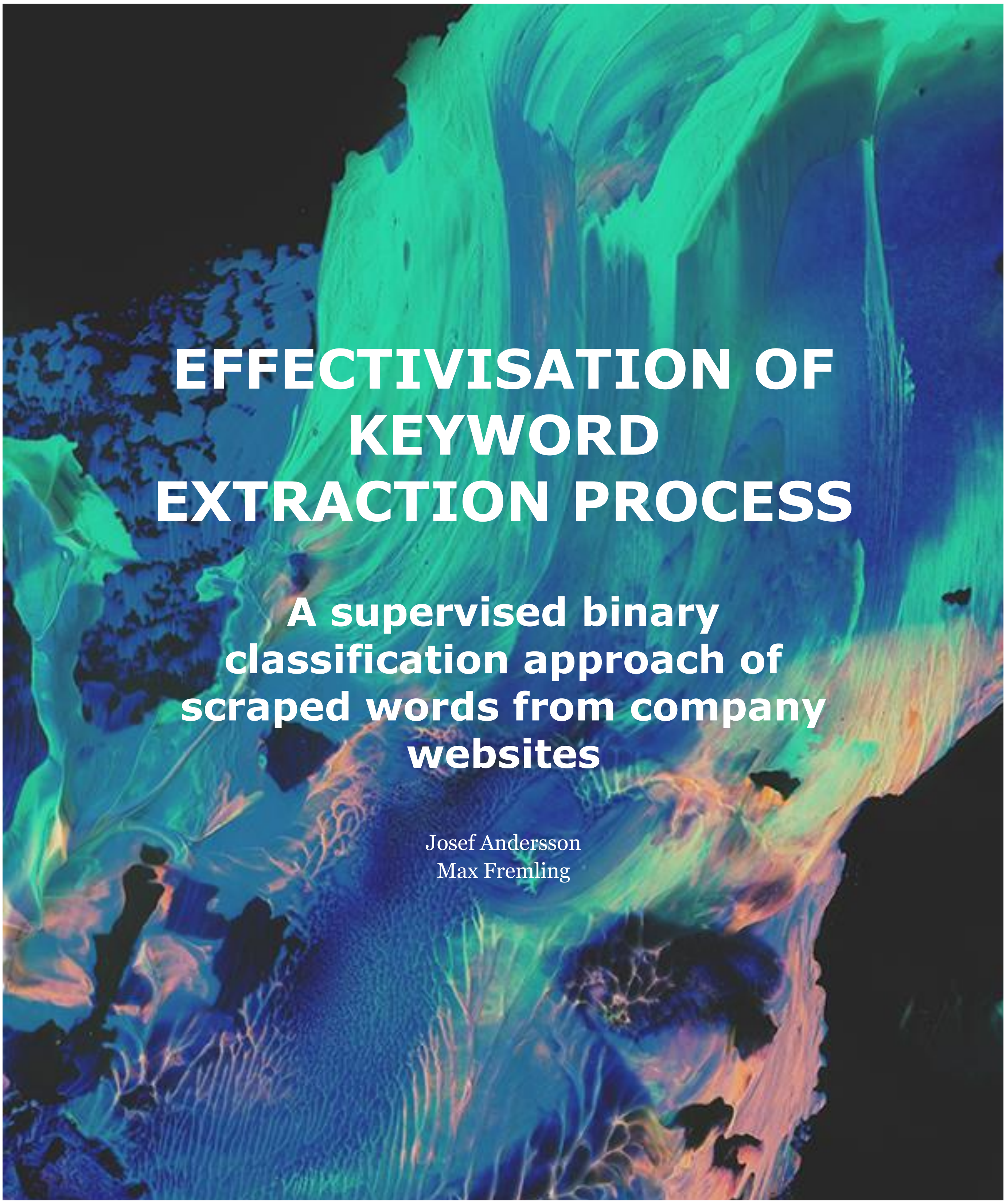
UMEÅ UNIVERSITY



EFFECTIVISATION OF KEYWORD EXTRACTION PROCESS

A supervised binary classification approach of scraped words from company websites

Josef Andersson
Max Fremling

Master Thesis, 30 Credits
M.Sc. Industrial Engineering and Management, 300 Credits
Spring term 2023

Effectivisation of keyword extraction process
A supervised binary classification approach of scraped words from company websites

Department of Mathematics and Mathematical Statistics
Umeå University
SE - 907 87 Umeå, Sweden

Supervisors:
Konrad Abramowicz, Umeå University
Daniel Hussam, Annalect Stockholm

Examiner:
Mehdi Moradi, Umeå University

Abstract

In today's digital era, establishing an online presence and maintaining a well-structured website is vital for companies to remain competitive in their respective markets. A crucial aspect of online success lies in strategically selecting the right words to optimize customer engagement and search engine visibility. However, this process is often time-consuming, involving extensive analysis of a company's website as well as its competitors'. This thesis focuses on developing an efficient binary classification approach to identify key words and phrases extracted from multiple company websites. The data set used for this solution consists of approximately 92,000 scraped samples, primarily comprising non-key samples. Various features were extracted, and a word embedding model was employed to assess each sample's relevance to its specific industry and topic. The logistic regression, decision tree and random forest algorithms were all explored and implemented as different solutions to the classification problem. The results indicated that the logistic regression model excelled in retaining keywords but was less effective in eliminating non-keywords. Conversely, the tree-based methods demonstrated superior classification of keywords, albeit at the cost of misclassifying a few keywords. Overall, the random forest approach outperformed the others, achieving a result of 76 percent in recall and 20 percent in precision when predicting key samples. In summary, this thesis presents a solution for classifying words and phrases from company websites into key and non-key categories, and the developed methodology could offer valuable insights for companies seeking to enhance their website optimization strategies.

Keywords - Machine learning, Keyword classification, Unbalanced data, Word embedding

Sammanfattning

I dagens digitala era är det avgörande för företag att etablera en närvaro online och upprätthålla en välskött webbplats för att förbli konkurrenskraftiga på sina respektive marknader. En avgörande aspekt av framgång online ligger i att strategiskt välja rätt ord för att optimera kundengagemang och synlighet i sökmotorer. Dock är denna process ofta tidskrävande och innefattar omfattande analys av ett företags webbplats samt dess konkurrenters. Detta arbete fokuserar på att utveckla en effektiv binär klassificeringsmetod för att identifiera nyckelord och fraser som utvinns från flera företags webbplatser. Datamängden som används för att skapa denna lösning består av cirka 92 000 skrapade prover, huvudsakligen bestående av icke-nyckelprover. Olika variabler extraherades ur datat, och en ordinbäddningsmodell användes för att bedöma varje extraherade ords relevans för dess specifika bransch och ämne. Logistisk regression, beslutsträd och random forest var de algoritmer som utforskades och applicerades på klassificeringsproblemet. Resultaten visade att logistisk regression utmärkte sig i att behålla nyckelord men var mindre effektivt när det gällde att eliminera icke-nyckelord. Å andra sidan visade de två träd-baserade metoderna sig överlägsna i klassificering av icke-nyckelord, dock på bekostnad av att felklassificera lite fler nyckelord. Totalt sett presterade random forest-modellen bäst baserat utifrån ändamålet, med ett resultat på 76 procent i recall samt 20 procent i precision när nyckelord predikterades. Detta arbete presenterar således en lösning för att klassificera ord och fraser från företags webbplatser i nyckel- och icke-nyckelkategorier, och den utvecklade metodiken kan ge värdefulla insikter för företag som vill förbättra sina strategier för webbplatsoptimering.

Acknowledgments

We would like to extend our deepest gratitude to our supervisors, Daniel Hussam and Konrad Abramowicz, for their exceptional guidance and support throughout the completion of this thesis work. Their expertise and invaluable contributions have significantly influenced the outcome of our research.

Daniel Hussam, our supervisor at Annalect, has been a very important asset to our thesis. His extensive expertise in the field of data science has provided us with valuable insights and practical perspectives. With his deep understanding of the search engine optimization process and as the creator of the project, Daniel has guided us exceptionally throughout our thesis work and its real-world applications.

Additionally, Konrad Abramowicz, our supervisor from the University, has been an irreplacable resource for the more theoretical aspects of our project. Konrad's guidance and thought-provoking suggestions have played a crucial role in refining our theoretical framework and conceptual understanding. His meticulous attention to detail, comprehensive feedback, and scholarly approach have elevated the academic rigor and depth of our work. We are deeply grateful to Konrad for his unwavering support, mentorship, and commitment.

Contents

1 Introduction

1.1 Annalect and their activities

Annalect is a unit under the umbrella of Omnicom media group (OMG) which works with data driven marketing strategy. Put simply, Annalect provides custom made decision bases for its customers through data analysis and insight generation (Annalect, 2023). In other words, Annalect works with providing data insight solutions the customers of OMG. The solutions that Annalect provides vary, but the main theme is solutions that provide the customers with insights that help them in forming marketing strategies, such as deciding on how to optimally distribute spendings within their marketing portfolio (Annalect, 2023).

1.2 Project background

Since the widespread adaptation of the internet in society, the way that people obtain information has shifted significantly, which in turn has shifted the structure and focus areas for companies' marketing strategies (Veglis and Giomelakis, 2019). Furthermore, the large role that search engines such as Google and Bing play in the customer's journey of going from the identification of a need to purchasing a product further amplifies this effect. For businesses, this is important to recognize and act upon when it comes to their marketing strategies. Subsequently, forming strategies to improve the company webpage is an important process for businesses to consider in order to succeed in the world of search engine customer journeys. The process of forming these strategies to improve the company website can be defined as search engine optimization, or SEO for short (Veglis and Giomelakis, 2019).

A part of having a successful company webpage composition consists of making sure that it contains important and relevant words for their specific industry, business and competitors. If that is the case, then when potential customers go into the search engine to search for a product or subject, their website appears higher up in the search results than their competitors. This is part of the challenge that companies of course need to face if they want to succeed against their competitors in the online sphere, and it can be defined as one of the main categories of the SEO process, namely keyword research and selection (Veglis and Giomelakis, 2019). This challenge is something that OMG, and as a result also Annalect, help companies face, and the goal is to help their customers websites appear higher up in the search results for browsing customers. The service of identifying these essential words and phrases is something that the Resolutions team of OMG normally develops. In this specific case however, Annalect contributed in building the service that helps select the key words and phrases from a company's website.

Normally, to start off an SEO project, specialists would manually go through competitors' and the customer firm's websites to gather keywords to analyze. This is a quite costly part of the SEO process as it is very time consuming while also needing highly qualified resources. At Annalect, they have developed an automated solution of this process to cut the time a specialist has to spend on such monotonous work. Their solution scrapes the website's text and removes the clearly unessential words and phrases and saves away the rest. The problem with this current solution is that there is no automation tool that can successfully classify the extracted words and phrases from the website as keyword or non-keyword. Currently, SEO experts therefore manually have to go through the all scraped words and phrases and select which of them should be classified as key and which should not. Annalect's current solution for keyword extraction is much more efficient than how it is traditionally done. However, it is still quite inefficient, as it requires a significant time of manual attention by the SEO experts. This results in the classification making up a large part of their SEO process.

1.3 The project

1.3.1 Project description

Due to the downsides of the current keyword extraction solution, Annalect has asked for a more streamlined solution to classify the scraped words and phrases as *key* or *not key*. The streamlined solution should greatly decrease the number of words that the SEO team has to check, while it is not necessary to remove every single one of the non-keywords from the scraped data. Moreover, the solution should be weighted towards correctly classifying the keywords, as a removed word or phrase runs the risk of never being checked again.

1.3.2 Purpose and goals

The main purpose of the project is to make OMG's existing search engine optimization pipeline more effective and therefore cost less in terms of time and resources. In the context of this project, that means reducing the time that SEO experts must spend manually classifying the keywords, as it is one of the least time efficient and most repetitive parts within the whole pipeline.

1.3.3 Boundaries and limitations

In order to limit the scope of the project and give more reasonable boundaries, the choice to limit the project to only the classification of the already extracted words and phrases was made. This was because the existing pipeline process that Annalect uses to gather the scraped words and phrases included several tools and different programming languages which we did not possess extensive knowledge within. Spending time learning and working with those languages and tools to improve the process from that angle would in all likelihood prove more time consuming than rewarding. Therefore, it was believed that including that part would cause more harm than good for the success and conclusions of the project as a whole.

1.3.4 Problem statement

Given the problem description, goals and limitations that are set, the following problem statement is defined:

1. Provide Annalect with a more streamlined solution to classify extracted words and phrases from their customers' websites as keywords or non-keywords.
2. The solution should especially avoid misclassifying actual keywords as non-keywords, i.e., avoid false negatives.
3. Multiple classification algorithms should be explored in order to tackle the classification problem.

1.4 General problem approach

The project goals are now set and the problem setting formed. The goal was to develop a classification system that accurately identifies scraped words and phrases from Annalect's customers' and their competitors' websites as either key or not key. Furthermore, this classification system should be weighted towards correctly classifying keywords and key phrases, as it is worse to miss a keyword than to include a non-keyword. Ultimately, the success of the project will depend on the ability to extract underlying information from the provided scraped data and accurately identify and leverage the most informative features to develop a reliable and scalable classification model. With this in mind, the following general steps will be followed to take on the given problem:

1. Investigate previous approaches to similar problems.
2. Explore the available data.
3. Clean the data.
4. Extract features from the cleaned data.
5. Create classification models using the extracted features.
6. Analyze results and provide final model(s).

2 Literature review

To begin solving the problem at hand with a fitting solution, a quite expensive analysis of previous works in the field was made. This was done to get an understanding of previous approaches that have been used to solve similar problems, as our own experience with keyword extraction was quite limited. From the literature review, three key main papers were chosen, all of which are presented in Sections 2.1, 2.2 and 2.3, along with their research questions and approaches to solve their similar problems. Each of these present their own main tools for classifying the keywords in their own respective problems, tools which also should prove useful in our thesis work. Subsequently, more general information regarding keyword extraction methods is presented in Section 2.4 to further shed light on existing and practiced techniques that these three approaches might not have used.

2.1 Improved Automatic Keyword Extraction Given More Linguistic Knowledge

In this paper, the author experimented with automatic keyword extraction from abstracts of scientific papers (Hulth, 2003). The research employed a supervised learning approach, with the primary objective being the exploration of incorporating linguistic features into the model, rather than relying solely on statistical features. Hulth conducted experiments using three distinct keyword extraction methods: *n-grams*, *noun phrase chunks*, and *part-of-speech patterns*. The n-gram approach involved the utilization of phrases with a total word count between 1 to n, where n was set to 3 by Hulth. Consequently, this allowed for the extraction of uni-, bi-, and trigrams, which encompassed more than 90 percent of all keywords Hulth (2003). Moreover, it was applied additional processing steps to the extracted phrases, including stemming and the removal of stopwords. Another approach employed by Hulth was the noun phrase chunk method, which relied on the descriptive nature of nouns and their capacity to enhance content explanation. She noted that a significant majority of the extracted keywords were either singular noun terms or noun phrases accompanied by adjectives. Lastly, Hulth investigated the approach based on part-of-speech (POS) patterns for keyword extraction. This approach involved leveraging the patterns derived from part-of-speech tags within phrases, as implied by its name. The author defined 56 distinct part-of-speech patterns, each of which occurred frequently in manually assigned keywords. Notably, 51 of these patterns included a noun. Among the various part-of-speech patterns, the most frequently occurring ones as keywords were:

- Adjective noun (singular or mass)
  - Example: tasty apple or fresh air
- Noun noun (both singular or mass)
  - Example: fruit basket or food research
- Adjective noun (plural)
  - Example: tall buildings
- Noun (singular or mass) noun (plural)
  - Example: fruit baskets or water resources
- Noun (singular or mass)
  - Example: apple or air

where singular nouns denote individual entities such as *book*, while plural nouns refer to multiple instances, for example *books.* Mass nouns represent uncountable substances or concepts, some of them are *water, knowledge* and *happiness.* Using the appropriate noun form is vital for clear and accurate academic writing.

For the three keyword extraction methods, the features used for training had been the same, being four in total. These four features were the term frequency, collection frequency, relative position of first occurrence and POS tags. Term frequency is simply the percentage of a specific word occurring in a document divided by the total number of words within a document. Collection frequency, or inverse document frequency as it is also called, is the number of documents containing a word divided by the total number of documents, i.e, the percetage of pages containing a word, and taking the logarithm of it. The next feature used, first occurrence, is the index of a word's first time occurrence given all words, divided by the total number of words. Lastly, the part of speech tag feature is simply the combination of all part of speech tags corresponding to each word within a phrase, where for example "Beautiful sky" becomes "noun, adjective".

The classification method Hulth used was *bagging*, which makes models have smaller variance and become more robust in their prediction (James et al., 2013). However, she makes clear that the purpose of the paper was not to compare machine learning approaches and that she makes no statement whether bagging as a machine learning approach is superior to others.

The findings of Hulth's paper are multiple, but the most significant one was that when including part of speech tags the performance of *F-score* and *precision* increased regardless of keyword extraction approach. The best method for *recall* as metric was using the POS pattern approach without including POS tags. These metrics are explained further in Section 7.2 when applied to our work in this thesis.

