + 0.251*"export" + 0.132*"ju" + 0.112*"vd" + 0.109*"växa" + 0.002*"leda" + 0.001*"verksamhet" + 0.001*"kommentar" + 0.001*"position" + 0.001*"usa"
0.461*"coop" + 0.158*"handel" + 0.103*"nära" + 0.083*"vara" + 0.079*"sälja" + 0.078*"producent" + 0.016*"ukraina" + 0.006*"försäljning" + 0.000*"produkt" + 0.000*"centrum"
0.378*"köpa" + 0.259*"konsument" + 0.130*"traditionell" + 0.129*"svar" + 0.013*"fysisk_butik" + 0.011*"lokal" + 0.009*"dagligvarubutik" + 0.009*"erbjudande" + 0.009*"icas" + 0.007*"dels"
0.227*"livsmedel" + 0.225*"pris" + 0.179*"forskare" + 0.090*"undersöka" + 0.089*"fungera" + 0.047*"grund" + 0.046*"studie" + 0.045*"jämföra" + 0.023*"påverka" + 0.003*"spannmål"
0.152*"värld" + 0.151*"bygga" + 0.151*"utveckla" + 0.149*"teknik" + 0.148*"möjliggöra" + 0.147*"samarbete" + 0.073*"syfte" + 0.003*"stiga" + 0.003*"växtbaserad" + 0.003*"basera"
0.315*"effekt" + 0.315*"mjölk" + 0.162*"människa" + 0.160*"kalla" + 0.009*"europa" + 0.002*"viktig" + 0.002*"miljard" + 0.002*"hos" + 0.000*"ungefär" + 0.000*"respektive"
0.777*"välja" + 0.170*"livsmedelsbransch" + 0.010*"affär" + 0.005*"lämna" + 0.002*"resa" + 0.002*"ekonomi" + 0.000*"for" + 0.000*"kännas" + 0.000*"förtroende" + 0.000*"restaurang"
```

Figure 5.2.2: All computed articles by the final LDA model.

5.2.3 User profile

An example distribution of the topics found in figure 5.2.2 is found in figure 5.2.3. The example is taken from a final user profile computed after the evaluation process was completed. The number shown on top is the unique user token for that specific user.

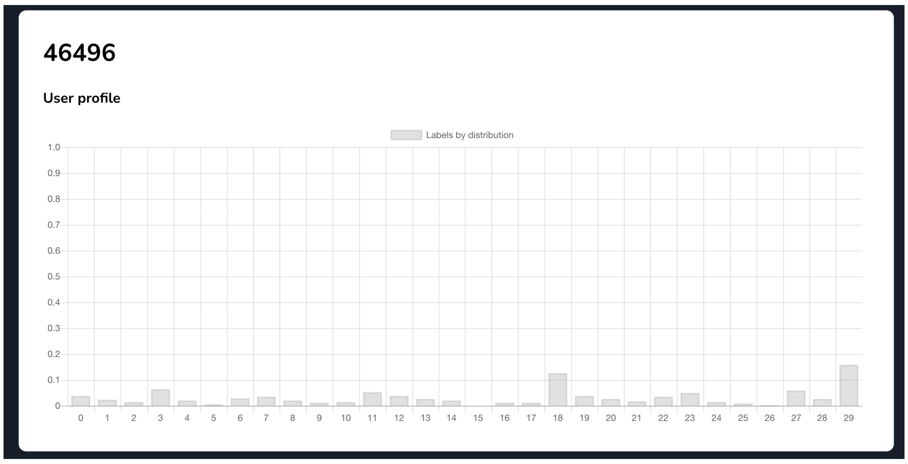


Figure 5.2.3: Example user profile.

5.2.4 Article profile

Another example showing the topic distribution inside of an article profile is shown in figure 5.2.4. This example depicts an arbitrary article taken from iteration number two. On the very top the identification number of the article, header and which iteration the article was added to is shown.

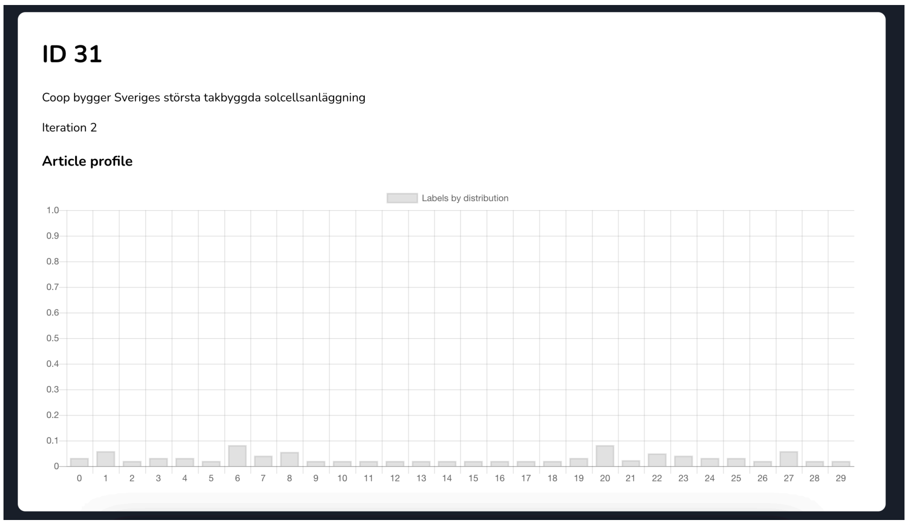


Figure 5.2.4: Example article profile.

5.2.5 Summarized text

Lastly, a final summarized text example is presented under this section. The original text taken from one of the articles from iteration number one is shown in listing 5.1.

```
"Kafferosteriet Löfbergs vinner kategorin "Newsroom of the "Year i årets
   upplaga av MyNewsdesks kommunikationstävling Digital PR Awards.Löfbergs
    får priset i hård konkurrens med Foodora, Tradera, Bauer Media och
   Advenica för att de med kreativ och meningsfull kommunikation lyckats
   engagera sin publik på ett innovativt sätt.Juryns motivering: Löfbergs
   visar med en genomtänkt och kreativ strategi hur man engagerar sin
   publik och bygger ett community i sitt nyhetsrum. Det är uppenbart vad
   deras huvudprodukt är, men med meningsfull och innovativ kommunikation
   använder de sitt nyhetsrum för att berätta historier och skapa känslor
   relaterade till sin produkt istället för att bara marknadsföra den.
