NTNU

Norwegian University of
Science and Technology

Extending Decision Support for the Norwegian Labour Inspection Authority Through Open and Unstructured Data Sources

Methods for detecting relevant information
based on external data

Kathrine Løfqvist
Caroline Odden

Master of Science in Informatics
Submission date: May 2018
Supervisor: Herindrasana Ramampiaro, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

The Norwegian Labour Inspection Authority supervises thousands of enterprises every year and strives for preservation of the employee's health and safety. The possibility to detect work related events in open sources is highly desirable, and a system identifying this information will serve as a supplement to the internal systems they already possess.

The world wide web contains a large amount of unstructured data on different platforms, and this makes it complicated to detect relevant information. Harvesting the data from the different sources may offer some challenges, since not all platforms offer open API solutions. In this thesis we focus on how we can detect relevant data for the Norwegian Labour Inspection Authority. The most important steps in the approach are to extract information from open sources and use Artificial Neural Network and Deep Learning to classify the information as relevant or irrelevant. The next step is to cluster and extract the topic from the clusters to get an overview of the data, and finally the relevant information is presented.

As there were no benchmark dataset to evaluate our proposed system, we gathered data from online news papers and assessed our system on our retrieved dataset. By using neural network on the data gathered from open news articles, we achieve good results for the Norwegian Labour Inspection Authority. To the best of our knowledge, it does not exist any previous work with the Norwegian Labour Inspection Authority on Norwegian text, and as a result of this our proposed system is state-of-the-art.

Sammendrag

Arbeidstilsynet fører tilsyn av tusenvis av bedrifter hvert år og arbeider for å bevare arbeidstakerens helse og sikkerhet. Muligheten til å oppdage arbeidsrelaterte hendelser i åpne kilder er svært ønskelig, og et system som identifiserer denne informasjonen vil tjene som et supplement til de interne systemene de allerede har.

Internett inneholder store mengder ustrukturerte data spredt på ulike plattformer, og dette gjør det vanskelig å oppdage relevant informasjon. Innhenting av data fra de ulike kildene medfører noen utfordringer siden ikke alle plattformer tilbyr åpne API-løsninger. I denne oppgaven fokuserer vi på hvordan vi kan detektere relevante data for Arbeidstilsynet. De viktigste stegene i fremgangsmåten er å hente informasjon fra åpne kilder, og bruke kunstig neurale nettverk og "deep learning" for å klassifisere informasjonen som relevant. Det neste steget er å klynge og trekke ut emner fra klyngene for å få en oversikt over dataene, og til slutt presenteres den relevante informasjonen.

Siden det ikke eksisterer et datasett for å evaluere vårt foreslåtte system, samlet vi inn data fra nyhetsartikler på internett og vurderte vårt system opp mot dette datasettet. Ved å bruke nevrale nettverk på data hentet fra åpne nyhetsartikler oppnådde vi gode resultater. Denne tilnærmingen med Arbeidstilsynet og norske tekster har ikke vært gjort i tidligere arbeid, så vidt vi vet, og som et resultat av dette er vårt foreslåtte system state-of-the-art.

Preface

This thesis is delivered as a final fulfillment of the requirements for a Master of Science in Informatics with specialization within Databases and Search, and Artificial Intelligence. The research was conducted at the Department of Computer and Information Science (IDI) from the fall of 2017 to the spring of 2018, and submitted to the Norwegian University of Science and Technology (NTNU) in Trondheim.

Acknowledgements

First of all we would like to thank our supervisor Heri Ramampiaro for great guidance, support and encouragement. In addition we would like to thank our co-supervisor Sigmund Akselsen for helpful feedback. We are very grateful for all the discussions and meetings throughout the project.

We would also like to thank the Norwegian Labour Inspection Authority and Ole André Brevik for providing this task and for the consultation during this project. Their feedback and support contributed to a great cooperation.

Finally, a big thank you to Magnus, Mikael, Michael and Kristin for valuable feedback on the report!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter the thesis is presented. It begins with the motivation for the research to define the value of this thesis. The context of our work is explained to establish which problem area our thesis applies to, followed by the problem definition with the research questions we will aim to answer. The motivation is presented in Section 1.1, and the context description in Section 1.2. The problem definition with the research questions is in Section 1.3, and finally the thesis outline is presented in Section 1.4.

1.1 Motivation

Cases as accidents, social dumping and harrassments take place at Norwegian workplaces every year. Consider the *#metoo* campaign[1] as an example, a campaign where people who have experienced sexual harassment at their workplace by their bosses publish their stories along with the hash tag. Another violation compared to Norwegian standards and laws, is the use of foreign employees as cheap labour (referred to as social dumping[2]). In addition, accidents due to poor adherence of environment, health and safety regulations leads to either injuries or even more serious outcomes[3].

All these cases influence the well-being of the employees and can cause serious consequences. Cases like this should not happen in work places and luckily there exists a governmental agency which focuses on occupational health and safety, named the Norwegian Labour Inspection Authority (Arbeidstilsynet). The Norwegian Labour Inspection Authority strives for a safe and good working environment for employees in Norway, and controls that the Working Environment Act[4] is followed. The Norwegian Labour Inspection Authority supervised approximately 13 thousand workplaces in 2017[5], and the inspections was both

[1] https://no.wikipedia.org/wiki/Me_too
[2] https://www.arbeidstilsynet.no/tema/sosial-dumping/
[3] https://www.ssb.no/helse/statistikker/arbulykker
[4] https://www.arbeidstilsynet.no/regelverk/lover/arbeidsmiljoloven
[5] https://www.arbeidstilsynet.no/om-oss/forskning-og-rapporter/arsrapporter/

announced and unannounced. All together they are responsible for supervising 180 thousand enterprises in the private and public sector, as well as 60 thousand units in agriculture[6]. To penalize enterprises for not following the law, the Norwegian Labour Inspection Authority may impose improvements of certain conditions that does not satisfy requirements in the regulations, as well as impose fines[7]. This is to prevent accidents and take care of the employee's health and welfare. Today, the Norwegian Labour Inspection Authority conducts arbitrary controls of organizations which may result in inspections with no findings, leading to unnecessary resources spent on enterprises where things are in order. To be able to contribute to the work the Norwegian Labour Inspection Authority perform, it is necessary to retrieve valuable data to research. Based on a number of unstructured sources available, it should be possible to determine whether a particular text from a data source is within the area of interest for the Norwegian Labour Inspection Authority.