2.2 ClRank: A Method for Keyword Extraction from Web pages using clustering and distribution of nouns

The paper by Rezaei et al. (2015) is about a new keyword extraction method named *ClRank*. The method aims to extract keywords on web pages and the method itself is an unsupervised approach based on clustering. They divided the method into six main steps, *1. Preprocessing, 2. Part of Speech Tagging, 3. Lemmatization, 4. Similarity measure, 5. Clustering, 6. Selecting keywords*. The preprocessing step involved extracting the information from the website into a document object model (DOM) tree. When extracting the text, styling, numbers, symbols and scripts were removed. Furthermore, navigation menus were removed as their intention was to only look at the *main text* of the page. Next, part of speech tagging was implemented on all the words and then they extracted only the nouns. This was done as the majority of keywords selected when manually assigning them are either a noun or noun phrases.

The next step, lemmatization, involved removing the endings of all words and transforming them into the base word, such as for example transforming *fishes* into *fish*. The similarity measure step is simply measuring the semantic similarity between all unique word lemma pairs. The measurement used was the *Wu and Pulmer measure*, a measurement based on WordNet which is a large lexical database of the English language (Wordnet, 2023).

The authors' choice of clustering method was hierarchical clustering using the agglomerative method, which is one of the most commonly used clustering methods and does require predefining how many clusters should be created (James et al., 2013). In the paper they argued that being able to control how many clusters were used was one of the reasons they choose to use hierarchical clustering over other clustering techniques, which was done by setting a distance threshold. Once the clustering was done, each cluster was supposed to represent a specific concept where the authors then calculated the frequency of words in each cluster (distribution), as more relevant clusters should have higher frequencies. By calculating the frequency of words within each cluster, the different clusters could be ranked and the clusters with the lowest frequencies could be removed. This was done to avoid using clusters with irrelevant nouns, such as clusters based on advertisements on the used webpage. Once the clusters were created, the keyword selection step began. To choose a keyword, they ranked the nouns based on term frequency within each cluster. The authors' rule for extracting the keywords is shown in equation 1, where *Frequency* represents the term frequency for the specific noun, and *maxFrequency* represents the highest term frequency in the cluster. These conditions allowed for extracting only the most relevant words within each cluster.

$$\begin{aligned} &Frequency \geq 0.2 \cdot maxFrequency \\ &Frequency > 3 \end{aligned} \tag{1}$$

The results of the ClRank extraction method proposed in the paper was compared to two widely known and used keyword extraction methods, namely TextRank and Term frequency. A summary of their findings is that the authors' proposed method was performing better than both TextRank and Term frequency, especially for the *precision* metric, going from as low as 0.36 to 0.51. When analyzing *recall*, there was not very much difference in performance, where the methods most of the time produced equal results. The results themselves were compared to two different manually assigned keyword setups by humans. Something that Rezaei et al. mentioned was that these manually selected keywords had a direct effect on the results. For example, from *Set 1*, the *precision* was highest as it contained the most keywords, whereas *Set 2* had the highest *recall* value as it contained less and more precise keywords.

From the paper, the main and most important approach for their process can be summarized as the clustering of nouns to find potential keywords, which allowed for subsequent ranking of the clusters by calculating the clusters' term frequency. This was way of using clustering methods to find keywords was an effective approach, and could have good application possibilities for the implementation of our thesis work.

2.3 ES2Vec: Earth Science Metadata Keyword Assignment using Domain-Specific Word Embeddings

In the paper written by Ramasubramanian et al. (2020), they discussed how keywords are important for the discovery of articles as well as their importance to ensure that information is consistent and in similar manner. Consequently, they created a new keyword extraction method tailored specifically for scientific papers. The proposed approach employed a so called *word embedding*, a concept that is explained in Section 5, to analyze the words within an abstract and suggest relevant keywords for its paper. The authors trained their own *Word2Vec* model, which is a specific type of embedding model, on a science corpus to achieve a domain-specific word embedding. In other words, they trained their own language model to represent words in a numerical space, and that model then helped in deciding which words are relevant, i.e. key, for a paper in a specific domain.

Since the authors trained their own embedding model on a text corpus within a specific field of study, their belief was that it would make the embedding capture more linguistic and domain relevant relationships for the words and therefore predict more relevant keywords in that certain area. Given the results from the paper, the word embedding did indeed achieve the best performance compared to more universal embeddings, getting an *precision* of 79.4%, where previously the best was 72.8%. However, as the paper only used precision and not recall, the results did not take into account missed or misclassified keywords. This could lead to the results being perceived better than they might actually be, especially when comparing to other keyword extraction methods. In summary, both models performed quite using their provided metric, though the subject specific one had a bit of an edge. Some of the models the authors compared against were glove-twitter-200, glove-wiki-gigaword-100 and Word2Vec-google-news-300.

Generally, the authors' method to extract keywords from domain specific abstracts is quite extensive. However, for us, the main takeaway is their use of a word embedding and its uses to help classify keywords. Another interesting finding is that despite the fact that the subject specific word embedding proved most effective, more general use word embeddings performed quite similarly.

2.4 General findings

While researching for keyword extraction methods, the metrics used for evaluating models have almost been unanimous. These metrics are precision, recall and F_1 score, which will be explained in Section 7.2. Likewise, there are some features that are used more than others when training a keyword extraction model. Among these, the most frequently used features from our research have been term frequency, part-of-speech tags and term frequency-inverse document frequency (TF-IDF). The usefulness and feasibility of these features will be investigated in relation to the thesis and the data available, while also inclusing their definitions in Section 6.2. Apart from these features, the first-occurance feature was also found to be one of the most frequently used features for the given problem setting.

Furthermore, a summary made by Firoozeh et al. (2020) of features used by other researchers when creating a keyword extraction method is shown in Table 1.

Table 1: Feature extraction methods used by different papers to conduct keyword extraction.

Approach	**Morpho-syn**	**Statistical**	**Info**
(Salton et al. 1975)		✓	
(Ohsawa et al. 1998)		✓	
(Frank et al. 1999)		✓	✓
(Turney 2000)	✓	✓	✓
(Hulth 2003)	✓	✓	✓
(Matsuo and Ishizuka 2003)	✓	✓	
(Turney 2003)		✓	✓
(Mihalcea and Tarau 2004)	✓	✓	
(Yih et al. 2006)	✓	✓	✓
(Medelyan and Witten 2006)		✓	✓
(Wan and Xiao 2008)	✓	✓	
(Zhang et al. 2008)	✓	✓	✓
(Lopez and Romary 2010)		✓	
(Rose et al. 2010)		✓	
(Boudin 2013)	✓	✓	
(Caragea et al. 2014)		✓	
(Sterckx et al. 2016		✓	

To be noticed, the most used features are statistical ones followed by morphological and syntactical features, then informational features. *Statistical features* are features based on metrics that are measured, such as term frequency and first occurance. *Morpho-syntactic features* are the features that explain words gramatically definitions, such as a words POS or if a noun is a plural, singular or mass. Lastly the *informational features* are features which explain how the text is written, for example if the word is written as a title or plain text but also if the text is capitalized or italic. Considering the three feature types at hand, their applicability to our thesis given the data used remains uncertain. However, given the results of the table, statistical and morpho-syntactic features seems to be the most relevant.

From the research papers reviewed, the preferred approach seemed to be unsupervised as it allows for a broader usecase of keyword extraction. However, many papers still used a supervised approach as it in theory it should result in a more precise model for the specific application. However, despite perhaps producing a better model, it will be at a cost of being language and domain specific, which unsupervised models does not have to be. A relevant reasoning why researchers use unsupervised over supervised methods seems to have been the simple fact that it is quite labor intensive to get good data to train on using a supervised approach. Since the data provided for our thesis is labeled, which is discussed more in Section 3, we made use of the labels and therefore used supervised approaches only. This is especially acceptable since the solution Annalect is looking for is specifically meant for Swedish websites and we can therefore disregard the issue of language dependancy.

A final finding made was that the majority of the researched papers used text from documents or papers and not websites. Since the data for our thesis work came from websites, this created a difference in what information could be retrieved and extracted from our data compared to previous approaches' data. Due to this, some previous approaches simply can not be directly implemented in our specific problem.

3 Data

This section is dedicated to presenting and explaining the data that was provided by Annalect. Firstly, an overview of the acquired data will be presented in a table in order to give a visual overview as well as a broad view of the available variables. Secondly, every variable will more thoroughly be explained under Section 3.2 in order to give more understanding of what each variable represents and what different values they may adopt.

3.1 General data overview

Presented in Table 2 is a summary of the data set that was provided by Annalect at the initial stage of the project. As can be observed, there are in total 10 variables and almost 95,000 samples available in the data set. The two probably most important variables are the *keyword or phrase* and *is excluded* variable, where the former represents the keyword or phrase which is to be classified and the latter the labeled classification for each sampled word or phrase. Naturally, this means that the data can be viewed as supervised, where the *is excluded* variable is the label for each sample. Finally, table 2 reveals that there are samples that contain empty values for some variables. This will be covered and explained more in depth in the following section.

Table 2: Overview of the available data for the project.

Index	**Client**	**Website**	**Industry**	**Topic**	**Keyword or phrase**	**Is excluded**	**Competitor frequency**	**Category**	**Subcategory**	**source**
0	Client 1	https://www.client1.se/	Retail	Clothing	dagars	True	{'Competitor 1': 2, 'Client 1: 11'}			scraped
1	Client 1	https://www.client1.se/	Retail	Clothing	design	True	{'Competitor 2': 30, 'Competitor 3': 928, 'Client 1': 715}			scraped
2	Client 1	https://www.client1.se/	Retail	Clothing	välkommen	True	{'Competitor 2': 4, 'Client 1': 13}			scraped
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
94551	Client 5	https://www.client5.se/	Beauty	Fat Removal	body-jet	True	{}			scraped
94552	Client 5	https://www.client5.se/	Beauty	Fat Removal	kropps-skulptering malmö	False	{}	Fat Reduction		scraped
94553	Client 5	https://www.client5.se/	Beauty	Fat Removal	kropps-skulptur dekoration	False	{}	Fat Reduction		scraped

Term usage

As is visible in Table 2, the keyword or phrase variable contains both words and phrases. As such, the terms "keyword" and "key phrase" will both be used somewhat interchangably throughout the report. In addition, the term "key sample" and "non-key sample" refer to samples containing a keyword or keyphrase and non-keyword or non-keyphrase respectively.

3.2 Variable definition & explanation

In this section each variable will be separately defined and explained in its own subsection. The idea behind this is that it will make the meaning of each variable easier to understand.

3.2.1 Client

Firstly, there is the *client* variable. This variable is very straightforward, and it simply represents the client company to which each sampled word or phrase belongs. There are in total 5 clients in the data, and the choice has been made to mask their actual names as to keep some form of confidentiality. As such, the five different clients have been renamed to Client 1, Client 2, and so on.

3.2.2 Website

The *website* variable represents the client's base website for each sample. This website, as well as the competitors' websites, have been scraped and it is from there that all the words and phrases in the data set come from. Naturally, since the clients' names have been masked, so have also the website names, which have been changed similarly as the client name as such: http:www.client1.se/. Subsequently, as with the client variable, the website variable contains 5 unique values in total represented by http:www.client1.se/, http:www.client2.se/, and so forth.

3.2.3 Industry

The *industry* variable is categorical and contains information about the industry to which the client company belongs. All in all, the clients belong to three different industries: retail, insurance and beauty. More specifically, Client 1, 2 and 3 belong to retail, Client 4 to insurance and Client 5 to beauty.

3.2.4 Topic

Similar to industry, the *topic* variable is also categorical. It describes the topic that the company's products belong to. In total, there are four different topics among the five clients in the data set. Client 1's topic is clothing, Client 2 and 3's is hair products, Client 4's is home insurance and Client 5's is fat removal.

3.2.5 Keyword or phrase

The *keyword or phrase* is the variable that holds the main role in this thesis project. The words and phrases in this column are the strings that are to be classified as either key or not key. The strings can be both singular words, composite words or multiple words separated by space. The vast majority of these words are in Swedish, however there are instances of words in other languages, mainly English and Nordic ones.

3.2.6 Is excluded

Is excluded is the label to each word or phrase, and it is this classification label that the supervised model will aim to emulate. Put simply, *True* in this column represents that a word or phrase is not a keyword and should be excluded from the set, while *False* means that the word or phrase is a keyword and thus should not be excluded from the set.

3.2.7 Competitor frequency

Competitor frequency represents the only variable containing numerical information. It represents the frequency of each word or phrase on the client's and its competitors' websites. Since each word or phrase thus has more than one frequency to display in the *competitor frequency* variable, they are represented by a dictionary containing the information for all competitor frequencies as well as the client's. In other words, to use Python terminology, the *competitor frequency* entries are represented by dictionaries containing key and value pairs.

Furthermore, as can be noted in Table 2, the given data set does contain some samples where the frequency column is empty. Specifically, there were in total exactly 700 scraped samples with missing frequency values. Of those 700 samples, 496 were samples from Client 4 and 204 were samples from Client 5. All in all, there are thus not too many samples without a competitor frequency when put in relation to the total 95,000 samples.

3.2.8 Category

The *category* variable is a categorical variable represented as strings that describes what category the word or phrase has. As can be noticed in Table 2, it is only the samples that have the label False in the keyword or phrase column, i.e., samples marked as key, that have been assigned a category. Therefore, this variable, in contrast to other variables, is a result from the classification itself. As such, it is not really relevant for this thesis work as the goal is to make the prediction itself.

3.2.9 Subcategory

Similar to the category variable, the *subcategory* variable only has values for samples that have been classified as key. However, unlike the category variable, most of the key samples do not have an assigned subcategory, which can be seen from Table 2. The variable itself, as the name states, represents what subcategory the word or phrase belongs to. Finally, just as with the category feature, this feature is not relevant to this thesis work.

3.2.10 Source

The final variable in the data set is the *source* variable and it represents the source from which each word or phrase comes from. There are four sources in total: *scraped*, *google related*, *SEMRush related* and *SEMRush volume*.

After some communication with Annalect, it stood clear that it were only the *scraped* samples that were relevant for this thesis work. The reason was that during the existing keyword extraction procedure the other sourced samples were produced from the scraped samples that were classified as key. In other words, the first step is to classify the scraped samples, and then the samples of the other sources can be be produced. For this reason, the three other sources than the scraped one will not be further explained or used as they have no real relevance to this project.

3.3 Data retrieval process

As to shed some light on the data available for the project, a brief, and more general description of the process will be covered in this section. However, as to keep some confidentiality, the exact method of how OMG collects their data will not be covered.

As previously noted, there are five clients in total in the data set. The data samples belonging to each client, one through five, come from previous key word extraction processes performed by OMG. To explain through an example, the process of extracting the scraped key words and phrases from Client 1's and its competitors' website will be described below.

First, Client 1's website, as well as their specified competitors' websites, are all scraped using scraping tools. From there, a natural language processing (NLP) model splits up the raw text that has been scraped into smaller strings while removing some clearly non-important elements within it, such as HTML tags. From these steps, the *keyword and phrase* variable has been obtained. Also, the *competitor frequency* is extracted variable by calculating the occurrence of each string on each specific website. The categorical variables *industry* and *topic* are set manually. As was described in Sections 3.2.3 & 3.2.4, each client has only one topic and one industry, so this manual ascribing is not so time consuming.

Now that each sample's word or phrase as well as its topic and industry are set, the SEO specialists begin to manually label each sample either True or False - relating to the *is excluded* variable. This is not done mainly by manually scrolling through the entirety of the scraped words, but instead by using a sort of manual search method. In the case of Client 1, the SEO specialists would look at their specific business, topic and industry, and search the scraped samples for words that define these. In other words, the process can be compared to using the *CTRL Find* command used on Windows computers and search for key words and phrases that way. For example, the SEO specialist could search for the Swedish word "tröja", which means sweater, find all the words and phrases containing that word, and from there choose which of those words and phrases classify as a key. As such, the classification itself is done qualitatively by the SEO specialist, and there are no formal rules as to what constitutes a key word or phrase. When the keywords and key phrases are set, they are manually ascribed a category by the SEO specialist. Furthermore, some of them are also ascribed a subcategory.

When the keywords from the scraped samples have been identified, the dataset is then expanded with the Google related, SEMRushVolume and SEMrushRelated samples as well. As explained in Section 3.2.10, however, these types of samples have no relevance to this thesis project and are not the samples which our solution will be classifying. For this reason, the process of producing those alternatively sourced samples will not be explained.

3.4 Exploratory data analysis

Worth to underline before introducing the exploratory data analysis is that all sections moving forward only concern the scraped samples in the data, and not those of the *Google related, SEMRush Volume or SEMRush Related* sources. The reason is, as stated in Section 3.2.10 that this thesis work focused only on classifying the scraped samples, as they were the most crucial and challenging source to classify within the keyword classification process. Once all non-scraped samples were removed, the remaining scraped data set consisted of a bit over 92,000 samples.

With a data set of over 92,000 samples and 10 variables, there was a lot of information to explore. Exploratory data analysis (EDA) was a crucial step in the data analysis process, as it can provide insights about the data and identify potential issues that may cause problems if not handled correctly early in the process. The EDA was further made to assist in identifying patterns, relationships, and trends in the data, as well as detect outliers and anomalies that may require further investigation. Furthermore, EDA also provided information to help understand which variables were important for a classification model.

3.4.1 Sample distributions for the different categories

In this section, the sample distributions for the categorical variables client, topic and industry will be presented. Specifically, it will be presented how many samples belong to each of the categories as well as their distribution of *key* and *non-key* samples. The findings were that a vast majority of the samples belonged to a smaller subset of the clients, topics and industries, while other ones had much fewer samples. Since this was the case, subsequent solution(s) presented to the keyword classification problem could risk being quite adept at correctly classifying samples from some types of clients, and less so on others.