   Enastående arbete"!Superkul! Det här är ett resultat av ett målmedvetet
   arbete där både kommunikationsgänget och andra på Löfbergs bidrar.
   Därför är vi många som är glada och stolta "idag, säger Anders Thorén,
   kommunikationschef på Löfbergs.Digital PR Awards delas ut av MyNewsdesk
    som vill uppmärksamma företag och organisationer som lyckats extra bra
    med sina PR- och kommunikationsinsatser. Tävlingen arrangerades i år
   för tionde gången.MyNewsdesk är en nordisk plattform för digital PR.
   Över 5 000 varumärken använder sina nyhetsrum på MyNewsdesk för att
   publicera nyheter och knyta relationer med media och journalister"
```

Listing 5.1: Original text of an article from iteration one.

Where after run through the final text summarizer is condensed into the text depicted in listing 5.2.

```
"Kafferosteriet Löfbergs vinner kategorin "Newsroom of the "Year i årets
   upplaga av MyNewsdesks kommunikationstävling Digital PR Awards.Löfbergs
    får priset i hård konkurrens med Foodora, Tradera, Bauer Media och
   Advenica för att de med kreativ och meningsfull kommunikation lyckats
   engagera sin publik på ett innovativt sätt.Juryns motivering: Löfbergs
   visar med en genomtänkt och kreativ strategi hur man engagerar sin
   publik och bygger ett community i sitt nyhetsrum. Därför är vi många
   som är glada och stolta "idag, säger Anders Thorén, kommunikationschef
   på Löfbergs.Digital PR Awards delas ut av MyNewsdesk som vill
   uppmärksamma företag och organisationer som lyckats extra bra med sina
   PR- och kommunikationsinsatser."
```

Listing 5.2: Summarized text of an article from iteration one.

Which is the final version presented for the evaluation process used to evaluate the article.

5.3 Results

Under this section the final data analysis is found. Initially presenting the chosen threshold of relevancy prediction for the prediction model, then going into detailed graphs of accuracy, precision and recall from the results. Lastly, a brief discussion on the evaluation for the text summarizer. Through the evaluation a total of 20 participants took part during a ten iteration long session, two iterations per week, in a total of five weeks.

5.3.1 Number of data points

In table 5.3.1 data for each iteration can be seen. Table 5.3.2 represents the number of people who performed x updates.

Iteration	1	2	3	4	5	6	7	8	9	10
Participants	17	17	15	13	12	10	9	7	3	6
Articles	14	7	11	4	7	4	14	5	12	4
Data points	238	119	165	52	84	40	126	35	36	24
Relevant answers	154	68	70	26	48	24	69	19	17	9
Neutral answers	46	21	42	13	19	10	42	10	13	10
Non-relevant answers	38	30	53	13	17	6	15	6	6	5

Table 5.3.1: Data for each iteration.

Updates	1	2	3	4	5	6	7	8	9	10
Participants	16	14	7	6	3	2	2	2	1	0

Table 5.3.2: Display of number of participants per update made.

The optimal number of contributors for each iteration is 20 as this is the total number of participants, however as can be seen from both tables the number of data points collected for each iteration respectively the number of participants who performed more than five updates are low which causes instability in the data presented below. Only one user managed to participate for nine iterations, whereas 16 people contributed with at least one update. Keep in mind that if a user submits evaluation answers for iteration one and six only, a total of six iterations is accounted for whereas only one update is performed regarding the user profile.

5.3.2 Prediction relevance threshold

The data from the evaluation is based on the prediction algorithm presented under section 4.8.3 where a threshold is introduced as the foundation for the prediction model performance. Altering this threshold will cause the model to predict in different ways producing more or less False Positive (FP)s relative True Positive (TP)s as well as False Negative (FN)s relative True Negative (TN)s. These values are used to then calculate a precision, recall and accuracy score for the model which is what is going to be presented later. Precision is a measurement of how many of the relevant predictions made were in fact correct and is calculated through

$$precision = \frac{TP}{TP + FP}$$

Recall depicts how many relevant predictions were found in the total scope of relevant evaluations by the model and is calculated through

$$recall = \frac{TP}{TP + FN}$$

Lastly, accuracy is a measurement of how many correct predictions the model made, regardless of if the prediction is relevant or irrelevant, this measurement is calculated through

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Generally, if a model is in its initial phase where the creation of user profiles is to be done, then a high recall value is wanted as it will in practice supply the user with a lot of irrelevant articles which essentially speeds up the user profile creation process by providing very different types of articles. On the other hand, a more productive model which presents as much relevant content as possible to a user is necessary for later stages when actually applying the model to a real life scenario. Here the highest possible precision value is wanted. This said, depending on where in the process an evaluation is, different values of precision and recall is wanted. Table 5.3.3 depicts the average measurements along with the resulting prediction values represented in TP, TN, FN, and FP. These values are taken as an average number from all iterations whereas the prediction relevancy threshold is altered for every run, as seen in the header of the table. The overall best combination of recall, precision and accuracy is seen to be 0.25, though the False Positive Rate (FPR) is high in relation to other thresholds, which essentially means that the model will suggest a lot of irrelevant articles using this threshold. It is possible to reduce the amount of false positives while still retrieving a fairly good number for recall, accuracy and precision by picking the threshold 0.5, which then will suggest less irrelevant articles.

Threshold	0.25	0.4	0.5	0.6	0.75	0.9
Accuracy	0.723	0.677	0.574	0.385	0.276	0.273
Recall	0.994	0.871	0.615	0.232	0.004	0.000
Precision	0.726	0.734	0.754	0.75	1.0	0.000
FPR	1.000	0.841	0.534	0.206	0.000	0.000
TP	501	439	310	117	2	0
TN	0	30	88	150	189	189
FP	189	159	101	39	0	0
FN	3	65	194	387	502	504

Table 5.3.3: Values for average measurements for each threshold with a changing user profile.

By analyzing the same model using a user profile which is created using default values and thus does not use a profile which updates over time, the following values are computed, as seen in table 5.3.4.