The world wide web contains much information and have a broad specter of data. The micro blogs[8] in social media (e.g. Twitter and Facebook), and more formal sources such as news papers and police updates have the possibility to contain much unstructured but valuable information. As social media has become a huge part of today's communication, being able to analyze data and see user-generated texts from this platform are interesting for the Norwegian Labour Inspection Authority. Due to the major use of Facebook in Norway[9] but restricted access settings and the open access but little use of Twitter in Norway, the possibilities for information extraction were somewhat limited. This caused some difficulties when researching the use of data from social media and led to investigation of public web documents such as news articles.

In this thesis we investigate how to develop a decision support system based on relevant data from web documents. By using methods from machine learning we will try to detect deviations such as accidents and illegal work to be able to report it to the Norwegian Labour Inspection Authority. The main goal for this thesis is the proposition of a machine learning method to detect interesting and valuable data for the Norwegian Labour Inspection Authority from web documents and make this information available for further use.

1.2 Context

The Norwegian Labour Inspection Authority conducts inspections of enterprises on a regular basis as explained earlier, and before carrying out an inspection valuable information is gathered about the enterprise. Where this information is gathered before each inspection varies, and so does the time and place. After a meeting with three inspectors from the Norwegian Labour Inspection Authority we gained valuable insight into their approach to gather information. They use different sources to obtain information, and one of the sources is *tips*; from customs, police or other trustworthy sources. Manual Google and Facebook search is

[6] https://www.arbeidstilsynet.no/om-oss/
[7] https://www.arbeidstilsynet.no/om-oss/tilsyn/
[8] https://www.lifewire.com/what-is-microblogging-3486200
[9] https://www.ipsos.com/nb-no/ipsos-some-tracker-q118

conducted to collect information about persons and companies, also companies' websites is used in the investigation. One of their main interests is connection between persons, verification of companies through persons information, and to detect information that confirms illegal work. A Google search may lead to a news article or social media posts containing interesting information, but the search in these documents is done manually.

The Norwegian Labour Inspection Authority have developed a service called *Virksomhetstjenesten*, providing information about all Norwegian enterprises. This service is based on data from *Enhetsregisteret*[10], the central coordinating register for legal entities in Norway, and extended with data from inspections and other registers managed by the Norwegian Labour Inspection Authority. Virksomhetstjenesten (Enterprise service) contains Virksomhetssøk (Enterprise search providing multi-faceted search) and Virksomhetskort (Enterprise card providing all relevant information on each specific enterprise). The work conducted in this thesis will serve as research for a decision support system, that can be a helpful asset for the Norwegian Labour Inspection Authority and facilitate Virksomhetskortet. Such a decision support system will not completely replace a manual Google search after a specific person or company, but such a system can contribute to detect relevant information that can reveal persons and/or companies worthy of further investigation.

To be able to detect relevant information there are several methods that can be used, and a potential approach to this task is machine learning. The Norwegian Labour Inspection Authority already have projects using machine learning[11] to predict types of companies with the highest probability of committing violations of the working environment act. This system uses data the Norwegian Labour Inspection Authority already have, e.g. reports and data from previous controls. A decision support system will be based on open data and will work as a supplement to the already existing systems. This supplement will give the Norwegian Labour Inspection Authority access to the latest relevant news and will contribute to the important work of revealing companies committing violations of the working environment act.

One of the main reasons the Norwegian Labour Inspection Authority may benefit from a new system is the number of companies they are responsible for overseeing. In Enhetsregisteret there are registered over 230.000 companies, with at least one employee. Our approach will provide a foundation to create a new and flexible machine learning system, providing more relevant information from open data sources to Virksomhetskortet. This new system will hopefully free resources and give the inspectors the opportunity to focus on the supervision itself. They will still need to evaluate cases where the need for supervision is the highest and decide if it is more severe to get injured at work today or getting sick of asbestos in 30 years. There are several cases that need to be taken into account, and a decision support system will not be the final decision maker for who the Norwegian Labour Inspection Authority should supervise but create a suggestion to a direction they could investigate.

[10] https://www.brreg.no/om-oss/oppgavene-vare/alle-registrene-vare/om-enhetsregisteret/

[11] https://www.dagensperspektiv.no/arbeidsliv/2018/arbeidstilsynet-bruker-maskinlaering-for-a-finne-skurkene-i-arbeidslivet

1.3 Problem Definition

The task of detecting enterprises of relevance for the Norwegian Labour Inspection Authority is a broad task, and to narrow it down a scope needs to be set. To research the possibility to develop a decision support system for the Norwegian Labour Inspection Authority a proper data base must be established, and further computation techniques need to be set to process the data sufficiently.