Sample distribution for the clients

As presented in Section 3.2.1, there are in total five clients in the data set. In Figure 1, the distribution of samples among the five clients can be seen, where the orange bars represent the clients' non-key samples and the blue bars represent the key ones. As is easily spotted, most samples come from Client 1, followed by Client 3 and Client 2.

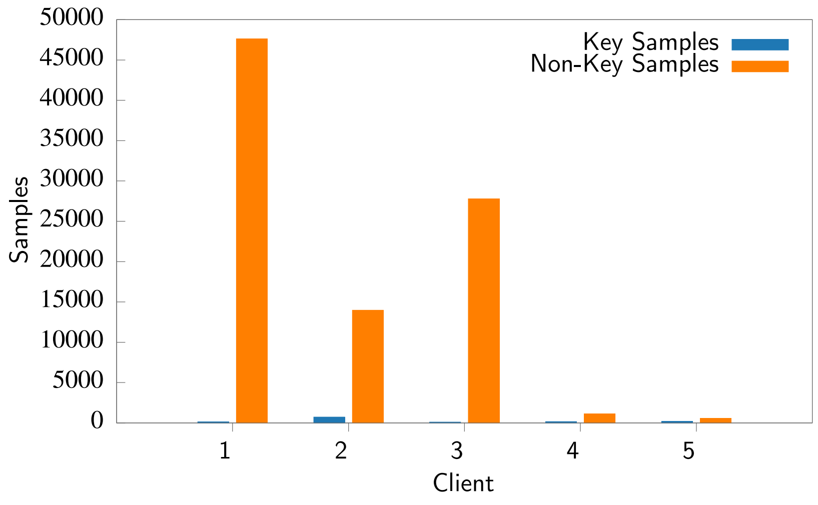


Figure 1: Number of key and non-key samples for each client.

In Table 3, the exact number of key and non-key samples for each client is listed along with the resulting percentage of key samples over total samples. As can be noted when comparing Client 1 to Client 2, more total samples does not necessarily equate to more key samples. Observing Client 4 and 5, it can also be noted that quite a large percentage of their total samples are marked as key. Most probably, this depends on the size of their websites and how much text they contain, which of course is where all the scraped samples come from.

Table 3: Distribution of key and non-key samples among the five clients.

Client	Non-key Samples	Key Samples	% of Key Samples
Client 1	47621	129	0.27%
Client 2	13968	727	4.9%
Client 3	27784	95	0.34%
Client 4	1118	155	12%
Client 5	546	192	26%

Sample distribution for the topics

Similarly to the samples of the clients, the distributions of the industries and topics will here be presented. Since some clients share the same industries and topics, the distributions will shift compared to the findings in Figure 1.

In Figure 2, the number of key and non-key samples are displayed for each topic. As can be observed, clothing and hair product samples make up the vast majority of samples in the data set.

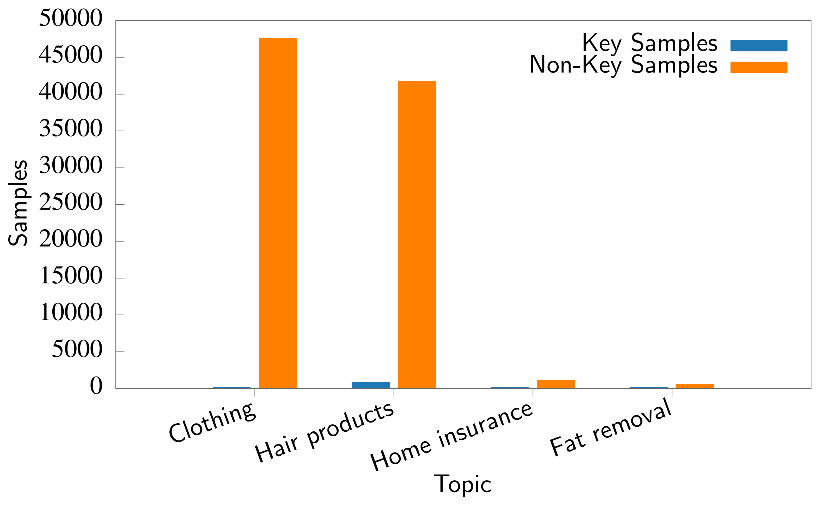


Figure 2: Number of key and non-key samples for each topic.

In Table 4 the number of key and non-key samples as well as the resulting percentage of key samples for each topic is shown. It is clear to see that most of the key samples come from the hair products topic, despite the fact that the clothing topic contains more total samples.

Table 4: Distribution of key and non-key samples among the four topics.

Topic	Non-key Samples	Key Samples	% of Key Samples
Clothing	47621	129	0.27%
Hair products	41752	822	1.9%
Home insurance	1118	155	12%
Fat removal	546	192	26%

Sample Distribution for the industries

Finally, there is the industry category. Here there are 3 categories in total, where Client 1,2 and 3 belong to the retail industry, Client 4 to the insurance industry and Client 5 to the beauty industry. Figure 3 shows the distribution of key and non-key samples among the three industries and, as can be observed, practically all samples belong to the retail industry while only a fraction belongs to the insurance and beauty industry.

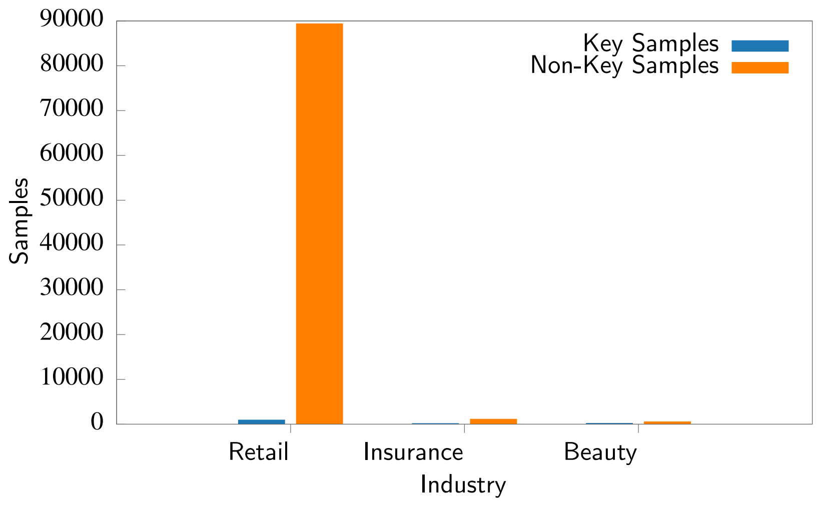


Figure 3: Distribution of key and non-key samples among the three industries.

When observing the findings in Table 5, it can be seen that most of the key samples belong to the retail industry. However, the percentage of key samples in relation to total samples of about 1.1% is the lowest for this industry. All in all, seeing how large a proportion of the samples belong to the retail industry, it should be pointed out that solutions and models trained upon this data run the risk of overfitting for this type of industry and thus not generalizing very well when it comes to other types of data.

Table 5: Number of key and non-key samples for each industry.

Industry	**Non-key Samples**	**Key Samples**	**% of Key Samples**
Retail	89373	951	1.1%
Insurance	1118	155	12%
Beauty	546	192	26%

3.4.2 Size and length distribution of the phrases

The most central part of the whole thesis revolves around the words and phrases that are represented in the *keyword or phrase* variable, which is most probably the main variable that the entire classification depends on and boils down to. Therefore, some distributional information about these words and phrases will be introduced in this section. In the total data set there are, as previously mentioned, about 92,000 samples. That means that there are 92,000 words and phrases that are available for training and testing models. Naturally, the words and phrases differ quite a bit in amount of characters and in amount of words. To begin this section, we define *phrase length* as how many characters are in a sample's keyword or phrase variable, and *phrase size* as how many space separated words are in a sample's keyword or phrase variable.

Length of the phrases

In Figure 4 the phrase lengths for the key samples are represented by the blue bars and those of the non-key samples by the orange bars. As is evident from viewing the plot, the length distributions for the key and non-key phrases differ somewhat from each other, and the key phrases show a tendency to have more characters than non-key phrases.

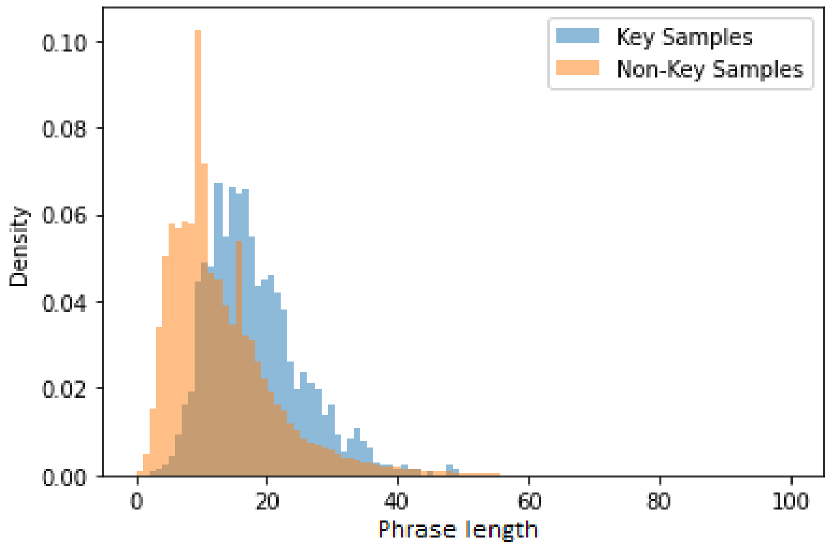


Figure 4: Length distribution of the keyword or phrase variable for key and non-key samples.

In addition to the histogram in Figure 4, some stats about the two distributions are displayed in Table 6. There it should be noted that there are phrases with 0 length, i.e., empty phrases. Naturally, this does not make sense and it indicates that the data set contains flawed samples. Looking at the key samples, the smallest phras length in the available data set is three, which does make some intuitive sense as it is hard to think of a word shorter than three characters which could be a keyword for a company's webpage. Furthermore, the maximum length of a key phrase is 108, which indicates that the final model will most likely need to be able to handle and accurately classify longer strings. Finally, the mean length of the key and non-key phrases are 18.62 and 13.45 respectively. As was noted from the histogram in Figure 4, the key phrases seem to generally be a bit longer than non key samples.

Table 6: Summarizing stats about the length of the key and non-key phrases.

	Non-key phrases	**Key phrases**
Maximum length	74	108
Minimum length	0	3
Mean length	13.45	18.62

Size of the phrases

Apart from the phrases' length, the size of them can also give good insight to the problem that a solution needs to tackle. In Figure 5, a histogram of the sizes of the key and non-key phrases is shown. Though not as obvious as the difference in length, it can here be noted that the key phrases seem to display bigger sizes than the non-key phrases overall, and that the bulk of the key samples seem to contain phrases of either one or two space separated words.

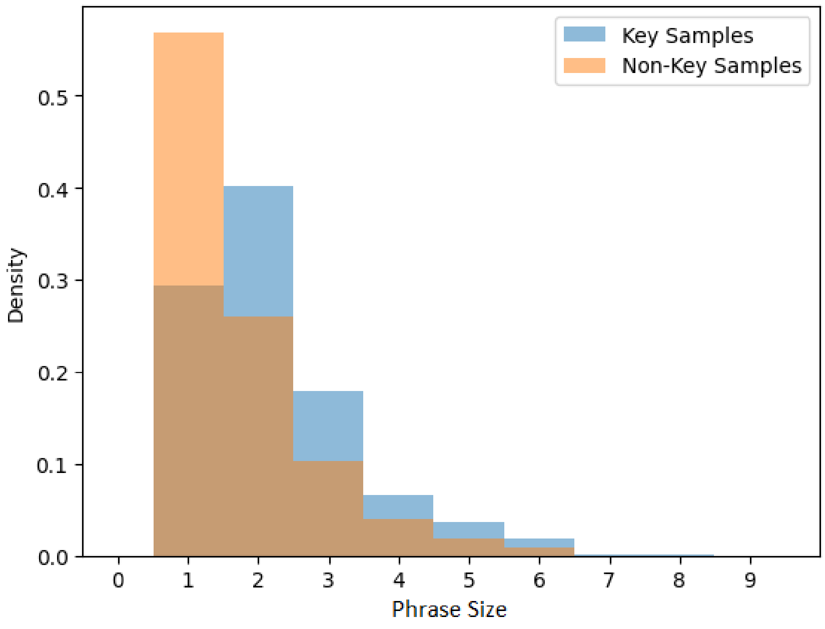


Figure 5: Size distribution of the keyword or phrase variable for key and non-key samples.

To compliment the histogram, some summarizing stats about the size of the key and non-key phrases is presented in Table 7. Inspecting the maximum sizes of the key and non-key phrases, it is evident that the data set consists of quite large strings in terms of size. A solution to the keyword classification problem in this data context thus needs to be able to handle and understand multiple words strings to an effective degree. Furthermore, identifying the same problem as in Section 3.4.2, it is clear from inspecting the minimum size of the phrases that there are some empty phrases. Finally, it can also be noted that the average size of the key phrases is slightly larger than that of the non-key phrases, which was also visible in the histogram in Figure 5.

Table 7: Summarizing stats about the size of the key and non-key phrases.

	Non-key phrases	**Key phrases**
Maximum size	9	9
Minimum size	0	1
Mean size	1.71	2.22

3.4.3 Sample frequency

A final relevant feature about the samples can be found in the *competitor frequency* variable. To reiterate, this feature contains the frequencies of a word or phrase for the client and its named competitors. To use an example from the initial view of the data set in Table 2, the word “dagars” had the frequency of 2 for Competitor 1’s website and 11 for Client 1’s website, which results in an average frequency of 6.5. As such, this section will investigate the average frequencies of all clients. In Figure 6, the average frequencies of key and non-key samples for all clients are presented in a joint plot. Each subplot represents the frequency distributions of a specific client, with the key samples represented by blue and non-key samples by orange. On the x-axises are the average frequencies, which are calculated through the same process as described with the “dagars” example, and on the y-axises are the densities of the distribution in decimal form. The general idea of the distribution analysis is to investigate the proof of concept that the frequency can be a useful variable in the model building stage of the thesis project.

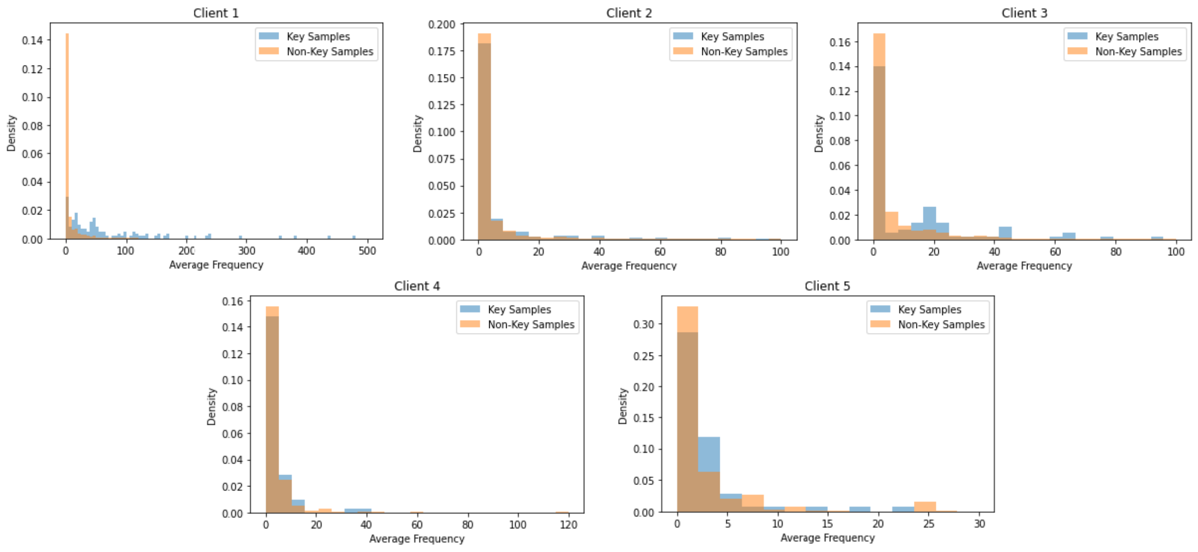


Figure 6: Average frequency distribution of each client’s key and non-key samples.

Judging from the plots of the five different clients in Figure 6, it can be noted that the density distributions of frequency are quite varying when it comes to key and non-key samples. For Client 1, for example, there seems to be quite a big difference between the key and non-key samples, where the key ones seem to display a higher average frequency. A similar conclusion could probably be drawn from inspecting Client 3’s plot, while the rest seem to display quite similar distributions for the key and non-key samples. Worth to note here, however, is that Client 1, 2 and 3 have substantially larger sample sizes than Client 4 and 5, which of course could be a contributing factor as to why these clients do not display the similar distributional behavior. Also worth noting here is that about 39% of Client 4’s samples had no values for the *competitor frequency* variable, and for client 5 that percentage was 28%. These samples with missing values for competitor frequency for Client 4 and 5 naturally had to be cut from the plots in figure 6, which further decreased their already small sample sizes.

4 Data cleaning

During the exploratory data analysis it was found that the data as a whole was quite noisy. Furthermore, from manual inspection of the data, it was found that many words and phrases in the samples were either not real words whatsoever or were simply words that are spelled incorrectly. This was likely caused by the NLP scraping phase where the procedure has split the text in an incorrect manner. If these bad samples were kept, the machine learning model would be trained on those words and phrases, which would probably change the performance of the model for the worse. For this reason, the choice was made to assign some removal rules for the samples in the data. The set of rules are individually presented in Section 4.1, and each of them was constructed as to remove or otherwise handle said troublesome samples.