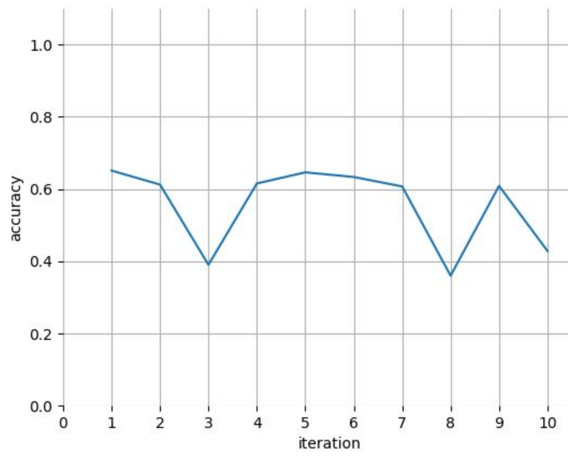
Threshold	0.25	0.4	0.5	0.6	0.75	0.9
Accuracy	0.727	0.703	0.657	0.558	0.312	0.273
Recall	0.857	0.759	0.637	0.393	0.044	0.000
Precision	0.651	0.636	0.607	0.541	0.199	0.000
FPR	1.0	0.96	0.820	0.471	0.101	0.0
TP	504	480	421	287	46	0
TN	0	7	34	100	170	189
FP	189	182	155	89	19	0
FN	0	24	83	217	458	504

Table 5.3.4: Values for average measurements for each threshold with a default user profile.

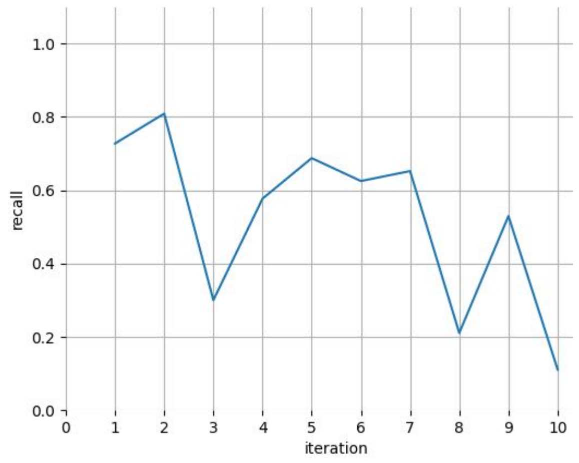
The measurements of accuracy, recall and precision are slightly higher but still comparable to the ones of the data shown in table 5.3.3 which uses an updating user profile. This leaves room for speculations as to if the user profiles really is necessary for this project, or rather if the articles were of sufficient quality to support the user profiles. However, solely looking at recall, precision and accuracy is insufficient in concluding that the user profiles did not contribute to the prediction model. Once again, looking at the FPR in table 5.3.4, one can see a higher value for almost every threshold compared to table 5.3.3, and as earlier mentioned, this means that the model with a default user profile will in general suggest more irrelevant articles to users overall.

5.3.3 Results based on threshold

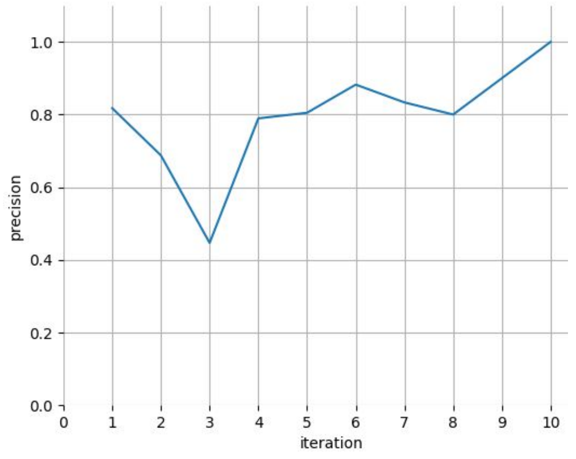
First under this section results for threshold 0.5 will be presented split into two different analyses. First an analyze of measurements over iterations, second an analyze of measurements over number of updates. After the analysis of different metrics, the evaluation results for the text summarizer is introduced and briefly discussed. Lastly, problems found with the results are discussed. The following graphs presented in figure 5.3.1 depicts the average numbers of accuracy, recall, precision and FPR according to each iteration.



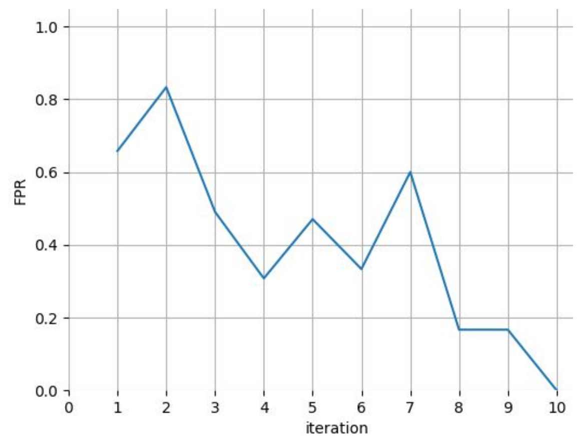
(a) Accuracy.



(b) Recall.



(c) Precision.