Deciding what data to be used, from internal reports or open sources, unstructured or structured, had to be decided. The Norwegian Labour Inspection Authority was interested in investigating open data sources, as they already had systems using internal data from earlier reports from supervisions. It was therefore desirable to examine online news articles and social media to investigate if and how this data can be useful. Regarding text processing there exist many techniques applicable for text representation and text classification and researching all methods will be a too comprehensive task. The methods we chose to investigate was narrowed down to machine learning techniques.

In this thesis we want to extract a representative amount of data from open sources and process this data by techniques within machine learning. We want to focus on which methods that are applicable for our data and gives the most sufficient results. Finally, we want to find a method to represent the data, making it easier to detect correlations among the retrieved data. Note that the scope of this thesis does not include developing a final decision support system, but rather research the methods creating beneficial results. Based on this, the scope of this thesis is set to investigate and extract information from open sources, classify the information and represent the data.

With this scope in mind, the thesis will try to answer the following main research question:

> **RQ:** *How can relevant information for the Norwegian Labour Inspection Authority be detected in open data sources?*

To be able to answer the main research question, we divide it into four sub questions:

> **RQ 1:** *Where can we extract the most usable data from open sources to create a valuable dataset?*
>
> **RQ 2:** *How can the data be processed and represented for further processing?*
>
> **RQ 3:** *How can we develop a suitable machine learning method that is able to find relevant data?*
>
> **RQ 4:** *How can we present the relevant data making it more applicable for the Norwegian Labour Inspection Authority?*

Based on this problem definition the main contributions of this work are:

i) *Exploration of new sources of data for the Norwegian Labour Inspection Authority*

ii) *A Norwegian dataset consisting of news articles annotated as relevant or irrelevant for the Norwegian Labour Inspection Authority*

iii) *A deep learning architecture for text classification of news articles*

1.4 Thesis Outline

Chapter 1 is the introduction of this thesis explaining the motivation and why we are researching this field, Chapter 2 addresses the related work relative to our proposed approach with focus on the classification models. The thesis then continues with explanation of the background material in Chapter 3 for the relevant methods we examine, followed by the approach of gathering the data and retrieval of relevant documents for The Norwegian Labour Inspection Authority in Chapter 4. The results and evaluation of the results from the approach is presented and discussed in Chapter 5, presenting the success of our proposed system. Finally, the drawn conclusion of which methods produced the best results along with suggestions for future work are presented in Chapter 6.

Chapter 2

Related Work

To the best of our knowledge there exists no directly related work to this thesis. This chapter will therefore present articles with approaches and methods that are relevant for our thesis, along with a summary to outline the aspects we decide to bring along further for this research. The main reasons this thesis differs from the related articles is the use of Norwegian text and the cooperation with the Norwegian Labour Inspection Authority, which require a unique collection of data and support for the Norwegian language. Related neural network approaches will be presented in Section 2.1, followed by standard classification methods such as Naive Bayes and Decision Trees in Section 2.2, and then other related work concerning clustering will be described in Section 2.3. A summary will be given in Section 2.4.

2.1 Neural Network Approaches

Georgios et al. (2018) present a Recurrent Neural Network (RNN) system composed of multiple Long Short-Term Memory (LSTM) based classifiers for detecting offensive language in tweets. They use a dataset collected by Warner and Hirschberg (2012). The solution is language agnostic and the model incorporated features from the users' behavioral data. A strong motivation for their choice of machine learning algorithm was the late tendency of using neural networks instead of Natural Language Processing (NLP) approaches. In addition, they argued that using pre-trained word vectors as Word2Vec was not feasible for their data from Twitter since the text may contain slang for the crucial words for detecting hateful speech. They describe cases of *word ambiguity*, when words can have different meanings based on context, and *spelling variations* as challenges that makes the NLP approach more ineffective than machine learning methods. Georgios et al. (2018) stated that challenges like these are important to be aware of when working with user generated text, such as tweets, and other online posts such as news articles. Their classification model used word-based frequency vectorization as input instead for pre-trained word vectors, which means that the words were indexed based on their appearance in the corpus and the value of the index for

each word was used as one vector to represent the tweet. This approach was a big advantage, because it made the model independent of the language. Further on, the model they presented consisted of four layers; input layer, hidden LSTM layer, hidden regular layer, and an output layer, and to avoid overfitting of the model, the training was allowed to run for maximum 100 epochs. After the training period the best model was identified. The model accomplished a F-Score of 93,2%, and outperformed current state of the art. Georgios et al. (2018) present interesting approaches, such as the use of multiple LSTM classifiers and the language independent model.

Repp (2016) presents work on regular supervised classifiers on natural language. His focus area was using data from the Twitter stream, representing the data as high-dimensional word vectors and inputting it to the classifier and clustering algorithm. The main goal for Repp (2016) was to detect news event from the Twitter stream, and his work is related to our work due the goal of detecting relevant data from open sources. With the use of Artificial Neural Network (ANN) classifier and mini-batch thread-clustering method he claimed he was able to achieve state-of-the-art performance. Repp (2016) experimented with different word representation methods with a focus on neural language models, and results with the use of average Word2Vec (AvgW2V) to represent the data. His data source differs from ours as he uses English data collected from Twitter (*Events2012 dataset*) which is annotated as categories of news events (e.g. sport events, armed conflicts & attacks events, disasters & accidents, etc.). The system classified the tweets as news event or not. The tweets classified as news event were then clustered to a category by being assigned to a thread, but this clustering was not relevant for our work as clustering is not a main target of this thesis. Nevertheless, his research contains important aspects relevant to our work; text representation method and the classifier. His goal was to detect news articles in the Twitter Stream, dissimilar from our goal which was to find relevant articles from news sources for the Norwegian Labour Inspection Authority.