4.1 Rules for removing bad samples

4.1.1 Phrases with incorrect spacing

The first rule set was not a removal but instead a change of the word or phrase with the keyword or phrase variable itself. The splitting of words in Annalect's scraping procedure seems to have left some samples to begin or end with a spacing. These samples have been *stripped*, which implies removing spacing on the edges of a string. Furthermore, it was found that spacing between words in a phrase could be more than one, and such spacing was changed into one space instead.

4.1.2 Phrases with digits

Many samples in the dataset included digits, which as is, is not a problem. However, what was found is that some samples contained words where there was no spacing between digits and the words, e.g samples like *93chair* or *1hes7sa*. Some of these were likely extracted texts from HTML code on the scraped website, which *can* have such elements while the visual website does not. To avoid including similar samples in the model, a removal of all samples where the word/phrase include digits with no space between letters(text) was made. While the rule removed most bad samples that included digits, it was found that some passed by, such as *info 347523457*. To combat this, another rule was made. There, if the number of digits consisted of more than 50 percent of the total elements in the string, the sample was removed.

4.1.3 Short phrases

As a keyword or keyphrase is meant to include relevant information, there is somewhat of a limitation as to how short a keyphrase can be to be. The rule was thus set that if a sample is shorter than 3 elements in total, the sample was removed.

4.1.4 Consecutive letters in phrase

Another finding was that there were samples whose keyword or phrase variable had the same letter consecutively more than 2 times, which is not allowed in the Swedish language. Thus a rule for removing samples containing this pattern was made.

4.1.5 Phrases containing units

From the EDA, it was also found that some word and phrases samples included units, specifically currency units. It was assumed that phrases that included currencies should not be considered a keyword and therefore these were also removed. The specific terms removed from the dataset are: *kr, sek* and *kronor*. Furthermore, there were phrases that included other units, for example: *ml, kilo, gram*, but these were ignored as it could not be ruled out that they could be seen as key for certain customers and industries.

4.1.6 Competitor name in phrases

Another rule made, which was more of a preference than the other rules set, was that if a phrase included the clients competitor name within it, it should be disregarded. The reasoning for this was that an actual keyword or key phrase should represent information relevant to the client, not a competitor.

4.1.7 Specific elements & words in phrases

Another finding made from the EDA was that there were many phrases in the dataset containing for example *http* and *kundservice*. Words such as these occur on most websites and do not hold any key information. If said specified substrings were found in a sample, then the sample was removed. Additionally, since thesis scope was limited to mainly Swedish, a removal of non Swedish letters such as *ü,ó,ø,æ* was performed.

4.1.8 Phrases starting with single element

A final rule for the removal stage was to remove any word/phrase that begins with only one element followed by a space. This rule was set as it turned out there were multiple samples that followed this layout, which probably is because the NLP model made a bad split when converting the text from the websites into samples. An example of this is *3 bilstol*. By qualitive analysis of these samples, there seemed to be no indication of such samples being defined as a keywords.

5 Handling the topic and industry variables

5.1 String representation

After having removed the noisy and “invalid” words through applying the rules introduced in Sections 4.1.1 - 4.1.8, the data cleaning stage was completed and usable features could be extracted from the data. Some features of numerical nature such as the phrase length and size, both introduced in the exploratory data analysis, could easily be extracted and implemented into a machine learning model. However, the string represented data in the keyword or phrase column as well as in the topic and industry columns posed a significant challenge that could not be addressed in the same manner. As was described in the literature review section, word embeddings are widely used for keyword extraction and is proven to be an effective approach to the problem, as it is a reliable and well-established method for representing words as vectors in a numerical space. The usefulness of looking at words as numerical vectors using a word embedding model can be inspected in Figure 7. The figure represents a two-dimensional example of words in a word embedding model, and each dot represents a specific word.

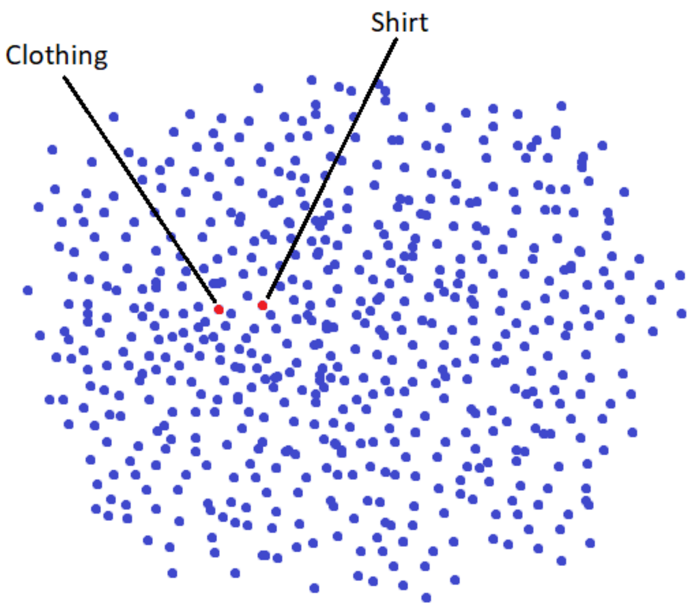


Figure 7: Two-dimensional representation of a word embedding examplification.

In the Figure 7, the words "clothing" and "shirt" are marked out. In this simple two-dimensional example, "clothing" could have a vector representation of $(-0.30, 0.10)$ and "shirt" could be represented by a vector $(-0.20, 0.12)$. Since these words are related, they position themselves closely in the embedding space. As such, word embeddings represent a possibility to calculate the nearness or relation between any two words. In our specific problem, this introduced the possibility to measure the nearness or relation between our phrases and their respective industries and topics. Given this, the choice was made to move forward with word embedding models as a way to numerically represent the words and phrases in our data.

5.1.1 Selecting word embedding model

With the decision made to use a word embedding model to tackle the string represented features, the next step was to decide upon which embedding model to use. Naturally, the availability and overall quality of English embedding models are better than those of Swedish variants. That being said, translating the scraped data into English to be able to use an English embedding model was deemed a worse alternative than using a Swedish one as it would risk losing sub contextual meaning in the samples or outright getting a wrong translation, while also being more computationally expensive. For that reason the choice was made to use a Swedish word embedding model.

When it came to the question of whether or not to use a pre-trained embedding model, the benefits of training an own embedding model were deemed small in relation to the time and computational power it would take to do so. Also, a well trained general use embedding model is likely to perform fairly well, as was shown in the comparison of models by Ramasubramanian et al. (2020) presented in Section 2.3. For these reasons, it was decided to use a pre-trained general use embedding model. Initially, the search for a suitable word embedding for the objective at hand proved to be a challenge as no clear winner emerged. Ultimately, however, two best potential embeddings were identified: one developed by the Nordic Language Processing Laboratory (NLPL, 2017) and the other by spaCy (spaCy, 2023). The spaCy model had a smaller vocabulary of approximately 200,000 words compared to NLPL's more extensive vocabulary of over 3,000,000 words. However, since the Swedish language is comprised of only around 150,000 words, it is evident that the NLPL model has been trained also on unconventional and peculiar words. Summarily, the NLPL model is broader but also runs the risk of recognizing and having a mapping for "invalid" words, while the spaCy model is smaller but probably does not recognize such peculiar words. From here, the choice was made to move forward with the NLPL model. This choice was rooted in the fact that generalization was one of the main goals for the final model, and the ability to recognize as many words as possible would likely aid in this endeavor. Moreover, the data cleaning stage was designed to remove as many peculiar and invalid words as possible, which would decrease the risk of such words being accepted into the ultimate machine learning solution.

5.2 Applying selected word embedding model

After the embedding model was chosen, it was time for it to be implemented on the string represented features in order to extract numerical representations for each of them. Before this could be done, however, a final sort of cleaning of the data had to be made. Although the embedding model represents a good way of tackling the problem of string represented features, it does also present a limitation to the implemented solution, namely the risk that some words simply do not exist in the embedding. This risk was minimized by choosing a large embedding model with a large vocabulary, but the fact remains that some words in the data, probably mostly peculiar or non-real words, do not have a vector representation in the chosen embedding space. Due to this reality, the choice was made to remove all samples containing words that do not exist within the chosen NLPL embedding model. This was done since samples with words that are not recognized simply cannot be fully interpreted by the word embedding model. Furthermore, if a sample contained words that were not recognized by the quite large embedding model chosen, the assumption was made that these samples had a very low chance of being keywords anyways. Subsequently, the removal of such samples would finally leave a clean and usable data set for the subsequent parts of the thesis work, leaving only samples where all words are recognized and have a vector representation in the NLPL model.

The process of filtering out the samples containing words that are not within the NLPL embedding model and thus only keeping the fully represented samples is summarized by the three following steps:

1. Check if all words in a phrase are in the word embedding. If yes, keep that sample.
2. Check if all words in a phrase are in the word embedding after splitting compound words into separated words. If yes, keep that sample.
3. Check if all words in a phrase are in the embedding after translating its words from Swedish to English, separating the words and then translating them back individually to Swedish. If yes, keep that sample.

These three steps were the chosen implementation of keeping as many of the samples as possible, while still making sure that they have a valid vector representation in the chosen embedding model. The first step is quite straightforward, and simply checks if all words in a sample's keyword or phrase variable exist in the embedding model. If that is the case, then the sample is kept. The second step is a bit more intricate, and splits the compound words in the keyword or phrase variable and then checks if all words are in the embedding model. Again, if that is the case, then the sample is kept. The third and final step is the most intricate one, which essentially makes use of Googles translation API to split the most troublesome words in the samples which were not caught in the second step. Here, the keyword or phrase column is translated for the remaining samples from Swedish to English, split and then finally translated back to Swedish. To use an example, this third and final step would translate “mammakläder” to “maternity clothes”. This sample would then be separated to “maternity, clothes”, and when it is finally translated back to Swedish it would be “moderskap, kläder”. If all words in a sample exist in the embedding model after the performed split, the sample is kept.

After the three steps described above were performed, the data only contained samples where each word in the keyword or phrase variable had a vector representation in the chosen NLPL embedding model. Now, features applicable to a machine learning model could be extracted from the data, leading into the feature extraction part of the thesis project.

6 Feature extraction

6.1 Summary of the extracted features

After cleaning the data and confirming that each sample existed in the selected word embedding, the feature extraction process could begin. It was the work leading up to this moment in the project that was the most time-consuming, as the quality of the data was crucial to generating relevant new features and training a model. The extracted features from the data can be summarized as:

- Term frequency
- Phrase size
- Phrase length
- Word Similarity
  - Industry to Phrase
  - Topic to Phrase
  - Subject to Phrase
- Part of speech
  - Is Noun
  - Is Adjective-Noun
  - Is Noun-Noun
  - Is Verb-Noun
  - Is Other
  - Pattern
  - Ordinal

6.2 Reasoning and calculation for each feature

6.2.1 Term frequency

As mentioned previously, term frequency was the only directly usable feature that was more or less given in the data set. There are however multiple ways to measure the relevancy of a word within a text, such as term frequency (TF), inverse document frequency (IDF) and a combination of those being term frequency-inverse document frequency (TF-IDF). While researching, it was found that the most used metric to calculate the importance of a word within a document was TF-IDF. Despite this, as the data provided did not allow for it, TF-IDF could not be used. Furthermore, IDF as a metric is a weighting scheme used to evaluate the importance of a term in a document. It lowers the weight for terms that appear frequently in a document or corpus and gives a higher weight to terms that appear infrequently. The metric itself is calculated by dividing the total number of documents in the corpus by the number of documents containing the term and taking the logarithm of the result. Since the data provided did not include the information needed for this, such as total amount of websites scraped, it was not possible to calculate. Subsequently, it was decided to instead only use the term frequency.

To calculate the term frequency, some assumptions were needed, however. Normally, term frequency is taken from a single text corpus, but this one is not. One idea was to set different weights based on if the term frequency was from the client or its competitors as it was believed that the client's frequency would be better adjusted for the client itself. While this might had been a better cause of action, no evidence from previous works nor a solution to what weights should be set was found. Therefore, it was settled to simply combine all term frequencies from the *competitor frequency* variable into one. As mentioned in Section 3.4.3, there were samples that had empty term frequencies, which are handled by setting them equal to 1.

6.2.2 Phrase size

As mentioned by Hulth (2003) in her work, 90 percent of all key-words/phrases were less than 3 words in total. With this in mind, it was believed that the phrase size would help classifying the samples. However, the paper by Hulth is based on classifying keywords for text summation and not meta tagging, which do not necessarily need to be classified in a similar manner. Furthermore, as a main objective of the thesis is to avoid misclassifying key-words or key phrases as non-key, it would not be logical to outright remove phrases longer than trigrams. Instead the feature variant adopted is to calculate the amount of words within each samples *keyword or phrase* string. A word was thus defined as a letter or number that is separated by space, if not only being one single word by having zero spaces.

6.2.3 Phrase length

This feature is quite similar to phrase size, both the motivation for using it and what it measures. It represents the total length of the *keyword or phrase* variables string as an integer. The reasoning to have both phrase length and size is that they should complement each other well when explaining the string composition, as the size does not directly translate to a specific length of word and neither does the length directly translate to a specific size. As had been seen in the data, there are for example multiple compounded words which would reduce the phrase size while increasing the phrase length. With this in mind, the two features will probably be quite correlated, but also likely complementary.

6.2.4 Word similarity

Word similarity was one of the most used features when creating a keyword classification model. There are different ways of measuring such similarity, for example Rezaei et al. used the *Wu and Pulmer measure*. More often, however, *cosine similarity* is used. Since the *Wu and Pulmer measure* is based on WordNet, which has a tree structural behavior that other word embeddings do not have, other word embeddings are unable to use that metric. For our thesis work, this was the case as the word embedding by NLPL is not built in a similar manner as WordNet. Instead, *cosine similarity* was used for the work in this thesis project, which from research seemed to be a more broadly used metric for word similarity measures within keyword classification.

Cosine similarity

Cosine similarity is a widely used measure of similarity between two non-zero vectors in a high-dimensional space (Xia et al., 2015). In the context of text analysis, it is a popular method to compare the similarity of text segments or singular words based on their feature vectors in the corresponding word embedding. What cosine similarity does is that it measures the cosine of the angle between two vectors, which can be interpreted as the degree of similarity between them. The similarity score ranges from -1 to 1, where a score of 1 indicates that the two vectors are identical, 0 indicates that they are orthogonal (i.e., not similar), and -1 indicates that they are completely dissimilar. The formula for cosine similarity is shown below in equation 2 (Xia et al., 2015).

$$\text{similarity}(\mathbf{A},\mathbf{B}) = \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n}(A_i)^2}\sqrt{\sum\limits_{i=1}^{n}(B_i)^2}} \tag{2}$$

where **A** and **B**, in our case, are the vector representation of two words in the embedding space.

Cosine distance

For this thesis work, the choice was made to use the *cosine distance* to measure word similarity. Conceptually, cosine distance and cosine similarity represent the same thing. The difference is that cosine distance produces values ranging from 0 - 2 instead of -1 - 1 (Gensim, 2022), following as the formula for cosine distance is:

$$distance(\mathbf{A},\mathbf{B}) = 1 - similarity(\mathbf{A},\mathbf{B}) \tag{3}$$

Frankly, the two concepts could be used almost interchangeably, but we preferred to display the similarity as a range of 0 to 2. With this choice, the interpretation is that a small value represented that two words are similar and a larger that they are different.

Industry and topic distance

As previously mentioned, the data provided both an industry and a topic category for each client. For each sample, the cosine distance was calculated between all its words and its topic and industry categories. From these, the lowest value for the topic cosine distance and industry cosine distance were extracted as the *topic distance* and *industry distance*. Below is an example of how the calculation was executed for each sample.

1. Extract the information of the sample
   - Industry: retail
   - Topic: clothes
   - Keyword or phrase: liten blommig tröja
2. Translate the industry and topic strings into Swedish
   - Industry: detaljhandeln
   - Topic: kläder
3. For each word in the keyword or phrase string, the cosine distance was calculated for both the topic and industry
   - Industry distance = [0.7, 0.5, 0.1]
   - Topic distance = [0.6, 0.35, 0.08]
4. The lowest value was then extracted
   - Industry distance = min([0.7, 0.5, 0.1]) = 0.1
   - Topic distance = min([0.6, 0.35, 0.08]) = 0.08

Subject distance

This feature is very similar to both Topic and Industry distance, where it calculates a cosine distance from the client's *subject*. The big difference with this feature is that there was no subject variable provided in the data given, but it was instead something that was extracted from the data by clustering all nouns from the *keyword or phrase* variable. The nouns were extracted using the SpaCy model,previously mentioned in Section 5, and not the NLPL model. The reasoning for this choice was simply that the NLPL model used did not have part of speech information included, which naturally was needed to inquire whether a word was noun or not. Furthermore, the choice to only use nouns when extracting subjects from the samples was based on Rezaei's, Gali's and Fränti's and Hulth's papers where they used similar methods of finding keywords themselves.