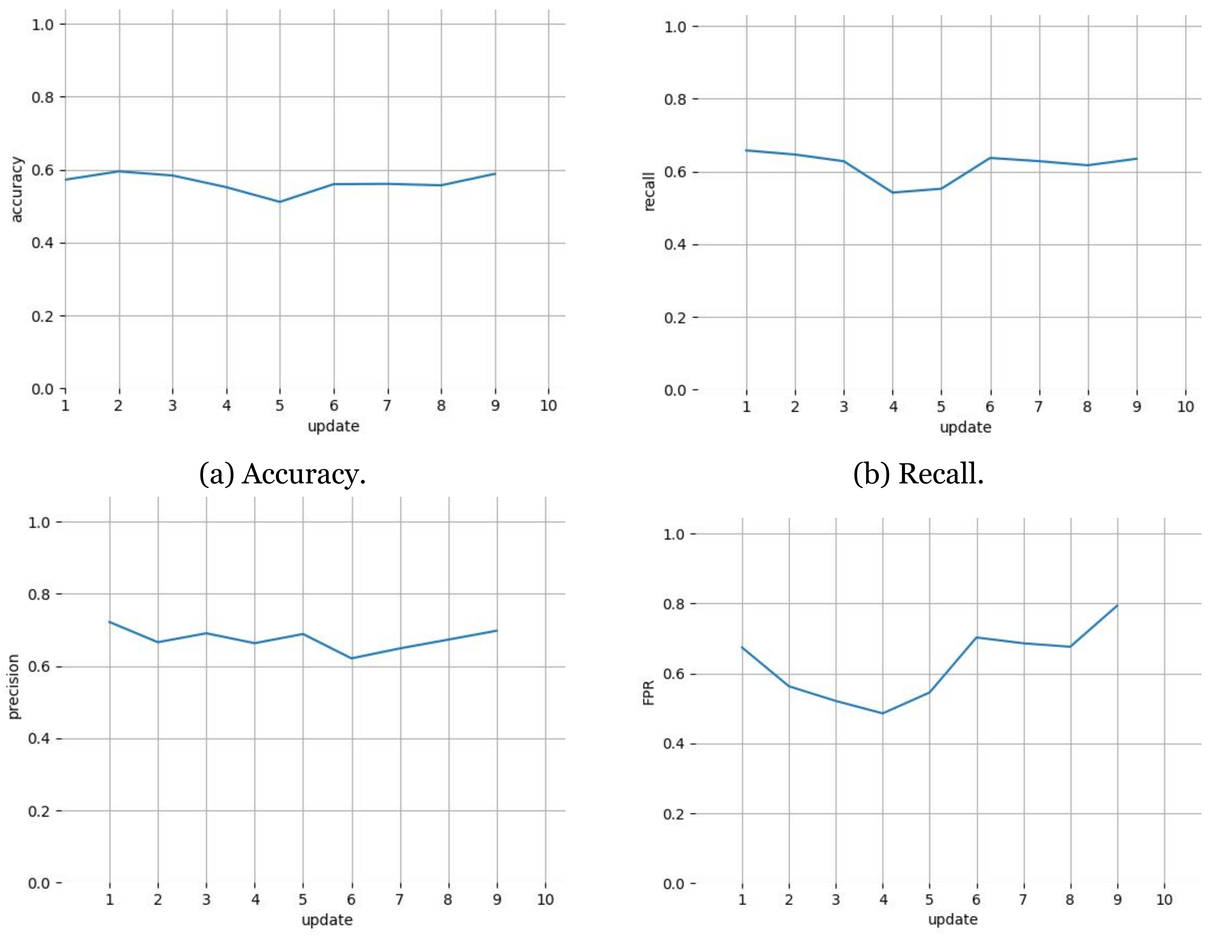
(d) False positive rate.

Figure 5.3.1: Measurement qualities for different iterations.

The accuracy per iteration is fairly stable but overall slowly decreasing over the iterations. This is not desirable results for this project as it indicates that the prediction model did not make more accurate predictions while progressing through the evaluation. However, due to the inconsistency in submitted evaluations and amount of articles to evaluate per iteration, this data is considered to be unstable and not very reliable. One iteration may consist of 5 articles and 7 participants (as seen in table 5.3.1, iteration 8), whereas another iteration may include 14 articles and 17 participants (iteration 1 in table 5.3.1). However, recall shows a decreasing pattern as the further into the evaluation process users go, the more the profiles should be updated, and as shown the recall value decreases, providing less irrelevant articles. As for the opposite to the recall value, the precision increases per iteration which indicates the opposite behaviour, providing more correct suggestions. Lastly, the false positive rate is dramatically decreasing per iteration which indicates an improvement in the model per iteration as per the amount of irrelevant articles presented. The further into the process the evaluation progresses, the more defined the user profiles should theoretically be and hence the model should be able to present less irrelevant articles in relation to the graph shown in 5.3.1d.

Further, since not every user participated in each iteration, a presentation of the same measurements but for the x amount of performed updates relative to each user is of interest.

The following graphs depicted in 5.3.2 shows the average numbers of accuracy, recall and precision compared to the amount of updates a user performed. Seen in table 5.3.4, there are not many data points for the higher number of update values, however there is a trend where the accuracy is increasing ever so slightly according to the amount of updates.



(a) Accuracy.

(b) Recall.

(c) Precision.

(d) False positive rate.

Figure 5.3.2: Measurement qualities for different updates performed by users.

It is hard to draw any conclusions to the amount of data retrieved for this project, however a slight positive growth is noticed as for accuracy and precision coming for update number five and upwards, which indicates a positively developing model. The precision as well as the recall values here are rather stable over the amount of updates performed with a somewhat high base value of around 0.7 and 0.65 respectively. As previously discussed, the wanted recall value should be lower the more updates a user completes as well as the precision should go up in the same manner, which could be indicated looking at the measurements over iterations and not at the measurements over the amount of updates. One minor detail is the steady increase of precision from update six and forth as compared to the overall decreasing value of recall also seen from update six, which indicates a small improvement of provided articles per update. Lastly, the analysis of the FPR in relation to amount of updates made is fairly hard to conclude as this measurement rather targets the models progress and not each individual. The numbers for the higher amount of updates are distorted due to the few amount of participants of the project who participated with a high amount of updates.

5.3.4 Text summarizer evaluation

In order to somewhat evaluate the text summarizer and see if the performance is sufficient two metrics are used. First, the understandability of the text, essentially if the text summarizer concatenated well enough summaries, second the length of the text. The length of the text is altered through the use of a score threshold mentioned under section 4.7, the metric of the length evaluation reflects on how well the chosen value is. By taking the average of each answer and summarizing this, a number representing the two factors is presented. The average understandability score is 3.867 and the average length score is 3.731. Since the average values of 3.867 for the understandability and 3.731 for the length are both above average, no further adjustment is made to the text summarizer as for this project. The results leave room for improvement, though realistically a near perfect score for a text summarizer is hard to achieve as the summarizer depends on a lot of factors which here can not be controlled such as the original content of the article. Though, by implementing an abstractive text summarizer instead of an extractive which is used here, the results may potentially be of better quality. Further, trying out different score thresholds for the extractive summarizer might increase the values further as well. This is not carried out through this project as the evaluation of the text summarizer had to be done in real-time by end users, and thus not enough time could be spent in validating these theories.

5.3.5 Problems

Due to the low amount of data points discussed under section 5.3.1 the confidence of final statements and conclusions is fairly weak. What further contributes to the low amount of data points is data points which do not reflect reality well enough. A scenario where a user misunderstands the evaluation technique and accidentally sends in answers without scoring relevancy effects the overall analyze of the model. Further, depending on the day to day mood of any user, the answers may differ a lot. One day a given user may feel more interested in topic A than topic B, whereas another day the same user might only be interested in topic B and not at all in A. This is something that can be neglected in an extensive research with a lot more data, however due to the small amount of time spent on this research, this factor may effect the results. Another issue is with the lack of variation in data where there is a majority of relevant evaluations vs non-relative evaluations. This may effect the results as to the high recall values found under table 5.3.3 as most evaluations found are relevant and thus increases the number. A data set where the proportion of relevant to non-relevant answers are closer to 50% would make it easier to evaluate the data and more confident conclusions. Other issues include user's cascading affect of reading several articles in a row, the further down the process one user goes the less they might consider the text or get tired of reading which per se impacts the answers.