Another approach using deep learning for classification purposes was proposed by Agarwal et al. (2017), which used it to detect paraphrases for sentences. They used a neural network architecture composed of Convolutional Neural Networks layers (CNN) and LSTM to compute the sentence similarities to label two sentences as paraphrase or non-paraphrase. Agarwal et al. present a similarity model which preserved the semantics of the sentences and based on pair-wise word similarities. As input they transformed the text to word vectors using Word2Vec. They evaluated their model on both noisy and clean dataset, the clean dataset from Microsoft (*Microsoft Paraphrase Corpus - MSRP*), and a noisy set from *SemEval 2015 Twitter dataset*. They got competitive results for the two datasets; on the SemEval dataset a F1-Score of 0.751, and on the MSRP dataset a F1-Score of 0.845. Even though their task was different than ours it was highly relevant for this thesis considering their approach with the Word2Vec text representation method and the deep learning architecture. Their joint deep learning model with CNN and LSTM is very relevant for our work and the architecture was a great inspiration when creating the model to classify work related documents. In addition, they used their classifier on both noisy and clean data.

Kim (2014) experimented with neural network models, and present research with CNN and pre-trained vectors for sentence-level classification tasks. They stated that simple CNN

with static vectors and little hyperparameter tuning achieved good results on several benchmarks. Their model was tested on multiple datasets, and two of them are; MR (movie reviews) detecting positive and negative comments and the TREC question dataset, which involves classifying questions into 6 different question types. Kim (2014) used the neural language model Word2Vec with pre-trained vectors with 300 dimensions as input to the classifier. They conclude that a simple CNN model performed well and gave competitive results. This article is of interest for this thesis as he performed a series of experiments with variations of the CNN model, and a CNN model is an interesting approach on our dataset along with the neural language models to represent the text.

Deep learning models have also been used for sentiment analysis, researched by Hassan and Mahmood (2018), among others. They presented a RNN and CNN architecture for sentiment analysis of short texts and proposed a solution that combines both recurrent and convolutional layers in one model and use it on top of pre-trained vectors. This approach is slightly different from the previous presented papers, because they utilized LSTM as a substitute for the pooling layer, to reduce the loss of detailed, local information and to capture long-term dependencies. They argue that the pooling layer is a reason for loosing details as the layers aggregate the results (e.g. average or max) and attempts to utilize a RNN as an alternative. Their proposed model therefore consisted of an Embedding Layer, which is the input, a CNN layer followed by a RNN (LSTM) layer. The model is evaluated on two benchmarks sentiment analysis datasets, IMDB (Movie Review) and SSTb (Stanford Sentiment Treebank). Pre-trained Word2Vec vectors were used in this approach as well, giving an indication that these neural language models are an appropriate choice as text representation for deep learning models. Their model performed well on the two datasets, and outperformed other methods, and is of interest for this thesis as they propose a combined CNN-LSTM neural network model that perform classification on top of pre-trained vectors. Hassan and Mahmood (2018) argued that the pooling layer is a reason for loosing much details when processing the text as explained earlier and achieved great results without it, and it is an interesting approach to test in this thesis.

2.2 Standard Classification Methods

In the previous paragraphs there have been presented different neural network classification approaches, but classification is a broad task that can be solved by using several methods. Balog et al. (2013) explained that Knowledge Based Acceleration systems (KBA) help humans to expand their knowledge bases, for instance a knowledge base as Wikipedia. This can be done by automatically recommend edits that is based on the incoming content stream. The process of identifying relevant content in the document (e.g. blogs or news articles) that imply modifications of e.g. entities, is a key step to be performed in a KBA system. This is a task called Cumulative Citation Recommendation (CCR) and is the challenge to retrieve documents for entities that are relevant to extend the content. Balog et al. (2013) proposed a multi-step classification approach with both two and three steps to address the centrality detection for a KBA. The dataset used for this purpose is called the *KBA Stream Corpus*, and

is composed of news, social media and linking (content from different URLs). The data is annotated with garbage and neutral (non-relevant) labels and relevant and central (relevant) labels. The first method was a binary classification approach, considering central and non-central documents, and an informal requirement for centrality was that the document relates directly to the target entity in e.g. a Wikipedia article. The second method firstly classified garbage and neutral against relevant and central, then separated the relevant and central documents from each other. The relevant documents were also of interest for the 3-step approach, but the relevant documents will only have an indirect relation on the target entity. Both methods share the same first component, that was the identification of entities mentioned in a document. This component can be directly connected to our thesis, since we are interested in detecting enterprises names in articles. Balog et al. (2013) tested two classifiers, J48 and Random Forest, and the comparison shows that Random Forest outperforms JF48 in most of the cases. The reason why the classification was done black box was to test the two methods proposed, not the performance of the classifier. Both of the methods performed very similar, which means that the 2-step approach was preferable due to the simplicity.

The work by Balog and Ramampiaro (2013) focused on the experimental comparison of classification versus ranking with supervised learning for CCR and is based on the work from Balog et al. (2013). They used a knowledge-based approach, given an entity as a target and incoming documents (e.g. social media or news) to compute a score considering the relevance for the text to the target. Their findings were that the Random Forest ranking approach achieved the best results for the CCR. The dataset they performed the research on was the *KBA Stream Corpus 2012*, same as Balog et al. (2013). Balog and Ramampiaro (2013) used earlier work as the feature set from Balog et al. (2013) as well. These features were the representation of their data and consisted of features associated to documents (length of document, source, language), entity (number of related articles), document-entity (number of times entity is in document, first occurrence of entity in document, etc.), and temporal attributes (average page views per hour, volume of page views for past hour, etc.), total of 23 features. They proposed the classification in a two-step and three-step approach as explained earlier. In their ranking approach each document-entity pair was assigned a numerical score, defining two settings; Central and Relevant+Central. In the Central-setting they excluded the documents labeled as relevant, the relevant documents were not excluded in the Relevant+Central setting. They used Random Forest for ranking the documents (as well as RankBoost and LambdaMART to compare the results). They concluded that their Random Forest Ranking was the best approach, delivering state-of-the-art performance. The work of Balog and Ramampiaro (2013) was related to our work as they desired to classify and rank the articles according to an entity, updating the information in a knowledge base. They used Decision Trees with a Random Forest approach which is an interesting machine learning technique. However, their large feature set required much preprocessing, and their already gathered dataset was a great advantage we do not possess. In addition, we do not aim to rank the data.