Furthermore, Rezaei et al. used hierarchical clustering in their implementation. However, the implementations of hierarchical clustering in python do not have an *automatic* threshold, causing it to be more useful as an analytical approach than as a pipeline approach. To circumvent this, another clustering method named Density-Based Spatial Clustering of Applications with Noise (DBSCAN) was used instead. The method is based on density, where a cluster is created based on a minimum density level (Schubert et al., 2017). The density level is set via a threshold using a minimum set of neighbours, a radius ϵ and *minPts*. *minPts* is a secondary threshold, where datapoints that have more than *minPts* neighbours within the radius set will be defined as *core points.* To be included within a cluster, a point needs to be within the radius set to any *core point.* Any point that does not reach any *core point* are considered as *noise.*

With the attributes of DBSCAN in mind, by setting the ruling very narrow and iteratively widen it, the first, and probably best, cluster could be achieved. The points themselves are from the word embedding, where it was possible to retrieve the 100 dimensional vector representing each word, which in this case were nouns. Then, once the clustering was made, the points were traced back to what words they represented. Once the nouns were found through the clusters, their term frequencies were calculated by finding all the strings from *keyword or phrase* variable containing them and sum them up. Then, the nouns corresponding to the five highest term frequencies were extracted. These five nouns are then what will be defined as the client's *subjects* and be used to calculate the cosine distance. The reason for taking five subjects and not one was based on findings that the noun with highest term frequency did not necessarily have to be the most relevant to the client. To see how the subject distance was calculated for each sample, consider the example below:

1. Extract the information of the sample
   - Subjects: skor, blus, klänning, jeans, halsband
   - Keyword or phrase: liten blommig tröja
2. For each word in the keyword or phrase string, the cosine distance is calculated with all subjects
   - Skor distance = [0.7, 0.6, 0.4]
   - Blus distance = [0.7, 0.3, 0.2]
   - Klänning distance = [0.7, 0.4, 0.3]
   - Jeans distance = [0.7, 0.6, 0.4]
   - Halsband distance = [0.7, 0.6, 0.5]
3. The lowest value out of all subject distances was extracted
   - Subject distance = min([0.7, 0.6, 0.4],[0.7, 0.3, 0.2],[0.7, 0.4, 0.3],[0.7, 0.6, 0.4],[0.7, 0.6, 0.5]) = 0.2

As a human, it is understandable that all these words are related to both detaljhandel and kläder while none of them match all types of clothing. Because of this, the approach used was chosen to have multiple subjects in hopes of one being the correct subject for each sample string. Similarly as with the topic and industry distance features, each word of a string given a sample of *keyword or phrase* variable, the cosine distance was calulated, but this time with all five of the subjects. Then the minimum of the set of cosine distance values were taken. Using this approach, the optimal subject-word combination from a sample was achieved and set to the *subject distance* feature. Which of the subjects used for the specific sample is not relevant, only that the one having the lowest cosine distance is used.

6.2.5 Part of speech

Part of speech (POS) is a linguistic term that refers to the grammatical category of a word, such as noun, verb, adjective, or adverb. The identification of POS provides information about the syntactic and semantic properties of the word, which can help in understanding the context in which it is used. With this in mind, a POS feature should help improve a keyword classification model. In addition, POS tagging can be used to extract specific types of words, such as verbs or adjectives, something that both Rezaei, Gali and Fränti and Hulth did in their papers for specifically nouns.

Part of speech pattern

One of the POS features that was created is based on Hulth's paper where one of the approaches used was specifically based on POS patterns that had the most assigned keywords. Hulth sets a threshold that a specific pattern needs a minimum of 10 keywords assigned to it to be used as a feature, if not it would be assigned *POS other*. The initial thought was to one hot encode the POS pattern to capture as much information as possible, but once it was noticed that there were more than 2300 unique patterns, it was decided to not a be feasible option. Instead, a similar approach as Hulth was used but the threshold was adjusted from a minimum of 10 of keywords assigned to a minimum of 50, which seemed to be more fitting for our data. This resulted in the following POS patterns as features:

- NOUN
- ADJ NOUN
- NOUN NOUN
- VERB NOUN
- OTHER

Has part of speech

To circumvent the problem with there being 2310 POS patterns and a belief that the POS pattern feature did not captures all relevant information that can be used, another feature set was created. This created feature represents if the *keyword or phrase* string contains a specific POS, such as verb and if so it will be set to 1, else 0. It is a sort of one hot encoding but instead of having all 2310 POS patterns, only the unique POS were used and assigned. So, the features within this one-hot encoded features were on the form *has noun*, *has adjective*, and so on, covering all 17 unique POS.

Ordinal part of speech

The one-hot encoded POS made the training data quite sparse, which is undesired. Because of this, an ordinal encoding of the POS patterns was also included ranging from 0 to 2310. The problem with ordinal encoded data is how higher numbers *should* represent a category being higher valued (e.g 1. middle school, 2. high school, 3. university), which is not the case for POS patterns. However, there are machine learning algorithms that do not take this ordinal effect into account, such as tree based ones.

6.2.6 Normalization

To make the features extracted from every client proportional to one another, normalization was applied on the numerical features. The normalization was done individually within each client to make the data more generalized and, again, proportional to each other. As a result, the features that were normalized were all converted into values between 0 and 1. The normalized features were the term frequency as well as the word distance features, namely the industry, topic and subject distances.

7 Modeling prerequisites

Before the modeling stage could begin, there were multiple things to be considered. For instance, choices such as which algorithms to use, how the performance of the model should be measured, which of all the features should be used when training the model and how to partition the data for training and testing, were all valid questions that needed to be answered. The answers and the motivations behind them are discussed in this section.

7.1 Training and test data

The data that was used for the subsequent parts of the thesis work consisted of the samples that made it through the data cleaning while also existing in the word embedding model. All in all, this remaining data consisted of roughly 70,000 samples. From here, the choice of how to split the data for the training and testing of the models needed to be made. It was decided to split the data so that 80 percent of all the samples made up the training set, and the remaining 20 percent made up the test set. However, when making the split to create the training and test data, some considerations had to be made. The first thing was that the data needed to be split individually for each client to make sure that both the training and test set contained proportional data from all clients, especially since they were so different in sample sizes. With this, 80 percent of each client's key samples were put into the training data and 20 percent into the test data, and same for its non-keyword samples. By doing this, both the training and test set would in extension also proportionally represent the different industries, topics and clients. The splitting of training and test data was done by using scikit-learn's train- and test split function (Pedregosa et al., 2011). In essence, the splitting procedure first shuffles all the samples. Then, the training data is extracted by taking the first X percent of the shuffled data, where X represents the percentage to be put in the training data. The remaining samples will be assigned to the test data. For a more in depth explanation of the splitting, we refer to the source code of the function (Buitinck et al., 2013). Once the split was done, we had a training set consisting of about 57,000 samples and a test set of about 14,000 samples, each containing all the 29 features extracted during the feature extraction section.

7.2 Metrics

How to compare model performances was one of the main challenges going into the modeling stage. Since the goal of the project was somewhat vague and consisted of keeping as many of the key samples as possible, while also removing as many of the non-key samples as possible, the choice of model performance metric was sure to be an important one. Furthermore if the choice of metric was not considered carefully, the risk of choosing a model that optimized for something else than intended would be very real. To avoid this, it was possible to apply a specific scoring metric into model evaluation tools. The tools used are described further in Section 7.3 and 7.5.

There are multiple metrics to choose from, but often used metrics for a classification problem are precision, recall and accuracy. Something all of these metrics have in common is however what they are based on, namely the predictions of a model and the true labels. These correctly and incorrectly classified samples can be divided into four classes, often visualized by a confusion matrix. The four classes are: True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). True Positive means that a sample is predicted to be true while also the true label is true (James et al., 2013). The True/False represents the if the prediction is correct or not while Positive/Negative represents what is predicted. With this in mind, the good outcomes are True Positive and True Negative. A visualization of all four classes are shown in Table 8.

Table 8: Example of confusion matrix

		Predicted class	
		True	False
True Class	True	True Positive (TP)	False Positive (FP)
	False	False Negative (FN)	True Negative (TN)

In our case, this would translate into the following Table 9.

Table 9: Example of confusion matrix in our case

		Predicted class	
		True	False
True Class	True	Included keyword	Excluded keyword
	False	Included non-keyword	Excluded non-keyword

Accuracy

As the goal for this thesis is to avoid misclassifying keywords as non keywords, accuracy would not provide relevant information as the performance score would be heavily weighted towards the non-keywords correctly classified since the data is unbalanced towards it. How accuracy is calculated is shown equation 4.

$$Accuracy := \frac{TP + FP}{TP + TN + FP + FN} \tag{4}$$

Precision

Precision on the other hand measures out of all keywords being predicted, how many are *correctly* classified, shown in equation 5. This means it does not consider keywords that would have been misclassified and since Annalect asked specifically to avoid misclassifying keywords as non-keywords, the metric does not align with our goal.

$$Precision := \frac{TP}{TP + FP} \tag{5}$$

Recall

On the contrary, that is what the recall metric does, where it calculates out of how many predicted being true are actually true, shown in equation 6. This would translate for us into how many keywords that are *correctly* classified without consideration of the misclassified non-keywords, which falls more in line in the goal of the thesis. However, optimizing via recall has its drawbacks. In our case, recall would not consider all non-keywords classified as keywords, meaning that if all samples would be classified as keywords, the performance of recall would still indicate a great model due to no keywords are missed.

$$Recall := \frac{TP}{TP + FN} \tag{6}$$

Precision recall trade-off

A reason why neither recall nor precision seemed fitting for us was because they are inversely related to each other. This specific problem is called the *precision-recall trade-off* (Buckland and Gey, 1994). To visualize the trade-off, a precision-recall curve can be made and an example of it is shown in Figure 8, made by Statology (2023).

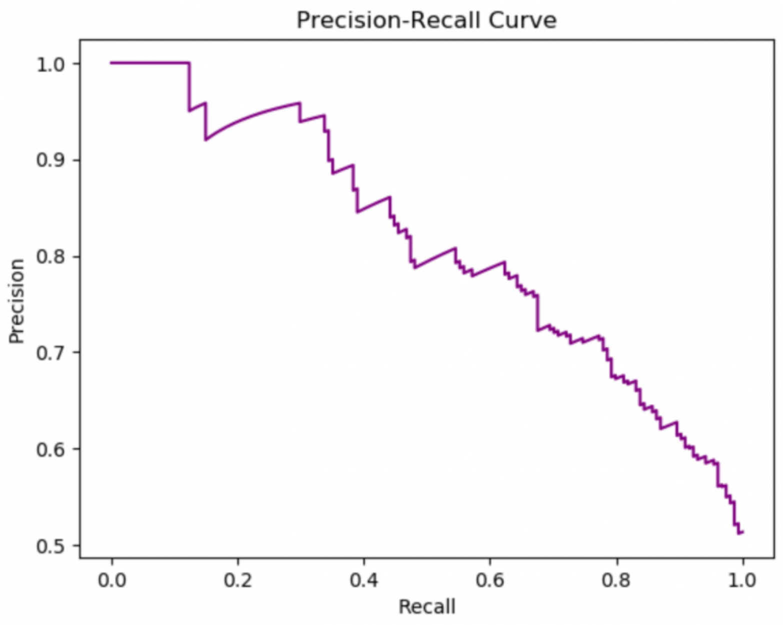


Figure 8: Example of precision/recall curve

F_1 score

To get the benefits of both precision and recall, F_1 score should be considered. The metric is a weighted harmonic mean of precision and recall and inherently is a representation of both metrics (Shalev-Shwartz and Ben-David, 2014). How F_1 score is calculated is shown in equation 7.

$$F_1 := \frac{2 \cdot TP}{TP + FN + FP} \tag{7}$$

F_β score

A more general F-score is F_β score, where a β parameter is included which controls the weighting of recall (Shalev-Shwartz and Ben-David, 2014). The metric makes it possible to adjust how important recall relative to precision should be. When $\beta = 1$, it means that precision and recall are equally weighted, if $\beta < 1$, the metric is weighted towards precision and if $\beta > 1$ then the metric weights more towards recall. The formula of F_β score are shown in equation 8 which is based on precision and recall, shown in equation 5 and 6 respectively.

$$F_\beta := (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{8}$$

7.3 Hyperparameters

In machine learning, hyperparameters are parameters that cannot be learned from training data and are set before training. They are used to control the learning process and can have a significant impact on the performance of a machine learning model. Some examples of hyperparameters in machine learning include learning rate, regularization techniques and strength, number of decision trees in a random forest. They can either be set manually by the practitioner, or can be tuned using optimization techniques such as grid search or random search.

The optimal values for hyperparameters are often problem-specific and can depend on the size and complexity of the dataset, as well as the machine learning algorithm being used. Because of this, it is unlikely that a manual search would find the *optimal* combination of hyperparameters and has therefore opted to use random search in favor of optimizing the results. The reason of using *random search* over *grid search* was based on the computational time difference. Grid search is a brute-force approach where a pre-defined set of hyperparameters are systematically searched over to find the optimal combination of values. In contrast, random search randomly samples a specified number of hyperparameter combinations given from the practitioner from the same pre-defined search space. Therefore one big advantage of random search is that it can efficiently explore a larger hyperparameter space than grid search, especially when the search space is high dimensional as the combinations become exponentially larger. Finally, the random search method can also optimize for a custom metric, such as F_β score. In other words, this means that the random search method returns the model with the hyperparameters that performs best given the custom set metric, e.g., F_β score.

7.4 Implemented machine learning algorithms

With the goal in mind to find a model that predicts low amounts of *false negatives*, i.e incorrectly classifying keywords as non-keywords while still correctly classifying the majority of samples, three different algorithms were implemented and tried. This was important because there is no single algorithm that performs the best on every type of dataset or problem. Different algorithms have their own strengths and weaknesses, and may be more or less appropriate for different types of data and tasks. Therefore, by applying multiple algorithms, one has a better chance to find an algorithm which performs well for a specific problem. For this thesis, the different machine learning algorithms chosen to be tried were Logistic regression, Decision tree and Random forest.

7.4.1 Logistic regression

Logistic regression is a natural choice to implement in a binary classification problem as it is an algorithm which is quite simple and also easy to interpret. Essentially, logistic regression models the probability $p(X) = Pr(Y = 1|X)$, i.e., the probability that given the features X = $[x_1, x_2, ..., x_q]$, the sample class Y is equal to 1 in a two class setting $[0, 1]$ (James et al., 2013). In our case, this would translate to the probability of the label being *key*, given the modeling features provided. To make the prediction return values between 0 and 1, the *logit function* is used. This is a *link function* which models a linear combination of the features provided into a probability, shown in equation 9.

$$logit(p(X)) := \log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \sum_{i=1}^{q} \beta_i x_i \tag{9}$$

Furthermore, maximum likelihood is used to fit the model, i.e., estimate parameters $\beta_0, \ldots, \beta_q$, that predicts said probability $p(X) = Pr(Y = 1|X)$ (James et al., 2013). Another advantage that came with applying the logistic regression algorithm to the keyword classification problem is that since the method solves for a probability, the probability decision limit could be manually changed when using the model for classification (James et al., 2013). In other words, this makes it possible to further control the trade-off between wrongfully classifying keywords as non-key and vice versa. With this reasoning, the logistic regression algorithm was chosen as one of the ways to tackle the keyword classification problem.

Class weight

As there was a very big overrepresentation of the non-key samples as opposed to the key samples, there was a need to handle this problem within the logistic regression modeling framework. This problem was tackled through *balancing* the training data. Weights were assigned to all training samples in order to make the total weight of each class equal. As the data for this problem contained far more non-key samples than key samples, the key samples were assigned a larger weight than the non-key samples, while, the total sum of weights for the key and non-key samples were equal. In other words, we used balanced class weights for the training of the logistic regression model. So, when training the model, the classification of a single key sample will have a larger impact on the model configuration than that of a non-key sample, but in total they have the same impact. For more information about the class weighting, we refer to scikit-learn's documentation (Pedregosa et al., 2011).

7.4.2 Decision tree

The decision tree algorithm is also well suited for binary classification problems. It can handle both numerical and categorical features and is also able to detect nonlinear relationships between feature variables and the response. In addition, it is robust to noisy data. A decision tree is a prediction model that predicts the label of a sample by traveling from the *root node* towards a *leaf node* through *branches*. At each step from root to leaf, there is a *internal node* where the algorithm splits the data into subsets by using the feature which best separates the data into two classes (Shalev-Shwartz and Ben-David, 2014). A visualization of a simple decision tree model is shown in Figure 9, where $X = [x_1, x_2, x_3]$ are the features, and $T = [t_1, t_2, t_3]$ are the values which together create the rules splitting the data. Once the samples have reached a leaf node, they are classified to the class the leaf represents.

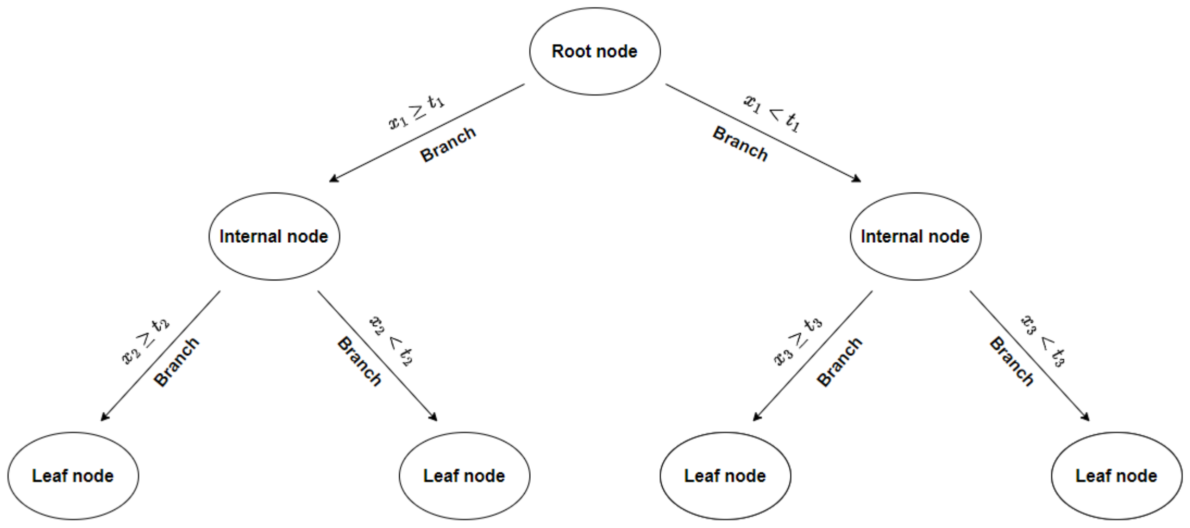


Figure 9: Example of a decision tree model

Notably, this algorithm works very differently from logistic regression and simply put constructs decision rules based on the input features that give the model the best performance, according to some chosen metric. All in all, the decision tree algorithm was deemed interesting and a potentially good fit for the given classification problem, and it was chosen to implement on the given data.