Chapter 6

Discussion

Under this chapter there will first be a discussion about thesis contribution to the research area, then problems with the project will be discussed as well as potential future work. Lastly, the author's personal experience with the project is shared.

6.1 Contribution

The objective of this thesis was to retrieve information about how effective text summarization and information individualization would be to improve information gathering within the area of IT. From spending time on researching relevant topics and areas related to this topic the conclusion could be made that there is not much work done trying to optimize the matter of information gathering. The results of this project show no clear answer to whether or not this ultimately will be a feasible approach. More data of better quality and a larger evaluation session is needed as well as more effort into fine tuning the topic model along with the text summarizer. However, there clearly is a problem where developers within IT face massive amounts of information and has to scatter through the texts manually whereas much time is spent. It is also proven to be taking more time than needed from a couple of studies conducted showing the time spent and possible improvements to be made. Moreover, the focus of this project is to construct a technical basis to contribute to the solution of the problem whereas most papers found focuses on the exploration of the issue as well as possible approaches to solve it. Hence, the work of this paper will contribute with a new approach of solution to the issue which is the technical basis.

6.2 Problems

Throughout the project there has been a number of problems discovered along the way. Most problems will be presented and discussed under this section.

6.2.1 Implementation of ML models

A problem which originated early in the process was a lack of understanding of how the process of computing topic distributions would work in its technicality. Initially, a supervised ML model called Support Vector Machine [14] was setup to label articles based on previous examples of labeled articles, where one article is given one topic. This of course lost its purpose fast as the essence of topic distributing was completely lost through this approach. Once a complete understanding of how LDA worked in an unsupervised matter and the understanding that the output is non-binary, the ML model implementations was completely removed.

6.2.2 Swedish models

The topic of NLP is constantly growing and has been for a few years now [50]. However, since it is fairly new technology the language support is largely in English. There were some issues trying to find models such as the NER model and the lemmatizer with the correct language dictionary, which for this project is Swedish. The NER model introduced under 4.3.1 was the only choice of NER model available at the time supporting Swedish as computational language. Because of this, it was hard to evaluate the performance of the model since no comparison was available. One issue with that model was that it could only compute texts with a max length of 500 characters for reasons which were not further investigated, which had to be addressed when implementing the model. Further, the lemmatizer faced the same issue where not many Swedish alternatives were found and thus the presented lemmatizer in section 4.3.1 was selected. However the model is not complete as to the Swedish dictionary and thus had to be provided manual corrections to some words. This of course affects the accuracy of the lemmatizer as not every faulty word could be manually fixed since the scope of words in total were +20000 words. Though some words were found and manually fixed, such as ”hindret” which was lemmatized to ”hindr” and fixed to represent the correct lemmatization ”hinder”. If the words differ with just one letter they will not match the final computation of topic distributions which analyses the words of the articles, thus this poses a threat to the performance of the final model.
The conclusion from this is that the research and progress of NLP for the Swedish language is fairly new as not many examples of models are found supporting Swedish. Looking at the English contributions to the same area one can find plenty of alternatives.

6.2.3 Articles

As previously mentioned there is a language barrier accompanied with the research of NLP techniques. Since the articles in question collected from livsmedelsnyheter and livsmedelifokus are mostly in Swedish, this was the chosen language for the project. There were considerations as to translating the texts to another language or finding a foreign news source, however it would end up being unauthentic as to the user’s actual work environment. There still is the issue of if an article happens to be in English, or the mix of English words inside of an article, then the model would not be able to compute the articles, or rather it would compute but the output would be nonsense. This is unfortunately not addressed as for this model and does pose a threat to the results. Not only is the language barrier an issue coming from the articles, the amount of published articles do also differ a lot depending on choice of news sources. From observations, the two news sources did publish around five articles one day and the next few days none of the sources would publish anything. Since the scraper is setup to scrape between specific dates, this posed a problem as to the inconsistency in number of scraped articles per iteration. The amount varied from some iterations consisting of just four articles to other iterations having around 15 articles. Because of the small amount of news sources scraped, a rather small amount of data was originally scraped for the setup of the LDA model as well. As any other ML model, the more data fed to the model, the more accurate the model will be (as long as the data keeps certain quality). Finally, since there constantly erupts new companies and names of people around the area of food and beverage, it is hard to handle new names of organizations and people in a consistent way through the processing of the article texts. An attempt to handle the names of people is integrated by the use of a NER model, however organisations were ignored since organisation names may also be the names of people and thus hard to distinguish.

6.2.4 Evaluation

As to the evaluation process there could be more time spent on this in a more extensive study in order to generate more evaluations and more concrete results. The time span was only about one month and included ten iterations which in practice is an insufficient amount for any solid conclusions. Practically, the evaluation process could run for a much longer time where the results can be analysed in real-time and the pipeline can be enhanced during the evaluation process and in this way create the most optimal conditions for this project.

6.3 Future work

Under this section potential future work and research is presented. A discussion of organisation names is presented as well as the need for more news sources and an alternative implementation of a text summarizer. Lastly, a discussion of an extended evaluation session is found.

6.3.1 Organisation names

As mentioned under section 6.2.3 there is an issue with having organisation names and names of people in the same text. In future work one can put more time into trying to solve this issue in order to potentially further enhance the computed topics in terms of quality. The proposed NER model is supposed to handle the recognition of a company name and differentiate it to a human name, however there are work to be done for Swedish NER models in particular in order to make this happen. If one could successfully identify company names then it could be used to force categorize articles under company names to further gather information whereas an implementation of recommending interesting companies along with articles could be in place.

6.3.2 More news sources

More news sources could be added to further increase the amount of data points used for this project. This would increase each iteration data and thus could improve the results further. It might even come to a point where if enough data is scraped for each iteration, an algorithm could be constructed to choose which articles are to be added for maximal impact. Not only will this help the iterations, but also the initial setup of the LDA model where the more articles input the more accurate the model becomes, theoretically. Furthermore, this opens up the use for an abstractive text summarizer. What could be further interesting is to include irrelevant data sources in order to snowball the user profile updates. The more irrelevant versus relevant data the model is presented with along with the actual user relevance scores, the more accurate the user profiles becomes.