Besides deep learning algorithms and Decision Trees, another approach for text classification was the Naive Bayes classification method presented by Tang et al. (2016). Their focus was using class specific features as input for the classifier, i.e. each class had its own

subset of features used when categorizing the texts. To evaluate their work, they used two benchmarks; *20-Newsgroups* and *Reuters*. The 20-Newsgroup dataset has data collected from news posts and each post was assigned to a specific topic, resulting in a dataset of total 20.000 documents and 20 different topics. Reuters had some data containing multiple classes and these were removed since it was desirable only working with one class per document. From this dataset they were left with 8.293 documents with 65 unique topics and chose only the first 10 topics due to imbalanced dataset. They compared their results with classification approaches using non-class specific attributes. Tang et al. (2016) stated that they achieved state-of-the-art results with this approach and concluded that the performance scores increase when a high number of features was used. However, the result was a small leap down for the classifier not using class specific features.This work is related to our research due to the task of classifying text, and the datasets based on online posts which is similar to our approach of using open data sources. However, this approach required a great part of feature engineering to be able to represent the text to be classified, and since their results did not differ much from earlier state-of-the-art results, this approach was not desirable to use due the little gain of valuable information and the time-consuming part of feature engineering.

Another interesting paper that can be related to this thesis was the work of Ku and Leroy (2014). They presented their work of developing a decision support system to analyze and classify automated crime reports. This was to increase the effectiveness and efficiency for law enforcement agents to prevent further crimes. This system gives information to the decision makers for support for further actions, but not giving the final call of act. They gathered the raw data from crime reports, processing the data with lexicons and information extraction techniques. Further on, a similarity phase was created, both entity and similarity before it was used as input to the text classification using Naive Bayes. Finally, the output represented whether there are different crimes, the same crimes, or if a crime analyst is required to manually decide the result. This article presented an example of a decision support system that works in real life and served as a tool to increase effectiveness for law enforcement agents.

2.3 Other Related Work

The last paper we want to present is by Tripathy et al. (2014) and their approach for clustering tweets. They believe that clustering based on the theme would be a natural way of grouping the tweets. Wikipedia topic taxonomy was used to discover the themes in the tweet, and they proposed to use this theme together with the traditional word-based similarity metric to create the clusters. The approach they proposed to use was to map the tweet to a set of topics from Wikipedia, then the distance between two tweets was computed by graph distance on the topic graph from Wikipedia to investigate how close the topics are. After this the distance between the words was calculated, as both calculations were used in the final clustering algorithm. Bi-grams computed by a sliding window was used in the search after match in Wikipedia, and if the bi-grams do not have any match the individual unigrams were used. Graph distance measure and cosine similarity was used as distance measurements, and

simple K-means algorithm was used for clustering. The proposed approach was referred to as WIKI-kmeans algorithm. Their model was compared with the well-known TF-IDF model. The validation of the methods was done by conducting a user study, which is explained more comprehensive in the article (Tripathy et al., 2014), but essentially each user was asked if a pair of tweets was related or not. The answers were used to calculate the F-score for the clustering algorithm. The TF-IDF-kmeans got an F-score of 0.438 and the WIKI-kmeans algorithm got 0.523, which shows that the WIKI approach gave the best results. This article relates to our thesis as we are researching methods to represent our data, making it more applicable for the Norwegian Labour Inspection Authority. Clustering the relevant data is a possible approach and will make it feasible to extract the topic of each cluster.

2.4 Summary

The approaches presented in Section 2.1 used neural network approaches, and these models were composed of different variations of RNN, LSTM and CNN. Word embedding vectors was also presented, and Word2Vec seems to be one of the most used models. This may be because it is easy to access and at the same time provide good results. The only main drawback with a pre-trained model for our data, is the language and we may need to train the model on Norwegian language. We choose to focus our main classification research on the CNN and LSTM architecture presented earlier (Agarwal et al., 2017; Hassan and Mahmood, 2018). We also got inspired to evaluate the models with plain LSTM, CNN and regular models (Georgios et al., 2018; Repp, 2016; Kim, 2014) and to compare them with the joint CNN-LSTM model. Several of the neural network approaches used Word2Vec to represent the text, and we will investigate this model as well as other text representation methods, to see if there are other methods that can achieve competitive results. In addition, Hassan and Mahmood (2018) gave us an idea to experiment with a deep learning model without pooling, to see if this can provide better results as well. Decision Trees (Random Forest) have also been researched earlier (Balog et al., 2013; Balog and Ramampiaro, 2013) with focus on gathering information for entities, and even though this approach has achieved great results, this method requires feature engineering. Ku and Leroy (2014) and Tang et al. (2016) focused on text classification using Naive Bayes, but these approaches also require feature engineering. Based on these articles about Decision Trees and Naive Bayes we see that feature engineering is used to get better results, and although these papers have achieved good results, the deep learning approaches get competitive results without the features using pre-trained vectors (with some experimentation with additional features). Because of the small profit and the challenging task of feature engineering, we will not proceed with Decision Trees and Naive Bayes approaches. Further on, regarding other related work, Tripathy et al. (2014) used an approach with K-means clustering achieving good results and it is desirable to include this partition-based algorithm as well. We will not include their approach of connecting it to Wikipedia, but this approach may be interesting for future work. A summary of the related work is given in Table 2.1, presenting the methods they use, what purpose they want to research and the used dataset.