Criterion

There were quite a few hyperparameters that were investigated for the random forest algorithm in this thesis work. The first one was the so called *criterion*, which specifies the function which measures the quality of a split within the decision tree (Pedregosa et al., 2011). Two criterion functions were tried: *Gini index* and *entropy.* The Gini index is the standard loss measure and is defined as:

$$G(Q_m) = \sum_{k=1}^{2} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{10}$$

Furthermore, the entropy is defined as:

$$D(Q_m) = -\sum_{k=1}^{2} \hat{p}_{mk} log(\hat{p}_{mk}) \tag{11}$$

For equation 10 and 11, Q_m is the samples at node m, $\hat{p}_{mk}$ represents the proportion of samples in the mth classification region created by the previous rules in the branch that are from the kth class, in our case being key or non-key (James et al., 2013). These are the two most common approaches to measuring the quality of a split in the decision tree model. For more information about these two criteria we refer to James et al. (2013) and Shalev-Shwartz and Ben-David (2014).

Maximum features

The second hyperparameter examined for the decision tree algorithm was the *max amount of features.* In clear language, this hyperparameter controls the number of features to consider when looking for the best split. The investigated alternatives for this hyperparameter were to set the max features to the total number of features, i.e., no limit, the square root of the total number of features and finally the two base logarithm of the total number of features. Here we refer to scikit-learn's documentation (Pedregosa et al., 2011).

Minimum sample split

Thirdly, the *min sample split* hyperparameter was considered. This controls the minimal amount of samples required to split an internal node, i.e., all nodes except the tree- and leaf nodes.

Minimum sample leaf

The fourth hyperparameter investigated in the decision tree algorithm was the *minimum samples per leaf.* Simply put, this controls the minimum amount of samples required to be at a leaf node. If the splitting of a node would leave an amount of samples less than this minimum value, then the split would not be performed (Pedregosa et al., 2011).

Maximum depth

The fifth and final hyperparameter considered for the decision tree model implementation was the *maximum tree depth.* As the name suggests, this specifies the maximum depth allowed for the decision tree, i.e., how many splits are allowed to be made before making a classification (Pedregosa et al., 2011).

Class weight

As previously stated, the data provided was unbalanced between the two classes. For the decision tree algorithm, this was solved through utilizing the *class weights* hyperparameter and setting it to *balanced.* The implementation of this hyperparameter is quite different from how it is done in a logistic regression algorithm. Typically, when assessing the suitability of a leaf node split, the decision tree algorithm considers the number of samples at the node, evaluating whether the split would improve the impurity using either the Gini index or entropy as a metric. An example of how it is done is shown in equation 12 using the Gini index as impurity measure.

$$\Delta_{G(Q_m)} := \frac{N(Q_m)}{N(Q_{total})} \cdot \left(G(Q_m) - \frac{N(Q_m^{right})}{N(Q_m)} \cdot G(Q_m^{right}) - \frac{N(Q_m^{left})}{N(Q_m)} \cdot G(Q_m^{left}) \right) \tag{12}$$

In equation 12, Q_{total} is the total samples and Q_m is the samples in node m. Moreover, Q_m^{right} and Q_m^{left} are the samples in m's potential new right and left leaf nodes. Finally, N(Q) represents the amount of samples in set Q.

When incorporating class weighting, the algorithm takes into account the class weights assigned to each sample. Each sample's class weight is calculated in the same manner as in the logistic regression algorithm, where the sum of the class weights of the key samples is the same as the sum for the non-key samples. As such, the total weight of a set of samples $S = [s_1, s_2, \cdots, s_j]$ can be described as

$$W(S) = \sum_{i \epsilon S} w_i \tag{13}$$

Now, the potential improvement in the Gini index of a split can be determined using the class weights instead of the raw amount of samples in a leaf node. This adjustment changes the calculation shown in equation 12 by replacing the sample amounts by the their corresponding weights, as shown in equation 14.

$$\Delta_{G(Q_m)} := \frac{W(Q_m)}{W(Q_{total})} \cdot \left(G(Q_m) - \frac{W(Q_m^{right})}{W(Q_m)} \cdot G(Q_m^{right}) - \frac{W(Q_m^{left})}{W(Q_m)} \cdot G(Q_m^{left}) \right) \tag{14}$$

For a more in depth explanation of the conditions to perform a node split, we refer to the source code of scikit-learn (Buitinck et al., 2013). Noteworthy is of course that equations 12 and 14 are using the Gini index metric to measure impurity. Naturally, given that $G(Q_m)$ is a loss function, the algorithm is trying to maximize $\Delta_{G(Q_m)}$, since the difference in the current impurity versus the potential impurities should be as high as possible. To find the $\Delta_{D(Q_m)}$, i.e., the delta impurity using entropy, the three impurity measures are instead calculated using entropy shown in equation 11.

7.4.3 Random forest

The final model that was chosen to be extensively investigated and tried on keyword classification problem setting was the random forest algorithm. In essence, the random forest algorithm is an extension to decision trees and, more specifically, an aggregation of multiple decision trees (James et al., 2013). The multiple decision trees, together making up the forest, are then combined to make consensus decisions on new samples. The key aspect that differentiate the trees apart are that each tree is using bootstrapped training samples (James et al., 2013). Bootstrapped sampling is a resampling method that creates multiple datasets of a single dataset. This is done by randomly picking samples from the original dataset until the same size of the original dataset is achieved. Here, a sample can be picked multiple times, meaning that the sampling is done with replacement. The combination of bootstrapped sampling and aggregated decisions is called *Bagging* and a visualization of it can be seen in Figure 10.

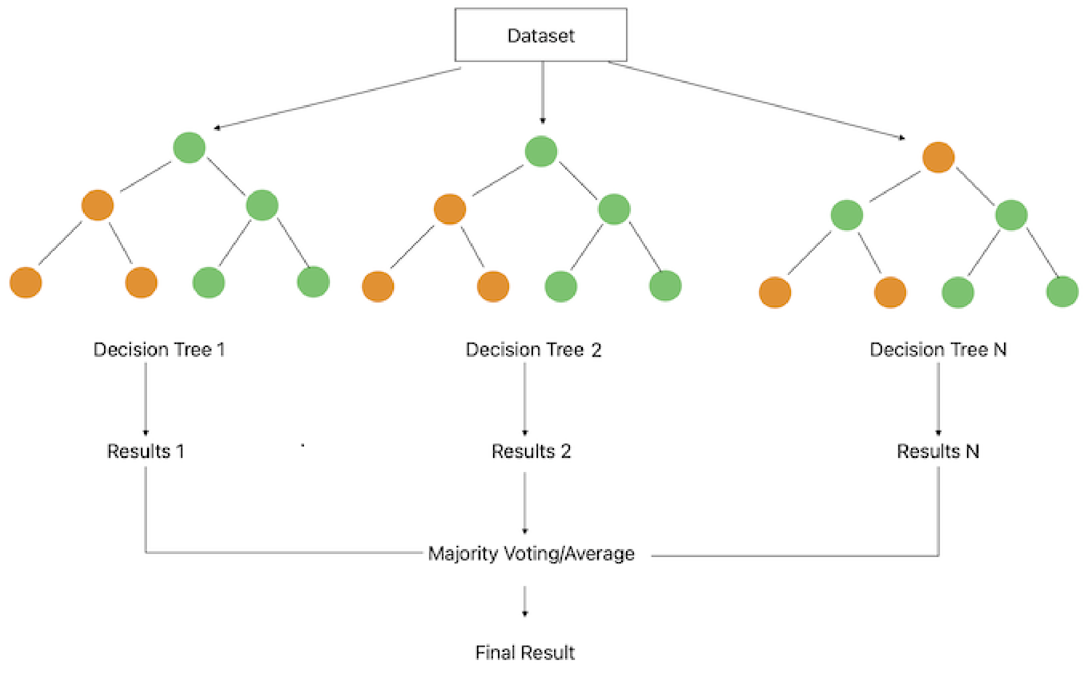


Figure 10: Example of a random forest model (Tutorialspoint, 2023)

In comparison to creating a single tree according to the decision tree algorithm, a random forest model can often result in quite large improvements in prediction performance (James et al., 2013). This is mainly due to the fact that random forest, as it is an ensemble method, is less prone to overfitting compared to decision trees. For these reasons, the choice was made to also apply the random forest algorithm to the keyword classification problem.

Hyperparameters

Since a random forest is constructed of numerous decision trees, the hyperparameter available hyperparameters to optimize for are the same as for the decision tree algorithm. As such, the choice was made to use the same set and range of hyperparameters for the random forest algorithm as for the decision tree algorithm. The reasoning is that, if the algorithms would have used different hyperparameters, then it could have caused bias towards one of the algorithms as it would then be unknown which hyperparameter was having the largest effect on the model. However, there was one single hyperparameter that only random forest utilized between the two algorithms, namely the *number of estimators* hyperparameter. This hyperparameter specifies how many individual trees are used in the random forest (Pedregosa et al., 2011). Here, the choice was made to use 100 to build up the random forest model. Finally, as with the logistic regression and decision tree algorithms, the problem of skewed data was solved through using hyperparameter *class weight* and assigning it to be balanced over the two classes. The implementation are identical to the one used in decision tree.

7.5 Feature selection

Performing feature selection is an important step in machine learning modeling where the objective is to select a subset of relevant features to improve model performance. The aim is to select the most informative features that contribute the most to the target variable and remove irrelevant and redundant features that do not provide any significant contribution to the model. Now, since a one hot encoding has been made for all possible part of speech variants, this was likely needed to avoid to much sparsity in the training data.

There are other benefits of performing feature selection also. First, it can help to reduce overfitting, which occurs when a model becomes too complex and starts to fit the noise in the data instead of the signal. Second, it can improve model interpretability by removing unimportant features that can obscure the relationships between the predictors and the target variable. Third, it can improve model efficiency by reducing the computational cost of training the model and making predictions.

As is known by now, the project aim was to achieve a model that caught as many keywords as possible and also excluded as many non-keywords as possible. With this somewhat vague goal in mind, the chosen feature selection method needed to allow for setting a custom metric. This is not possible with most feature selection methods, but one that allowed for this was the wrapper method *Recursive feature elimination with cross-validation* (Pedregosa et al., 2011). The RFECV algorithm works by first fitting the model on the full set of features, and also all feature combinations where $n-1$ features are used, n being the number of features. Then, the performance scores for all model configurations, given a chosen metric by the user, are calculated. From there, the model composition with the best performance score is "chosen". If that is the full model, then no feature is removed. If that is a model configuration without some feature, then that feature is removed and the process is repeated until the model without removing a feature yields the best score. Furthermore, in order to prevent overfitting, a k-fold cross-validation is used at each iteration to evaluate the model performance.

8 Modeling

8.1 Finding Suitable β Value

As was presented in Section 7.2, utilizing the F_β score metric was a key part in finding models that could solve the introduced classification problem. However, as is indicated from the formula for F_β, the choice of β is fundamental. A low value represents weighting towards improving precision, while a larger value represents weighting towards improving recall. As such, identifying a suitable β needed to come before it would be possible to find the final models for each algorithm implementation.

The identification of a suitable β value was done in an iterative manner. For the first iteration, a search of a wide range of β values was conducted. For each value of β in the wide range, multiple models were trained using the random search method to find optimal hyperparameters and RFECV to find the best features to include in the models. For logistic regression, as this had no hyperparameter space, this only consisted of selecting the best feature composition using the F_β metric. After training multiple models with different hyperparameters for each algorithm type, each β value in the wide range then had a corresponding best model, measured by the F_β, with a resulting confusion matrix. From there, the resulting confusion matrices were qualitatively inspected, and a new narrower range was chosen depending on which β values produced the best models in terms of their confusion matrices. The reason for needing to rely on qualitative inspection of the confusion matrices instead of only the F_β performance is because a model with a too high β value would get an almost perfect score if the model predicted all samples as key. Similarly, the F_β metric for a model with a too low β would be almost perfect if very few key samples were classified as key, while all the other samples were classified as non-key.

After the first iteration, where a new and narrower β value range had been found, another iteration with the same procedure was conducted. Furthermore, if a model corresponding to one of the chosen β values in an iteration reached the edge of its hyperparameter space, then the corresponding range of hyperparameter values was widened for the next iteration. Finding the best performing β value took two iterations for the logistic regression model and three iterations for the decision tree and random forest models. The results were that a $\beta = 4$ seemed to perform the best for all algorithms, again, judged from qualitative inspection of the confusion matrices conducted by ourselves and verified by Annalect. For further information and documentation of each iteration, we refer to Appendix A.

8.2 Optimizing models for chosen beta

8.2.1 Logistic regression model

Optimizing the logistic regression model for $\beta = 4$ yielded a model containing the parameters shown in Table 10. As can be seen there, the final logistic regression model contains many features, including all the distance features, the term frequency and also numerous part of speech patterns and individual part of speech features on the *has* form.

Table 10: Features used for logistic regression model

β **value**	**Features used**
4	term frequency, has ADV, has CCONJ, has DET, has INT, has PART, has PRON, has PROPN, has PUNCT, has SYM, industry dist, topic dist, subject dist, noun noun

Furthermore, the logistic regression model with $\beta = 4$ yielded the results on the test data presented in Table 11. As is observable, a large chunk of the non-keywords were removed, specifically 11,019 out of the 13,907. Furthermore, the solution was able to correctly classify and include 212 out of the total 240 keywords in the training. All in all, we find this a quite well performing model with an acceptable result to the given classification problem.

Table 11: Logistic Regression Model Performance

		Predicted class	
		True	False
True Class	True	212 Included keywords	28 Excluded keywords
	False	2,888 Included non-keywords	11,019 Excluded non-keywords

A good thing that can be done using logistic regression models is that the prediction threshold can be adjusted, mention in Section 7.4.1. By moving the threshold in 0.1 increments between 0-1, the following results was made shown in Table 12.

Table 12: Logistic regression results by adjusting prediction threshold

Threshold	**TP**	**FN**	**TN**	**FP**
0	240	0	0	13907
0.1	235	5	5454	8453
0.2	230	10	7876	6031
0.3	227	13	9304	4603
0.4	220	20	10338	3569
0.5	212	28	11019	2888
0.6	203	37	11665	2242
0.7	175	65	12293	1614
0.8	145	95	12931	976
0.9	62	178	13694	213
1	0	240	13907	0

Arguably using threshold equals to 0.6 could be interpret to improve the performance, while our take on it is split. Such threshold did indeed remove more non-keywords while at a cost of losing keywords. The performance trade-off can be argued upon but given the thesis description, we believe it is better. A confusion matrix of the from the best logistic regression model are shown in Table 13.

Table 13: Logistic Regression Model Performance

		Predicted class	
		True	False
True Class	True	203 Included keywords	37 Excluded keywords
	False	2,242 Included non-keywords	11665 Excluded non-keywords

Moreover, the precision-recall trade-off using the different thresholds can be seen in Figure 11.

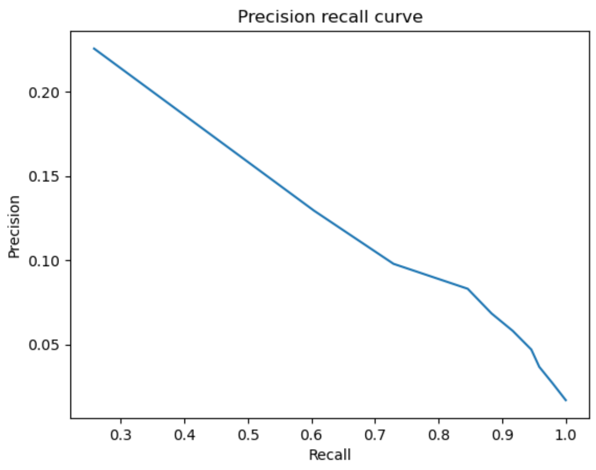


Figure 11: Precision-recall trade-off for thresholds 0-0.9 using logistic regression model

By analyzing the figure, one can that the trade-off is quite linear while the threshold is between 0-0.7 and after that, the recall increases more than the precision decreases between 0.7-0.85 and then returns somewhat to the linear relationship afterwards. What this means, is that the performance between kept keywords in relation to the discarded non-keywords are linear up until 0.7. Therefore, it might seem more sensible to choose the model with threshold 0.7. However, we think there are too many lost keywords for that model, being 65, compared to 37 of the total 240 keywords available when using threshold 0.6.