6.3.3 Abstractive text summarizer

The proposed extractive text summarizer works well with the small amount of data collected for this project. However, if enough data was collected, an abstractive text summarizer could be constructed and compared to the extractive one in order to retrieve the optimal summarization. There are existing interesting models which support abstractive text summarization such as Google's BERT [33] which is a pre-trained model on hundreds of thousands of articles.

6.3.4 Extended evaluation

The evaluation process for this project was limited in amount of testers and amount of time. If more people could attend the evaluation process, more data points can be collected and thus a more precise conclusion can be made.

6.4 Experience

This project was meant to further improve the knowledge about information gathering by the use of text summarization and individualization. In my opinion this has been achieved to some extent. Even though the results are not of the best quality or magnitude, the backbone of this thesis still contributes to the research and progress of information gathering within the area of IT. The conceptual prototype is constructed and assessed over time, however not enough participants evaluated their articles for every iteration which left a gap as to the final analysis of the evaluation results. Another factor was that most articles were in fact relevant, and thus the user profiles could not be properly updated. In terms of project plan and execution the overall process and implementation was according to plan. The implementation of the pipeline was fairly smooth but lacks in technical depth in some places such as adjustments of the LDA model and the extent of the implementation of Swedish models, where not enough models exists as of today. The constructed technical basis can once again be used for an extended study in order to gather more confident results in the future using the same implementations described under chapter 4. The proposed problems under section 1.2 were both addressed and solved in two separate ways where both were evaluated and assessed. The issue of too much information was targeted through a text summarizer where users could evaluate the length and understandability to the text whereas the difficulty of finding relevant information is addressed through the construction of user profiles.

Chapter 7

Conclusions

Introduced under section 1.3 the question of focus for this thesis is ”How effective is text summarization combined with individualized information recommendation in improving information gathering of IT experts?”. After evaluating the results of the constructed technical basis, unfortunately not much can confidently be said about the performance. Though through analyzing graphs of the results a trend can be seen as to the positive increase of accuracy in amount of updates performed by users. The precision and recall measurements per iteration shows an increasing tendency as well, theoretically making the model produce more relevant content over time. Furthermore, it could be stated that by the use of user profiles the prediction model predicted fewer false positives and thus in theory also provides fewer irrelevant articles based on this. The text summarization performed well enough where it on average resulted in a score higher than average. This could not be further investigated as the evaluation of the text summarizer requires real-time analysis where for this project the time came as a limitation. Even though the results are fairly inconsistent, the potential of performing this experiment in a bigger setting is good where the data shows slight tendencies of improvements. Properly applying the technical basis to a more robust and prolonged experiment has a greater chance of producing much better end results. Regardless of the insufficient amount of data points, this paper contributes to the research of efficient information gathering as well as proposes a technical solution with some analysis of the area. The paper proposes an alternative solution to the issue of information gathering and a foundation for a technical basis which can be further improved and implemented in a more authentic scenario. As for the research contributions of this paper, there are evident contributions to the research of information gathering, text summarization for Swedish language, topic modelling for the Swedish language and user profiling in general.

Bibliography

[1] Ahmad, Sharique, Wasim, Saeeda, Irfan, Sumaiya, Gogoi, Sudarshana, Srivastava, Anshika, and Farheen, Zarina. “Qualitative v/s Quantitative Research”. In: 6 (Oct. 2019), pp. 2828–2832. DOI: `10.18410/jebmh/2019/587`.

[2] Ajax. *Ajax Documentation*. 2022. URL: `https://www.w3schools.com/js/js_ajax_intro.asp` (visited on 05/17/2022).

[3] Alagha, Emily Couvillon and Helbing, Rachel Renee. “Evaluating the quality of voice assistants responses to consumer health questions about vaccines: an exploratory comparison of Alexa, Google Assistant and Siri”. In: *BMJ Health & Care Informatics* 26.1 (2019). DOI: `10.1136/bmjhci-2019-100075`. eprint: `https://informatics.bmj.com/content/26/1/e100075.full.pdf`. URL: `https://informatics.bmj.com/content/26/1/e100075`.

[4] Amazon. *AWS*. 2022. URL: `https://aws.amazon.com/?nc2=h_lg` (visited on 05/17/2022).

[5] BeatifulSoup. *Beatiful Soup*. 2022. URL: `https://www.crummy.com/software/BeautifulSoup/bs4/doc/` (visited on 05/15/2022).

[6] Blei, David M., Ng, Andrew Y., and Jordan, Michael I. “Latent Dirichlet Allocation”. In: *J. Mach. Learn. Res.* 3.null (Mar. 2003), pp. 993–1022. ISSN: 1532-4435.

[7] Christian, Hans, Agus, Mikhael Pramodana, and Suhartono, Derwin. “Single Document Automatic text summarization using term frequency-inverse document frequency (TF-IDF)”. In: *ComTech: Computer, Mathematics and Engineering Applications* 7.4 (2016), p. 285. DOI: `10.21512/comtech.v7i4.3746`.

[8] Dahlgren, Peter M. *Swedish stop words*. 2021. URL: `https://gist.github.com/peterdalle/8865eb918a824a475b7ac5561f2f88e9` (visited on 05/27/2022).

[9] David, Johnson, Malhotra, Vishv, and Vamplew, Peter. “More Effective Web Search Using Bigrams and Trigrams”. In: *Webology* 3 (Dec. 2006).

[10] Dorn,
Brian, Stankiewicz, Adam, and Roggi, Chris. "Lost while searching: Difficulties in information seeking among end-user programmers". In: *Proceedings of the American Society for Information Science and Technology* 50.1 (2013), pp. 1–10. DOI: https://doi.org/10.1002/meet.14505001059. eprint: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/meet.14505001059. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/meet.14505001059.

[11] Elvenite. *Om oss - Elvenite*. 2022. URL: https://elvenite.se/om-oss/om-oss (visited on 05/19/2022).

[12] Face, Hugging. *Hugging Face*. 2022. URL: https://huggingface.co/ (visited on 05/15/2022).

[13] Fatma, Fatma. *Industrial applications of topic model*. 2019. URL: https://medium.com/@fatmafatma/industrial-applications-of-topic-model-100e48a15ce4 (visited on 05/20/2022).

[14] Gandhi, Rohith. *Support Vector Machine — Introduction to Machine Learning Algorithms*. 2018. URL: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (visited on 05/22/2022).

[15] Gensim. *Gensim*. 2022. URL: https://pypi.org/project/gensim/ (visited on 05/15/2022).