Table 2.1: Summary of related work

Work	Method	Task	Dataset
Repp (2016)	ANN	Detect news events in Twitter Stream	Events2012, by McMinn et al. (2013)
Georgios et al. (2018)	LSTM	Classify tweets as Neutral, Sexism, and/or Racism.	Twitter dataset by Warner and Hirschberg (2012)
Agarwal et al. (2017)	LSTM and CNN	Detect paraphrasing in sentences.	SemEval, MSRP
Kim (2014)	CNN	CNN with pre-trained vectors for sentence classification	MR (Movie review), SST-1, SST-2, Subj (Subjective dataset), TREC, CR (customer reviews), MPQA (opinion polarity subtask)
Hassan and Mahmood (2018)	CNN, RNN, and LSTM	Classifying model with RNN replacing pooling layer	IMDB and SSTb (Stanford Sentiment Treebank)
Balog et al. (2013)	Random Forest	Update information about entities in Wikipedia	KBA Stream Corpus 2012
Balog and Ramampiaro (2013)	Random Forest	Update information about entities in Wikipedia	KBA Stream Corpus 2012
Tang et al. (2016)	Naive Bayes	Using class specific features to classify text to several topics	20-Newsgroups, Reuters
Ku and Leroy (2014)	Naive Bayes	Classify crimes as equal or different	Internal Reports
Tripathy et al. (2014)	Clustering	Clustering Tweets and map to topics from Wikipedia	Data from Twitter collected by Yang and Leskovec (2011)

To the best of our knowledge, it does not exist a completely similar project to this thesis, and thus we have no directly competing results that we can compare with. The main reasons are the use of Norwegian texts and our main goal of detecting relevant articles for the Norwegian Labour Inspection Authority which have not been done before (not for Norwegian nor English texts).

Chapter 3

Background

In this chapter the background of this thesis is explained. The different parts necessary for our approach are described theoretically to give a deeper knowledge of the methods used. Description of the different methods of text preprocessing, namely text summarization will be presented in Section 3.1 and text representation will be described in Section 3.2. An overview of the classification approaches is described to understand what advantages and disadvantages the different methods possess, in Section 3.3. Section 3.4 addresses the task of clustering, introducing a partition-based algorithm. As our data is gathered from open sources, natural language processing is central for our task and will be explained in Section 3.5. This section encompasses part-of-speech tagging, sentiment analysis, named entity recognition and automated translation service. Finally, an overview of the different performance measures is presented in Section 3.6.

3.1 Text Summarization

Text summarization is used to create a summary of text documents. It is necessary for us to reduce the amount of texts into short and precise summaries, so the processing of the text gets easier. One of the main goals for this thesis is to find documents with relevant information for the Norwegian Labour Inspection Authority, and text summarization is a key method to reduce the amount of data necessary to process. A text summarizer finds the most important sentences from the text and ties them together in a short summary (Nenkova and McKeown, 2012). A summary makes it easier to compare the document with other text documents. TextRank and LexRank are two methods mentioned in Nenkova and McKeown (2012) for summarizing text and will be presented in the following sections.

3.1.1 TextRank

Mihalcea and Tarau (2004) presents TextRank, an unsupervised graph based algorithm extracting keywords and sentences. A graph based ranking algorithm is basically a way of

deciding the most important vertex in a graph. TextRank uses Google's well known PageRank ranking algorithm, but other algorithms such as Positional Function and HITS can also be used in the model. (Mihalcea and Tarau, 2004). When the algorithm creates a summary, the goal is to rank sentences and therefore each sentence gets a vertex. After this the relation between the sentences is defined, and if the sentences are measured to be similar an edge is drawn between the two sentences. The similarity between the sentences can be based on counting words of the same syntactic category or the number of common tokens between the lexical representation of the sentences (Mihalcea and Tarau, 2004). This results in a fully connected graph, which the ranking algorithms run through. The sentences are then sorted in reverse order of their score and the top ranked sentences are included in the summary (Mihalcea and Tarau, 2004). One of the advantages with TextRank is that the algorithm does not require any domain or language specific corpora, nor a deep linguistic knowledge (Mihalcea, 2004).

3.1.2 LexRank

Another algorithm developed to create summaries is LexRank (Erkan and Radev, 2004). LexRank is similar to TextRank, and it also uses an unsupervised graph-based approach. The algorithm uses Inverse Document Frequency (IDF) - modified Cosine to measure the similarity between two sentences, and the same value is used as weight of the graph edge. Erkan and Radev (2004) explains that their approach is based on finding the centrality of a sentence in a cluster, and then find the text worth inclusion in the summary. They define the centrality for a cluster based on the words it contains, and how relevant the words are. To be able to connect sentences to a topic, they continue by assuming that when a group of sentences are similar to each other, they are more related to the topic than other dissimilar sentences. Each sentence is represented as a n-dimensional vector, with a bag-of-words approach, where n is the number of words in the respective language (Erkan and Radev, 2004). They describe the computation of the sentence similarity by:

> *"For each word that occurs in a sentence, the value of the corresponding dimension in the vector representation of the sentence is the number of occurrences of the word in the sentence times the idf of the word. The similarity between two sentences is then defined by the cosine between two corresponding vectors."*
> (Erkan and Radev, 2004: p. 406)

The sentences with the highest centrality will be returned as the summary of the input text.