Furthermore, it is also important to evaluate the model and its solution in its environmental context. With applying this solution to the test data, an SEO specialist would need to sift through roughly 2,300 words of which 203 would be key words. This would be in comparison to Annalects current solution, which would be manually go through about 14,000 samples containing 240 are keywords. We argue that the first alternative is preferable. The reason is that 2,300 samples is a small enough set for an SEO specialist to manually go through more thoroughly, while 14,000 samples are simply too many, and would require the SEO specialists to resort to the method of manually searching for words in the 14,000 samples.

8.2.2 Decision tree model

To find the optimal decision tree model, a grid of all 990 model combinations with different hyperparameters was searched to make sure that no optimal solution was missed. Furthermore, now that an extensive search for an "optimal" model was to be conducted, the *max depth* hyperparameter of the decision tree function was set to "None". This meant that there was no maximum branch depth and that each branch would be allowed to grow without limitation. This was implemented here and not during the β search iterations due to the large computational time and power that it required. Furthermore, since we found that some models in the last iteration of the β search had reached the range limits in the hyperparameters *min sample leaf* and the *min sample split*, these increased to 20 and 30 respectively. A summary of the hyperparameter space searched for the decision tree model is shown in table 14.

Table 14: Decision tree hyperparameter search space for final model

Hyperparameter	**Defined space**
Criterion	Gini, Entropy
Max features	Sqrt, Log2, None
Min sample split	2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30
Min sample leaf	1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20
Max depth	None

Furthermore, from the findings presented in Appendix A.3.1, we concluded that the two most suitable features to include in the decision tree model was the topic and subject distance , shown in Table 15. For this reason, the RFECV step was skipped and instead the features provided for training the model were manually assigned set to *topic distance* and *subject distance.*

Table 15: Features used for decision tree model

β **value**	**Features used**
4	topic dist, subject dist

Table 16: Final model: Decision tree hyperparameters with β value 4

β value	**Criterion**	**Max features**	**Min sample split**	**Min sample leaf**	**Max depth**
4	Entropy	Sqrt	6	1	39

From the training of the decision tree, the best model's hyperparameters are shown in Table 16. What can be seen is that there are no hyperparameter on the edges of the search space defined in Table 14.

Finally, the classification results for the model with the hyperparameters shown in Table 16 showcased the predictive performance on the test data shown in Table 17. As can be observed, the decision tree model included far less non-keywords than the logistic regression model, with 794 to the logistic regression model's 2,888. However, this came at a cost of excluding 31 more keywords than the logistic regression model did. All in all, the use of the decision tree model would reduce a set of about 14,000 samples containing 240 keywords to a set of about 1,000 samples containing 181 keywords. Most probably, it would be quite manageable for an SEO specialist to manually look through this much smaller set of about 1,000 samples and make high quality decisions on which samples to mark as key or not key.

Table 17: Decision Tree Model Performance

		Predicted class	
		True	False
True Class	True	181 Included keywords	59 Excluded keywords
	False	794 Included non-keywords	13,113 Excluded non-keywords

8.2.3 Random forest model

The hyperparameter space was used for the random forest model is shown in table 18. As can be seen from inspecting the ranges, the hyperparameter space searched was narrower than that of the decision tree, where the random forest had 360 hyperparameter combination to the decision tree's 990. The reason for using a smaller grid space here was simply because of the computational power and time that training the random forest models needed since for every hyperparameter combination that was tried, 1000 individual trees were trained to make up the forest. However, this more narrow space was defined using the experience gained from training the decision trees och random forests, using a wider hyperparameter space, during the iterations to find the optimal β. As such, we argue that the hyperparameter space shown in Table 18 was suitable and fitting for the random forest model implemented.

Table 18: Random forest hyperparameter search space for final model

Hyperparameter	**Defined space**
Criterion	Gini, Entropy
Max features	Sqrt, Log2, None
Min sample split	4, 6, 8, 10, 12, 14, 16, 18, 20, 22
Min sample leaf	1, 2, 4, 6, 8, 10
Max depth	None

Moreover, just as with the decision tree model, the findings from iteration 3 presented in Appendix A.3.2 when searching for the optimal β value showed that the two best features to be used in the model was the topic and subject distance, shown in Table 19. For this reason, RFECV was not used during the grid search and training step, which greatly decreased the computational time and made possible for a wider hyperparameter space and a more thorough grid search which checked all combinations of hyperparameters.

Table 19: Features used for random forest model

β value	**Features used**
4	topic dist, subject dist

Training the random forest model for $\beta = 4$ and all hyperparameter combinations in the space shown in Table 18 produced the model presented in Table 20. As can be seen, no hyperparameter in the found solution were on the edge of the searched hyperparameter space.

Table 20: Final model: Random forest hyperparameters with β value 4

β value	**Criterion**	**Max features**	**Min sample split**	**Min sample leaf**	**Max depth(average)**
4	Entropy	None	20	6	44(27.25)

The trained final random forest model's predictive performance on the test set is displayed in Table 21. As can be observed, the overall predictions are quite similar to those of the final decision tree model. However, there are some small changes, with the random forest model catching one more keyword while also removing 85 more non-keywords. Essentially, this makes the random forest's performance on the test data objectively better. In the case of an SEO specialist applying this model the test set, then the resulting set would consist of about 900 samples containing 182 keywords.

Table 21: Final random forest model performance

		Predicted class	
		True	False
True Class	True	182 Included keywords	58 Excluded keywords
	False	709 Included non-keywords	13,198 Excluded non-keywords

8.3 Model comparison

For each of the three final models, multiple classification metrics were calculated. These are all displayed in Table 22, with the best performance highlighted in bold. It is worth reminding that since the same β value was used for the three models, a direct comparison between their results using the F_β metric was made possible.

Table 22: Model metrics from each algorithm

Metric	**Logistic Regression**	**Decision Tree**	**Random Forest**
Precision	0.08	0.16	**0.20**
Recall	**0.85**	0.75	0.76
Accuracy	0.84	0.93	**0.95**
F_1	0.16	0.27	**0.32**
$F_{\beta=4}$	0.55	0.60	**0.61**

Using the F_β metric, the random forest model performed best with a $F_{\beta=4}$ metric of 61 percent. Not far behind is the decision tree model with 60 percent and logistic regression with 55 percent. To understand how the F_β results came to be, both precision and recall should be investigated. By analyzing precision, a similar ordering is obtained, with random forest achieving 20 percent, decision tree 16 percent and logistic regression 8 percent. Moreover, the recall performance is best for logistic regression with 85 percent followed by random forest of 76 percent and lastly decision tree with one percent lower at 75 percent. This means that the tree based models are missing around 10 percent more keywords than the logistic regression model. However, the trade-off is that they also had the least amount of non-keywords in the final predictions as keywords. Tracking the accuracy and F_1 metrics, a similar order of performance is found as with other metrics, with the random forest at the top followed by the decision tree and logistic regression model.

By analyzing Table 22, one could possibly think that optimizing for F_1, precision or perhaps accuracy could be done instead since they provide the same model rankings as F_β. Doing this would however have caused the models to optimize for a metric that does not represent the goals of the thesis, and all the metric results would be completely changed from the results shown in Table 22. With this in mind, we reiterate that optimizing for $F_{\beta=4}$ was preferable over using the other metrics and subsequently provided us with better solutions.

Given the overall results, our interpretation is that the random forest model produces the best results out of the three. This is shown in practically all metrics, but most importantly in the F_β metric which most accurately reflects the sought after trade-off between keeping keywords and removing non-keywords.

9 Discussion, future work and limitations

Annalect's request for this thesis work was to find a solution that removed the main chunk of non-key samples, while keeping as many key samples as possible from a given set of data of the same outlook as the given data set. We argue that the models presented in Section 8 achieve this goal, and it can thus be said that the primary purpose of the thesis project has been fulfilled. Nevertheless, the chosen methods and found solutions do have their inherent limitations and issues, and there do exist opportunities for improvement. In this section, such questions and topics are covered.

9.1 Final model remarks

As stated, we argue that the three final models all fulfill their intended use and the ultimate goal of this thesis work. That being said, they do not all necessarily provide the same type of solutions. The found logistic regression model manages to catch almost all keywords, but also includes a lot of non-keywords. On the other side, the two tree based algorithms catch fewer keywords, but manage to exclude far more non-keywords. As a result, the three solutions can be said to fulfill somewhat different purposes. If faced with a solution where it is imperative that almost no keywords are missed, then the logistic regression is probably a better option. On the flip side, if some keyword loss is acceptable and a larger decrease in the total data set size is prioritized, then the two tree based models would probably be preferable.

Out of the two tree based models, the random forest model seemed to be the better choice as it both caught more keywords and excluded more non-keywords. As is known, the random forest algorithm is an ensemble of multiple decision trees all constructed through bootstrapped samples. As such, the random forest algorithm reduces the impact of individual tree errors and can thus be said to provide more reliable predictions. Summarily, it is probable that the random forest's greater performance stems from the superior stability that the procedure provides.

9.2 Solution's generalizability

As was presented in the exploratory data analysis section of this report, there was quite a large discrepancy between the number of samples from the three industries as well as from the four topics. This introduced the possibility that the final models, whatever they may be, could perform well on the overrepresented industries and topics but worse on the underrepresented ones. Furthermore. the models could possibly perform even worse on industries and topics not included in the data at all. Due to this potential problem, we trained some models on four of the five clients, and tested the models on the fifth. This was done five times in total, i.e., meaning that every client was once the "test set" for this experimental procedure. The findings were that the models performed well for clients, and therefore industries and topics, that they had not been trained on. This somewhat put the worries of the final models not being generalizable to rest.

9.3 Train and test split

Worth noting about the training and testing data for the models, is that the the split of training and testing data was done only once. Naturally, this meant that all the subsequent results somewhat depended on how this split was made and the randomness therein. However, the effect of said randomness in the splits is hard to measure and contextualize in a precise manner. For Clients 1, 2 and 3, their large quantity of data samples decrease the risk of the randomness in the split affecting the results too greatly. On the other hand, the randomness in the splits runs a greater risk of affecting the results for the smaller clients, namely Client 4 and 5. All in all, however, it is not believed that this randomness in the splits affected the final results too greatly. The reason for this is that the different clients all performed quite similarly, as was described in Section 9.2.

9.4 Pipeline structure of the project

A large part of the project work as a whole, which perhaps has not received so much focus and attention in this report, was the pipeline building aspect of the whole project. The solution that was to be presented to Annalect needed to be a coherent program that could be run on a data set with the same structure as the one that was given for this whole project. This was because the goal was to construct an implementation that Annalect could incorporate right away into their own pipeline structure. This meant that we needed to construct a program that performed the whole process from start to finish with *the push of a button*, as it were. That meant constructing a program that cleans the data, splits the troublesome words in the samples, runs the words through the embedding model, extracts the features, and then finally makes a prediction on the data. Given that many of these steps were quite substantial and consisted of many intermediate steps respectively, this became quite a big part of the thesis project as a whole.

9.5 Limited quantity of similar works

When conducting the literature review for this thesis work, we found that the quantity of similar previous works was quite limited. Our suspicion is that this was because the research that this thesis work concerns would most often be conducted in order to gain an advantage over its competitors. This type of research is thus probably more often conducted in the private sector rather than the public one. Also, if the research findings were made public, the research itself would become quite fruitless. This falls in line with the fact that most keyword extraction methods identified in the literature review were not made for websites, but instead for finding keywords for a larger coherent text. As such, the research stage of the thesis proved quite difficult as finding similar previous projects proved all but impossible. That being said, there were lessons to be learned from the findings presented in the literature review, even though they were not perfectly comparable to our own problem setting.

9.6 Label misclassification

From the data provided, there are some limitations that has been ongoing throughout our work and one of these are how the classification of labels has been done. What can been seen is that the labeling of keywords/non-keywords are not very consistent, where we have found samples labeled as keywords that should probably be non-keywords. It is unknown how many samples that are like this, but for our work, it has been disregarded. This is assumed to reflect upon the models performance, but since we do not have other data to work with, we cannot verify this assumption. Given this, we believe that for one to continue this work, the labels of the data should be checked once more.

9.7 Missing variables

Another limitation, which we believe would have a great improvement on the model is including a feature that described which type of text the string was on the website, such as *title, subtitle, body, header* and *footer*. Our interpretation of other papers was that such information has been used quite frequently as a feature, where more relevant information is often written in titles and subtitles compared to plain text. However, since the data provided did not have any of such information or underlying information where we could extract it from, it could not be included into our models. We have provided this information to Annalect, which they have taken into consideration and are planning on including for their new clients. Other information about each sample that is believed to provide possibly better performance in the models is information such as the *first occurrence* and *TF-IDF*.

9.8 Suboptimal word splits

While working with and exploring the data, we noticed that there existed some samples where the splitting of the scraped text from the website had been handled quite questionably. We can only assume that this, to some extent, caused the models to perform somewhat worse than they might otherwise had if the splitting had been done more correctly. As previously mentioned, the splitting itself was done by a natural language model during the process of extracting the data from the company websites. As such, we can present no direct for this problem, but it could be worthwhile looking further into this segment as to get a better original data set with samples that make more sense.

9.9 Choice of word embedding

In our solution of the thesis work, we chose to use a word embedding made by NLPL to calculate the relevancy between words. When researching Swedish word embeddings, it was found that there did not seem to exist that many open sourced embedding models, which in itself has been a limitation for us. If the language for the project would have been a more widespread language, such as English, it would have allowed us to pick between a larger variety of models as well as more advanced and extensively trained models. Despite this, we did only apply one word embedding during this work. If someone were to set upon this same project, or a similar one, a plausible improvement to the result could be to dive deeper into the world of word embedding models, as it is a foundational pillar upon which the result of this project stands. Perhaps one could even translate all the words in the data to English and through that open up the possibility to use an English word embedding model. This approach was considered but during discussions with Annalect abandoned, as they belived that too much subcontext would be lost in such a translation. Nevertheless, this might be an approach worth investigating.

Furthermore, as was loosely discussed during the literature review, the possibility does exist to train a problem specific embedding model. In this instance, that would most likely mean training an embedding model exclusively on a plethora of different company websites. If this was done correctly and in a large enough quantity, this could very well also improve the performance of the word embedding model, and in extension the performance of the ultimate classification models.

9.10 Extracting subjects and calculating relevancy

One of the things we identified quite early in the process was that the assigned topics and industries could not catch enough information related to each sample string. With this in mind, we extracted words based on all the samples via clustering, which we defined as *subjects*. We think the current solution is quite good and does give more topical information than only those of the industry and topic variables, but the solution could of course be improved upon. The most obvious way to try and improve this method would likely be to investigate and implement other types of clustering methods. This was done by us to an extent, but a deep dive into the different methods and their respective implementations would surely open up some possible improvements for extracting information about the data not found in the topic and industry features.

The identified subjects aimed to be more closely related than both topic and industry which we believe is achieved. However, what has been found is that it seems to be too closely related to many samples instead because of this. This is possibly due to the current implementation only uses the most closely relevant word within the whole sample string. To solve this, an implementation that allows more than one word within the sample string to check the samples relevancy could be worth testing. However, with such implementation, a weighting scheme should be considered, since it is often mostly one or two words that are closely related to the subjects. Furthermore, such weighting scheme could be implemented for topic and industry distance as well.

Another approach that was lightly tested was using an intelligent chat-bot such as ChatGPT to extract such subjects. These initial tests proved fairly promising, but we did not follow through with a broader investigation since our approach of extracting subjects had already been created.

9.11 Other classification algorithms

Initially we were planning on also implementing a neural network model as a potential solution but given the poor performance that was achieved from our initial testing, we felt it was not worth continuing testing. However, a assumption we have is that such solution is one of the better since neural networks is especially good on finding subtle nuances in data. The problem could be that we did not find a good structure to the network or that it might have needed much more data, especially compared to the other algorithms tested.

References

Annalect. Data-powered marketing, 2023. URL `https://www.annalect.com/effectiveness/`. Retrieved 2023-02-26.

Michael Buckland and Fredric Gey. The relationship between recall and precision. *Journal of the American society for information science*, 45(1):12–19, 1994.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

Nazanin Firoozeh, Adeline Nazarenko, Fabrice Alizon, and Béatrice Daille. Keyword extraction: Issues and methods. *Natural Language Engineering*, 26(3):259–291, 2020.

Gensim. Keyed vectors, 2022. URL `https://radimrehurek.com/gensim/models/keyedvectors.html`. Retrieved 2023-05-15.

Anette Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223, 2003.

Gareth James, Trevor Witten, Daniela & Hastie, and Robert Tibshirani. *An Introduction To Statistical Learning*, volume 112. Springer, 2013.

NLPL. Nlpl word embeddings repository, 2017. URL `http://vectors.nlpl.eu/repository/`. Retrieved 2023-05-11.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

Muthukumaran Ramasubramanian, Hassan Muhammad, Iksha Gurung, Manil Maskey, and Rahul Ramachandran. Es2vec: Earth science metadata keyword assignment using domain-specific word embeddings. In *2020 SoutheastCon*, pages 1–6. IEEE, 2020.

Mohammad Rezaei, Najlah Gali, and Pasi Fränti. Clrank: A method for keyword extraction from web pages using clustering and distribution of nouns. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 1, pages 79–84. IEEE, 2015.

Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory To Algorithms.* Cambridge university press, 2014.

spaCy. Swedish, 2023. URL `https://spacy.io/models/sv`. Retrieved 2023-05-11.

Statology. How to create a precision-recall curve in python, 2023. URL `https://www.statology.org/precision-recall-curve-python/`. Retrieved 2023-05-16.