[16] Griffiths, Thomas L. and Steyvers, Mark. "Finding scientific topics". In: *Proceedings of the National Academy of Sciences* 101.suppl_1 (2004), pp. 5228–5235. DOI: 10.1073/pnas.0307752101. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.0307752101. URL: https://www.pnas.org/doi/abs/10.1073/pnas.0307752101.

[17] Gross, Paul and Kelleher, Caitlin. "Non-programmers identifying functionality in unfamiliar code: strategies and barriers". In: *Journal of Visual Languages & Computing* 21.5 (2010). Part Special issue on selected papers from VL/HCC'09, pp. 263–276. ISSN: 1045-926X. DOI: https://doi.org/10.1016/j.jvlc.2010.08.002. URL: https://www.sciencedirect.com/science/article/pii/S1045926X10000431.

[18] Jiang, Kai and Lu, Xi. "Natural Language Processing and Its Applications in Machine Translation: A Diachronic Review". In: *2020 IEEE 3rd International Conference of Safe Production and Informatization (IICSPI)*. 2020, pp. 210–214. DOI: 10.1109/IICSPI51290.2020.9332458.

[19] Jolliffe, Ian T. and Cadima, Jorge. "Principal component analysis: A review and recent developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016), pp. 1–16. DOI: 10.1098/rsta.2015.0202.

[20] Kalepalli, Yaswanth, Tasneem, Shaik, Phani Teja, Pasupuleti Durga, and Manne, Suneetha. “Effective Comparison of LDA with LSA for Topic Modelling”. In: *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*. 2020, pp. 1245–1250. DOI: 10.1109/ICICCS48265.2020.9120888.

[21] Kersten, Mik and Murphy, Gail C. “Using Task Context to Improve Programmer Productivity”. In: *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. SIGSOFT ’06/FSE-14. Portland, Oregon, USA: Association for Computing Machinery, 2006, pp. 1–11. ISBN: 1595934685. DOI: 10.1145/1181775.1181777. URL: https://doi.org/10.1145/1181775.1181777.

[22] Laravel. *Laravel*. 2022. URL: https://laravel.com/ (visited on 05/17/2022).

[23] Laravel. *Laravel Artisan*. 2022. URL: https://laravel.com/docs/9.x/artisan (visited on 05/17/2022).

[24] Laravel. *Laravel Chart*. 2022. URL: https://v6.charts.erik.cat/getting_started.html#why-a-laravel-library (visited on 05/17/2022).

[25] Laravel. *Laravel Eloquent Documentation*. 2022. URL: https://laravel.com/docs/9.x/eloquent (visited on 05/17/2022).

[26] Laravel. *Laravel Forge*. 2022. URL: https://forge.laravel.com/ (visited on 05/17/2022).

[27] Laravel. *Laravel Mail Facade*. 2022. URL: https://laravel.com/docs/9.x/mail (visited on 05/17/2022).

[28] Liu, Chang, Liu, Jingjing, Cole, Michael, Belkin, Nicholas J., and Zhang, Xiangmin. “Task difficulty and domain knowledge effects on information search behaviors”. In: *Proceedings of the American Society for Information Science and Technology* 49.1 (2012), pp. 1–10. DOI: https://doi.org/10.1002/meet.14504901142. eprint: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/meet.14504901142. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/meet.14504901142.

[29] Liu, Haibin, Christiansen, Tom, Baumgartner Jr, William, and Verspoor, Karin. “BioLemmatizer: A lemmatization tool for morphological processing of biomedical text”. In: *Journal of biomedical semantics* 3 (Apr. 2012), p. 3. DOI: 10.1186/2041-1480-3-3.

[30] Livsmedelifokus. *Livsmedelifokus - om oss*. 2022. URL: https://www.livsmedelifokus.se/om/ (visited on 05/10/2022).

[31] Livsmedelsnyheter. *Livsmedelsnyheter - om oss*. 2022. URL: https://www.livsmedelsnyheter.se/om-livsmedelsnyheter/ (visited on 05/10/2022).

[32] Lu, Yihan, Hsiao, I-Han, and Li, Qi. “Exploring Online Programming-Related Information Seeking Behaviors via Discussion Forums”. In: *2016 IEEE 16th International Conference on Advanced Learning Technologies (ICALT)*. 2016, pp. 283–287. DOI: 10.1109/ICALT.2016.63.

[33] Lutkevich, Ben. *BERT language model*. 2020. URL: `https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model` (visited on 05/22/2022).

[34] Milano, Silvia, Taddeo, Mariarosaria, and Floridi, Luciano. “Recommender Systems and their ethical challenges”. In: *AI & SOCIETY* 35.4 (2020), pp. 957–967. DOI: `10.1007/s00146-020-00950-y`.

[35] Murphy, Gail. “Attacking information overload in software development”. In: *2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 2009, pp. 4–4. DOI: `10.1109/VLHCC.2009.5295312`.

[36] Nichols, James A., Herbert Chan, Hsien W., and Baker, Matthew A. “Machine learning: Applications of artificial intelligence to imaging and diagnosis”. In: *Biophysical Reviews* 11.1 (2018), pp. 111–118. DOI: `10.1007/s12551-018-0449-9`.

[37] NLTK. *NLTK*. 2022. URL: `https://www.nltk.org/` (visited on 05/16/2022).

[38] O’Brien, M.P. and Buckley, J. “Modelling the information-seeking behaviour of programmers - an empirical approach”. In: *13th International Workshop on Program Comprehension (IWPC’05)*. 2005, pp. 125–134. DOI: `10.1109/WPC.2005.24`.

[39] Oates, Briony J., Griffiths, Marie, and McLean, Rachel. *Researching Information Systems and computing*. SAGE, 2022.

[40] Ortiz, Marco A., Kurvers, Stanley R., and Bluyssen, Philomena M. “A review of comfort, health, and energy use: Understanding daily energy use and wellbeing for the development of a new approach to study comfort”. In: *Energy and Buildings* 152 (2017), pp. 323–335. ISSN: 0378-7788. DOI: `https://doi.org/10.1016/j.enbuild.2017.07.060`. URL: `https://www.sciencedirect.com/science/article/pii/S0378778816319089`.

[41] Perera, Nadeesha, Dehmer, Matthias, and Emmert-Streib, Frank. “Named entity recognition and relation detection for Biomedical Information Extraction”. In: *Frontiers in Cell and Developmental Biology* 8 (2020). DOI: `10.3389/fcell.2020.00673`.