3.2 Text Representation

Preprocessing of text is an important task to perform before any classification methods can be applied (Aggarwal and Zhai, 2012a). Becker and Plumbley (1996) state that this part is important but difficult, since the raw data from various sources may be large and complex,

which can cause the *curse of dimensionality*. The classification process may already be time consuming, and therefore it is important to extract and represent the text before it is used as input to the classifier (Becker and Plumbley, 1996). Traditional language methods have been a major part of the phase of information retrieval, some of the methods are bag-of-words representation or frequency of terms. These models uses among other a word count approach; TF-IDF (*Term Frequency X Inverse Document Frequency*) (Salton and Buckley, 1988). Some drawbacks with these techniques may be sparse vector representation and that they do not take the context or semantics of a word into consideration.

Hassan and Mahmood (2018) states that in vector space words with similar semantics are close, avoiding the sparsity problem typical for traditional methods. Since the source of data contains natural language, representing text as vectors seems to be sufficient with focus on embedded words and by taking morphology into consideration. This introduces the neural language models *Word2Vec*, *GloVe*, and *FastText* we chose to explore as preparation for the classification approach. The *TF-IDF* representation is explained as a representation of text for the clustering algorithm. These will be explained in the following sections.

3.2.1 Word2Vec

The word vector model is named *Word2Vec* and is introduced by Mikolov et al. (2013a). This text representation method is based on learning the high-quality vectors of words from a big dataset (unsupervised). In addition, Mikolov et al. (2013a) claim that the representations of words are more than just predetermined dictionaries and rules of syntactic structures, it also shows that with algebraic operations it is possible to find the vectors closest related to one another. A known example is the operations of vector *"king"*, and by subtracting *"man"* and adding *"woman"* vectors it results in the vector for *"queen"* (Mikolov et al., 2013a). Mikolov et al. researched two different methods; skip-gram and continuous bag-of-words (CBOW). The skip-gram model uses the given sentence as a base of classification, it predicts the word based on the surrounding words in the sentence instead of the context. This differs from the CBOW model since CBOW tries to predict based on a context window. They concluded that the *skip-gram* model achieved the highest accuracy.

3.2.2 GloVe

The unsupervised learning algorithm GloVe *(Global Vectors)* is used for obtaining vector representations of words (Pennington et al., 2014). These words are represented with global vectors and by having the model trained on the non-zero entries of a global word-word co-occurrence matrix, it avoids the complexity of training on separate windows depending of the context or the sparse matrices of an enormous corpus (Pennington et al., 2014). This tabulates to see how frequent words co-occur with other words from a given document collection. The matrix is filled after only one iteration through the corpus. GloVe produces substructures and represents it by vectors. Other methods take the difference (distance or angle between vectors) of word pairs as a main evaluation method of the word representations. However, GloVe focuses on the differences of various dimensions rather than the

scalar distance of word vectors. Pennington et al. (2014) provides the following example as an analogy for the equation $king - queen = man - woman$:

> *"King is to queen as man is to woman"* (Pennington et al., 2014: p. 1532)

Pennington et al. (2014) proposed a method that uses statistics efficiently by weighted least square model that trains on the counts of global word-word co-occurrence. They state that they outperformed other models on analogies and similarities among words with their log-bilinear regression model for word representation on unsupervised learning.

3.2.3 FastText

FastText is a word representation method developed by Bojanowski et al. (2016). They propose a method of text representation by learning on character n-grams, and then represent the words by computing the sum of the n-gram vectors. The main focus for this model is to take morphology of words into consideration, which they claim some of the earlier propositions do not include. FastText is based on the general skip-gram model proposed by Mikolov et al. (2013b), but instead of using a distinct vector for the word they take the internal structures of a word into account. For languages with huge vocabularies, assigning each word to a distinct vector is a limitation. Furthermore, they explain that each word is represented as a collection of n-grams, an example from the paper of the word *"where"* with n = 3: *<wh, whe, her, ere, re>*. Bojanowski et al. (2016) state that:

> *"This simple model allows sharing the representations across words, thus allowing to learn reliable representation for rare words ... Ultimately, a word is represented by its index in the word dictionary and the set of hashed n-grams it contains."* (Bojanowski et al., 2016: p. 3)

Bojanowski et al. (2016) compare their method by taking different aspects into consideration; using human similarity judgment, analogy of the words, morphology representation, consideration regarding size of training data and the n-grams, and finally modeling of language. They conclude that they achieve state-of-the-art results.

3.2.4 TF-IDF

Term frequency - inverse document frequency (TF-IDF) is a weight metric that ranks the importance of a term in a document, and terms with high TF-IDF number implies a strong relationship with the document the term appear in (Ramos et al., 2003). TF-IDF is combined by term frequency (TF) and inverse document frequency (IDF). TF measures how often a term occur in a document and is calculated by taking the number of times term t occurs in the document divided by the total number of terms in the document. IDF measures how unique a term is in a collection of documents and penalizes frequents words and rewards rare words. IDF is calculated by taking the log of the total number of documents in the collection

divided by the number of documents the term occurs in. The TF-IDF weight is calculated by multiplying these two values. The TF-IDF computation of a term *t* in document *d* can be shown in the following equation.

$$w_{t,d} = (1 + log(tf_{t,d})) * log(\frac{N}{df_t})$$

3.2.5 Discussion and Conclusion

Since we are dealing with natural language processing, representing text as vectors with word embeddings seems to be a sufficient solution based on earlier work, and have achieved good scores (Repp, 2016; Agarwal et al., 2017; Kim, 2014). The text representation methods Word2Vec, GloVe and FastText, as explained in the previous sections, are all suitable for representing text depending on the situation. The Word2Vec model has become a well-known method and have achieved good results. On the other hand, Pennington et al. (2014) state that GloVe outperforms Word2Vec on the same vocabulary, corpus, training time and window size. This substantiates that both methods are proven to produce good results dependent on the situation.FastText is more or less an extension of the Word2Vec model, but the main difference is how the vectors are composed from each word, as explained in the sections above. To decide which text representation method to use, we decided to test all three methods in the classification model and select the method that produces the best results in the model.