Tutorialspoint. Does label encoding affect tree-based algorithms?, 2023. URL `https://www.tutorialspoint.com/does-label-encoding-affect-tree-based-algorithms`. Retrieved 2023-05-12.

Andreas Veglis and Dimitrios Giomelakis. Search engine optimization. *Future Internet*, 12(1):1, 2019.

Wordnet. What is wordnet, 2023. URL `https://wordnet.princeton.edu/`. Retrieved 2023-05-10.

Peipei Xia, Li Zhang, and Fanzhang Li. Learning similarity with cosine similarity ensemble. *Information sciences*, 307:39–52, 2015.

Appendices

A Iterations for finding beta

Here, the detailed steps for finding the final β's for each algorithm type is described. The overall procedure is presented in Section 8.1. To summerize, by qualitative inspection of the confusion matrices provided by each best performing model based on the β values used, we chose a smaller range of β values until we found a single β value. In total, the procedure consisted of three iterations, from each of which some conclusions were drawn going into the next iteration.

A.1 Modeling iteration 1

A.1.1 Logistic regression model

For the logistic regression algorithm, the were no hyperparameters assigned since none of them were deemed fitting or relevant for the given problem. As such, it was quite fast and simple to train a model for each β value in the wide range for the initial search iteration. As can be seen from the results presented in Appendix B, the best performing models seemed to be the ones found using β values somewhere between 3-15.

A.1.2 Decision tree model

Contrary to the logistic regression model, the decision tree model had multiple hyperparameters to be examined when finding a suitable β value. The hyperparameter space used for this first search iteration is shown in Table 23.

Table 23: Decision tree hyperparameters

Hyperparameter	**Defined space**
Criterion	Gini, Entropy
Max features	Sqrt, Log2, None
Min sample split	2, 3, 4, 5, 6, 8, 10
Min sample leaf	1, 2, 4, 6, 8, 10, 20, 30
Max depth	2, 3, 4, 6, 8, 10, 12, 15, 20

As with logistic regression, numerous β values ranging from $0.1 - 100$ were investigated. Here, however, there was now a hyperparameter space to be taken into account. For each beta value in the range, 10 separate models were trained on 10 random hyperparameter combinations. The results shown for each β value in Appendix B belong to the best performing model for that β, determined by the F_β score.

The best model performances, judged by the qualitative evaluation of confusion matrices, seemed to be models found using β values ranging somewhere around 3-15. Furthermore, analyzing the hyperparameter combinations within the range 3-15, namely those with $\beta = 5, 10, 15$ showed that the edge of the range for the *max depth* hyperparameter had been reached for the model using $\beta = 5$.

Table 24: Decision tree hyperparameters with β values between 5-15

β **value**	**Criterion**	**Max features**	**Min sample split**	**Min sample leaf**	**Max depth**
5	Entropy	Sqrt	8	1	20
10	Entropy	None	2	2	6
15	Entropy	None	2	2	6

A.1.3 Random forest model

The hyperparameters used for the first iteration are the same as those for the decision tree, shown in Table 23. This algorithm did also seem to have a preferred β between 3-15, see appendix B. The analysis of the hyperparameters showed a similar result as that of the decision tree model where one can see that *Max depth* is once again on the edge when $\beta = 5$.

Table 25: Random forest hyperparameters with β values between 5-15

β **value**	**Criterion**	**Max features**	**Min sample split**	**Min sample leaf**	**Max depth**
5	Entropy	Sqrt	8	1	20
10	Gini	Sqrt	6	6	12
15	Gini	Sqrt	6	6	12

A.1.4 Findings iteration 1

From the results of the first iteration for the three algorithms, the best β value seems to lie somewhere between 3-15 for all of them. Furthermore, as was found for the decision tree and random forest algorithms, the range for *max depth* should be increased from the current maximum 20. Thus, the max depth of 25 and 30 will also be included in the hyperparameter space for the next iteration Also, in order to make possible for more thorough searches for each subsequent β value, the number of hyperparameter combinations investigated per β will be increased from 10 to 50 for both the decision tree and random forest.

Finally, since the β values ranging from 3-15 seemed to produce the best models during this first iteration for all algorithms, that range is the natural choice for further investigation during the second iteration. As a result, the β values to be investigated during that iteration will be all odd values between 3 and 15, namely a range consisting of 3, 5, 7, 9, 11, 13 and 15.

A.2 Modeling iteration 2

A.2.1 Logistic regression model

Given the chosen range of β values to investigate in this second iteration, the results presented in Table 26 were found. Interestingly, all models seem to find a very similar, if not identical, predictor since the test results are equal between all β values.

Table 26: Logistic regression results with β values between 3-15

β **value**	**True Positive**	**False Negative**	**True Negative**	**False Positive**
3	212	28	11019	2888
5	212	28	11019	2888
7	212	28	11019	2888
9	212	28	11019	2888
11	212	28	11019	2888
13	212	28	11019	2888
15	212	28	11019	2888

A.2.2 Decision tree model

The results from the new β values are shown in Table 27. Given the instructions of the thesis, a β value between 3-7 seems most appropriate as these seem the best at removing non-key samples while also including many key samples. This interpretation was further verified with the supervisor at Annalect as to make sure that the modeling continued in the right direction.

Table 27: Decision tree results with β values between 3-15

β **value**	**True Positive**	**False Negative**	**True Negative**	**False Positive**
3	177	63	13102	805
5	192	48	13008	899
7	198	42	11996	1911
9	203	37	11999	1908
11	203	37	11999	1908
13	203	37	11999	1908
15	203	37	11999	1908

The hyperparameters used for β = 3-7 are shown in Table 28. To be noticed, there seems to be a separation of the hyperparameters used based on the β value. For β = 3,5 identical hyperparameters are used which differs quite much compared to $\beta = 7$. This is also reflected upon the model predictions, where the model using $\beta = 7$ produced much higher *False positives* shown in Table 27 compared to the other two models. Furthermore, for the hyperparameter combinations using β value of three and five, *Max depth* = 30, which was on the edge of the search space.

Table 28: Decision tree hyperparameters with β values between 3-15

β **value**	**Criterion**	**Max features**	**Min sample split**	**Min sample leaf**	**Max depth**
3	Gini	Sqrt	6	1	30
5	Gini	Sqrt	6	1	30
7	Entropy	None	6	4	10

A.2.3 Random forest model

Investigating β values between 3-15 produced models whose performances are presented in Table 29. Through qualitative inspection, the models that produced the preferable confusion matrices seemed to lie in the 3-7 range.

Table 29: Random forest results with β values between 3-15

β **value**	**True Positive**	**False Negative**	**True Negative**	**False Positive**
3	187	53	13092	815
5	187	53	12542	1365
7	191	49	12299	1608
9	202	38	11949	1958
11	203	37	11850	2057
13	203	37	11850	2057
15	203	37	11850	2057

Finally, when checking the hyperparameters from the chosen models shown in Table 30, one can see that $\beta = 3$ resulted in a model where *Max depth* reached the edge of the space once again. Furthermore, the *Min sample split* hyperparameter was also on the edge for $\beta = 3{,}5$ and therefore increased the range by 12, 14 and 16.

Table 30: Random forest hyperparameters with β values between 3-7

β **value**	**Criterion**	**Max features**	**Min sample split**	**Min sample leaf**	**Max depth**
3	Entropy	Sqrt	10	2	30
5	Entropy	None	10	6	10
7	Entropy	None	6	4	10

A.2.4 Findings iteration 2

Concerning the logistic regression algorithm, the results provided by the different β values were identical, which implies that a smaller interval of β values would not yield better results. For that reason, the logistic regression model was not trained any further since it seems that it has reached its highest potential.

From the tree algorithms, both decision tree and random forest have clear new intervals to be searched. Interestingly, the interval that seemed most correct to be searched during the next iteration was 3-7 for both algorithms. As such, the next iteration will use this range for the two tree based algorithms. Furthermore, the hyperparameter space for both these algorithms will also be expanded for the next iteration for the *max depth* and *min sample split* hyperparameters. This is because the limits for these were reached during this second iteration. Consequently, the hyperparameter space for those hyperparameters was expanded by 35, 40 and 50 and 12, 14 and 16 respectively.

A.3 Modeling iteration 3

A.3.1 Decision tree model

From the results of the best performing models for β values 3-7, shown in Table 31, one can see that the *True positives* are quite similar for all β vales, while the *False positives* increase a significant amount for β value five and six. Given this, the optimal β should be between 3-5, where the results from $\beta = 5$ seem best. However, the difference in result between β four and five are very small, where it could be from the random search provided a more optimal hyperparameter combination. Since β three and four are identical and five is only marginally better, the average of the three values will be taken, being $\beta = 4$.

Table 31: Decision tree results with β values between 3-7

β **value**	**True Positive**	**False Negative**	**True Negative**	**False Positive**
3	190	50	12984	923
4	190	50	12984	923
5	193	47	12943	964
6	194	46	12608	1299
7	200	40	12077	1830

By analyzing the hyperparameters the model using for $\beta = 4$ in Table 32, one can see that this time the *Max depth* did not use the edge of the search space as in previous attempts.

Table 32: Decision tree hyperparameters with β value 4

β **value**	**Criterion**	**Max features**	**Min sample split**	**Min sample leaf**	**Max depth**
4	Entropy	None	6	1	35

Finally, something interesting was noted when studying the features that remained in each model after the feature selection done by RFECV had removed all features which were deemed to not improve the model. In fact, it was only the topic distance and subject distance that were included in the best performing models for β values ranging from 3 through 7. These results are shown in Table 33.

Table 33: Features used in the best models for the decision tree algorithm, betas 3-7.

β **value**	**Features used**
3	topic distance
4	topic distance
5	topic distance
6	topic distance
7	topic distance, subject distance

Considering the findings in Table 33, the likely conclusion is that it is only the topic distance that should be included in the decision models that we had experimented with. However, we knew that the topic feature itself was quite unreliable. This was both due to that it was set manually by an SEO specialist, and thus contained somewhat of a risk for human error, but mostly due to the fact that it is not at all certain that all important aspects of a client would be accurately represented in the topic feature. As a result, using only the topic distance as a feature for the classification model would run the risk of producing an unstable and ungeneralizable model. For that reason, we chose to manually set the RFECV to at least keep two features in the models, and run the procedure again for β values 3 through 7. Given the findings presented in Table 33, it was believed that this would result in the models containing both the topic distance as well as the subject distance, whose very meaning was to catch categories in the data that was not represented in the topic or industry parameters. Our belief was confirmed, and the findings from this second search of β values 3-7 are displayed in Table 34.

Table 34: Features used using minimum 2 features

β **value**	**Features used**
3	topic distance, subject distance
4	topic distance, subject distance
5	topic distance, subject distance
6	topic distance, subject distance
7	topic distance, subject distance

Summarily, the findings from the third iteration showed that the best β value for the decision tree algorithm was $\beta = 4$. Furthermore, the two features that should be included when training the model should be the topic and subject distance.

A.3.2 Random forest model

The results from training training random forest models using β values 3-7 are shown in Table 35. The results have a similar trend to those for the decision tree, where the *True positives* are similar for all models except $\beta = 7$, while *False positives* is heavily increased between $\beta = 5{,}6$. With this in mind, the best choice seems to be using a β value of 3, 4 or 5. Using the same reasoning as for the decision tree, taking the average of the three best β values seemed to be a good choice. Furthermore, choosing the same β value for both algorithms would would allow for direct comparison of the two methods. With this reasoning, $\beta = 4$ was chosen also for the random forest.

Table 35: Random forest results with β values between 3-7

β **value**	**True Positive**	**False Negative**	**True Negative**	**False Positive**
3	185	55	13129	778
4	186	54	13115	792
5	186	54	13115	792
6	192	48	12313	1594
7	220	20	10790	3117

Furthermore, analyzing the model hyperparameters in Table 36, no hyperparameters were on the edge of the search grid, but *Max depth* had once again increased.

Table 36: Random forest hyperparameters with β value 4

β **value**	**Criterion**	**Max features**	**Min sample split**	**Min sample leaf**	**Max depth**
4	Entropy	None	10	1	40

Similarly as for the decision tree, an analysis of the features used for the models presented in Table 35 was made. The features used in the best performing random forest models for β values 3-7 can be seen in Table 37, where it is observable that all β values but 7 produced a model containing only the topic distance. This falls in line with the similar findings for the decision tree model, where the topic distance also showed to basically be the only well performing feature.

Table 37: Features used for random forest models

β **value**	**Features used**
3	topic distance
4	topic distance
5	topic distance
6	topic distance
7	phrase length, term frequency, has NOUN, industry dist, topic dist, subject dist, POS encode

With the same reasoning as for the decision tree, new models for β values 3-7 were calculated, where the RFECV procedure was set to keep at least two parameters. Again, similarly for the decision tree, the results were identical. Looking at Table 38, it seems quite clear that the best performing models using at least two features are models using the topic and subject distance. Noteworthy to point out here is the fact that the model using $\beta = 7$ now produced a model only using the two mentioned features where it previously, as showed in Table 37 used far more. The reason is that during this stage of investigating which two features should be used, the hyperparameter space used was far smaller as to shorten the computational time. All in all, the takeaway was, similarly for the decision tree, that the best two features to use were the topic and subject distance.

Table 38: Features used using minimum 2 features

β **value**	**Features used**
3	topic dist, subject dist
4	topic dist, subject dist
5	topic dist, subject dist
6	topic dist, subject dist
7	topic dist, subject dist

A.3.3 Findings iteration 3

The findings from this third and final iteration can be summarized quite simply. The chosen final β values are $\beta = 4$ for both the decision tree model and the random forest model. With this result, the possibility to optimize these two models for said β given their respective hyperparameter spaces was achieved. The final take-away is that for the decision tree and random forest models, the best performing models both used only the topic and subject distance features.

A benefit that comes with two of the β values being the same, is that it allows for comparison of the F_β results between the models. Moreover, to make the results comparable with the logistic regression model, the choice was made to also for this algorithm use $\beta = 4$, since it had identical results over the interval 3-15 during iteration 2.

B Iteration 1 results

	Classifier	β	**Precision**	**Recall**	**Confusion matrix**	F_β **score**
0	Logistic_regression	0.1	0.07	0.88	[[11087, 2820], [30, 210]]	0.066619
1	Logistic_regression	0.5	0.07	0.88	[[11087, 2820], [30, 210]]	0.080933
2	Logistic_regression	1.0	0.07	0.88	[[11087, 2820], [30, 210]]	0.122457
3	Logistic_regression	2.0	0.07	0.88	[[11022, 2885], [28, 212]]	0.251488
4	Logistic_regression	5.0	0.07	0.88	[[11019, 2888], [28, 212]]	0.581212
5	Logistic_regression	10.0	0.07	0.88	[[11019, 2888], [28, 212]]	0.757481
6	Logistic_regression	15.0	0.07	0.88	[[11019, 2888], [28, 212]]	0.804254
7	Logistic_regression	20.0	0.02	1.00	[[8, 13899], [0, 240]]	0.874276
8	Logistic_regression	30.0	0.02	1.00	[[8, 13899], [0, 240]]	0.939849
9	Logistic_regression	50.0	0.02	1.00	[[8, 13899], [0, 240]]	0.977463
10	Logistic_regression	100.0	0.02	1.00	[[8, 13899], [0, 240]]	0.994267
11	Decision_Tree	0.1	0.15	0.57	[[13155, 752], [104, 136]]	0.180091
12	Decision_Tree	0.5	0.15	0.57	[[13155, 752], [104, 136]]	0.203837
13	Decision_Tree	1.0	0.17	0.71	[[13087, 820], [70, 170]]	0.248827
14	Decision_Tree	2.0	0.14	0.77	[[12803, 1104], [55, 185]]	0.392421
15	Decision_Tree	5.0	0.14	0.81	[[12718, 1189], [45, 195]]	0.645602
16	Decision_Tree	10.0	0.08	0.87	[[11386, 2521], [31, 209]]	0.781062
17	Decision_Tree	15.0	0.06	0.91	[[10370, 3537], [22, 218]]	0.831096
18	Decision_Tree	20.0	0.06	0.91	[[10370, 3537], [22, 218]]	0.850322
19	Decision_Tree	30.0	0.06	0.91	[[10370, 3537], [22, 218]]	0.864664
20	Decision_Tree	50.0	0.06	0.91	[[10370, 3537], [22, 218]]	0.872214
21	Decision_Tree	100.0	0.06	0.91	[[10370, 3537], [22, 218]]	0.875443
22	Random_Forest	0.1	0.55	0.38	[[13832, 75], [150, 90]]	0.529509
23	Random_Forest	0.5	0.52	0.42	[[13813, 94], [140, 100]]	0.472609
24	Random_Forest	1.0	0.45	0.46	[[13769, 138], [129, 111]]	0.431438
25	Random_Forest	2.0	0.19	0.78	[[13123, 784], [53, 187]]	0.463643
26	Random_Forest	5.0	0.19	0.78	[[13123, 784], [53, 187]]	0.654629
27	Random_Forest	10.0	0.07	0.91	[[11005, 2902], [21, 219]]	0.816650
28	Random_Forest	15.0	0.07	0.91	[[11005, 2902], [21, 219]]	0.867444
29	Random_Forest	20.0	0.07	0.93	[[10747, 3160], [16, 224]]	0.886713
30	Random_Forest	30.0	0.07	0.93	[[10747, 3160], [16, 224]]	0.902717
31	Random_Forest	50.0	0.07	0.93	[[10747, 3160], [16, 224]]	0.911154
32	Random_Forest	100.0	0.07	0.93	[[10747, 3160], [16, 224]]	0.916843