[42] Prabhakaran, Selva. *Cosine Similarity – Understanding the math and how it works (with python codes)*. 2018. URL: `https://www.machinelearningplus.com/nlp/cosine-similarity/` (visited on 05/22/2022).

[43] Python. *Python DataFrame*. 2022. URL: `https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html` (visited on 05/15/2022).

[44] Qader, Wisam A., Ameen, Musa M., and Ahmed, Bilal I. “An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges”. In: *2019 International Engineering Conference (IEC)*. 2019, pp. 200–204. DOI: `10.1109/IEC47844.2019.8950616`.

[45] Ramsri, Goutham. *Simple abstractive text summarization with pre-trained T5 Text-To-Text Transfer Transformer*. 2020. URL: `https://towardsdatascience.com/simple-abstractive-text-summarization-with-pretrained-t5-text-to-text-transfer-transformer-10f6d602c426` (visited on 05/02/2022).

[46] Selenium. *Selenium*. 2022. URL: `https://www.selenium.dev/` (visited on 05/15/2022).

[47] Shin, Kilho, Ishikawa, Taichi, Liu, Yu-Lu, and Shepard, David Lawrence. "Learning DOM Trees of Web Pages by Subpath Kernel and Detecting Fake e-Commerce Sites". In: *Machine Learning and Knowledge Extraction* 3.1 (2021), pp. 95–122. ISSN: 2504-4990. DOI: `10.3390/make3010006`. URL: `https://www.mdpi.com/2504-4990/3/1/6`.

[48] Srivastava, Ridam, Singh, Prabhav, Rana, K.P.S., and Kumar, Vineet. "A topic modeled unsupervised approach to single document extractive text summarization". In: *Knowledge-Based Systems* 246 (2022), p. 108636. ISSN: 0950-7051. DOI: `https://doi.org/10.1016/j.knosys.2022.108636`. URL: `https://www.sciencedirect.com/science/article/pii/S0950705122002878`.

[49] Stanza. *Stanza*. 2022. URL: `https://stanfordnlp.github.io/stanza/pipeline.html#basic-example` (visited on 05/15/2022).

[50] Statista. *Revenues from the natural language processing (NLP) market worldwide from 2017 to 2025*. 2020. URL: `https://www.statista.com/statistics/607891/worldwide-natural-language-processing-market-revenues/#:~:text=The%5C%20NLP%5C%20market%5C%20is%5C%20predicted,interpret%5C%20and%5C%20manipulate%5C%20human%5C%20language.` (visited on 05/22/2022).

[51] Sun, Wenlong, Nasraoui, Olfa, and Shafto, Patrick. "Evolution and impact of bias in human and machine learning algorithm interaction". In: *PLOS ONE* 15.8 (2020). DOI: `10.1371/journal.pone.0235502`.

[52] Svensson, Karin and Blad, Johan. *Exploring NMF and LDA Topic Models of Swedish News Articles*. 2020.

[53] Widyassari, Adhika Pramita, Rustad, Supriadi, Shidik, Guruh Fajar, Noersasongko, Edi, Syukur, Abdul, Affandy, Affandy, and Setiadi, De Rosal Ignatius Moses. "Review of automatic text summarization techniques & methods". In: *Journal of King Saud University - Computer and Information Sciences* 34.4 (2022), pp. 1029–1046. ISSN: 1319-1578. DOI: `https://doi.org/10.1016/j.jksuci.2020.05.006`. URL: `https://www.sciencedirect.com/science/article/pii/S1319157820303712`.

[54] Zoetekouw, K.F.A. *A critical analysis of the negative consequences caused by recommender systems used on social media platforms*. July 2019. URL: `http://essay.utwente.nl/78500/`.

Appendix A

List of stop words

aderton adertonde adjö aldrig all allra alla allas allt alltid alltså andra andras annan annat artonde artonn att av bakom bara behöva behövas behövde behövt beslut beslutat beslutit bland blev bli blir blivit borde bort borta bra bäst bättre båda bådas både dag dagar dagarna dagen de del delen dem den denna deras dess dessa det detta dig din dina dit ditt dock dom du där därför då efter eftersom elfte eller elva en enkel enkelt enkla enligt er era ert ett ettusen fall fanns fast fem femte femtio femtionde femton femtonde fick fin finnas finns fjorton fjortonde fjärde fler flera flesta fram framför från fyra fyrtio fyrtionde få får fått följande för före förlåt förra första ge genast genom ger gick gjorde gjort god goda godare godast gott gälla gäller gällt gärna gå gång går gått gör göra ha hade haft han hans har hela heller hellre helst helt henne hennes heter hit hjälp hon honom hundra hundraen hundraett hur här hög höger högre högst i ibland idag igen igår imorgon in inför inga ingen ingenting inget innan inne inom inte inuti ja jag jämfört kan kanske knappast kolla kom komma kommer kommit kr kunde kunna kunnat kvar kör legat ligga ligger lika likställd likställda lilla lite liten litet lägga länge längre längst lätt lättare lättast långsam långsammare långsammast långsamt långt man med mellan men menar mer mera mest mig min mina mindre minst mitt mittemot mot mycket många måste möjlig möjligen möjligt möjligtvis ned nederst nedersta nedre nej ner ni nio nionde nittio nittionde nitton nittonde nog noll nr nu nummer när nästa någon någonting något några nån nåt nödvändig nödvändiga nödvändigt nödvändigtvis och också ofta oftast olika olikt om oss på rakt redan rätt sade sagt samma samt sedan sen senare senast sent sex sextio sextionde sexton sextonde sig sin sina sist sista siste sitt sju sjunde sjuttio sjuttionde sjutton sjuttonde själv sjätte ska skall skulle slutligen små smått snart som stor stora stort står större störst säga säger sämre sämst sätt så ta tack tar tidig tidigare tidigast tidigt till tills tillsammans tio tionde tjugo tjugoen tjugoett tjugonde tjugotre tjugotvå tjungo tolfte tolv tre tredje trettio trettionde tretton trettonde tro tror två tvåhundra under upp ur ursäkt ut utan utanför ute vad var vara varför varifrån varit varje varken varsågod vart vem vems verkligen vet vi vid vidare viktig viktigare viktigast viktigt vilka vilken vilket vill visst väl vänster vänstra värre vår våra vårt än ändå ännu är även åtminstone åtta åttio åttionde åttonde över övermorgon överst övre nya procent ser skriver tog året