3.3 Classification

Classification has been a major part of the study regarding central computational techniques, e.g. machine learning, information retrieval, and data mining (Aggarwal and Zhai, 2012a). Due to the huge amount of available data online, it is difficult to handle all this data manually. Categorizing the data we gather is necessary, and classify the articles as interesting for the Norwegian Labour Inspection Authority or not is a typical classification task. Learning a computer to behave automatically can be done in different ways, and two of these are *supervised* and *unsupervised* learning. In supervised learning the program observes an example of input-output pairs and learns how to map them. In unsupervised learning the program observes only the input and tries to detect patterns or useful clusters to learn the output (Russell et al., 2010). An example of supervised learning is when an algorithm is trained to classify images of persons and animals. In this example the training data will be images of persons or animals with the correct label. Unsupervised learning does not have such an answer key to tell the algorithm which label is correct for a respective image.

Because there exists a wide variety of classification algorithms using machine learning, we have chosen to explore three algorithms, used in earlier work that is described in Chapter

2. In the following sections the machine learning algorithms Decision Trees, Naive Bayes and Neural Network will be explained.

3.3.1 Decision Trees

Decision Tree is a machine learning algorithm, where the tree-based design contains nodes with attributes resulting in a final classification of the data in the leaf nodes (Russell et al., 2010). It takes a vector of attributes and will return a single output value. A decision tree uses the attributes which gives the highest information gain as a split-attribute (node) early in the tree, and the lesser information gain the later the node will appear in the tree. When the best split attribute has been computed, a connection with a predicate is made to the next node. Figure 3.1 illustrates an example (randomly generated) of a binary decision tree whether or not a person should buy a property. As mentioned earlier, it divides the data hierarchically

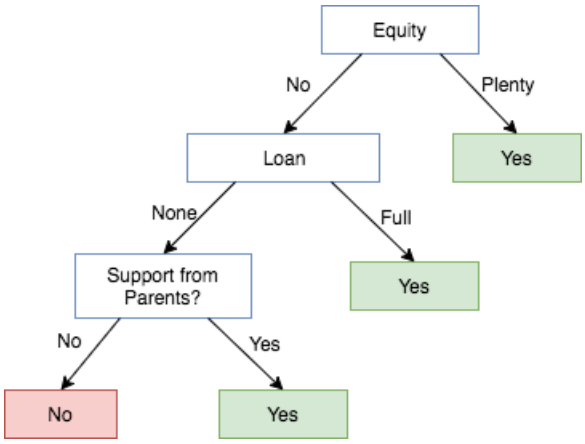


Figure 3.1: Binary decision tree example

based on whether or not the documents contain the word(s) or other features and is performed recursively until some conditions have been met (Aggarwal and Zhai, 2012a). The algorithm is based on a training set which creates the model, and a test set to verify the accuracy of the classification. Based on this, it follows that to create a decision tree we are dependent on a dataset with describable features.

There are some problems regarding the learning of the decision trees, one of them is overfitting, which is when the model is too adapted to the training set (Russell et al., 2010; Patel and Mistry, 2015). It also requires some time generating a tree. Another drawback with Decision Trees is if a mistake has been made at a higher level of the nodes, it may result in lower subtrees with incorrect classification patterns (Patel and Mistry, 2015). Furthermore, Russell et al. (2010) state that to apply the Decision Tree algorithm on various domains there are some problems that need to be solved. What if there are some attributes that are missing in a dataset? Sparse training set may occur on data gathered from the real-world. There will be a challenge to compute the information gain of an attribute when some of the data is missing. Regarding the type of attributes as multi-valued attributes, continuous and

integer-valued input and output attributes, these may cause problems with infinite number of representation models (Russell et al., 2010). Regardless, Decision Trees are possible to understand for the human eye and make the reasoning of the system possible to examine which is an advantage.

One way to solve the problem of choosing the best Decision Tree is using Random Forest (Liaw et al., 2002), which is an aggregation of the Decision Trees generated from a training set. This modeling approach it is more powerful than a single Decision Tree, because of the aggregation of the different trees into one final Decision Tree. It restrains the possibility for overfitting and bias errors (different classification output for the set of attribute patterns) due to the combination of different models. This will reduce the possibility of too adaptive trees. However, it may result in a huge size of used memory space.

3.3.2 Naive Bayes Classifier

The Naive Bayes Classifier is a probabilistic classifier and is based on a particular class of Bayesian Network, which is the conditional independence class. This is a conditional independent model, where the number of effects is directly influenced by a single cause (Russell et al., 2010). Before describing the Naive Bayes classifier, let's take a look at Bayes theorem which is the basis of probabilistic inference for most of Artificial Intelligence (AI) systems, and used by the Naive Bayes algorithm. Given $P(A \wedge B) = P(A|B)P(B)$ and $P(A \wedge B) = P(B|A)P(A)$ it is constituted that $P(B|A) = (P(A|B)P(B)/P(A))$. Russell et al. describes the specification of a Bayesian Network as:

> *"1. Each node corresponds to a random variable, which may be discrete or continuous.*
>
> *2. A set of directed links or arrows connects pairs of nodes. If there is an arrow from node X to node Y , X is said to be a parent of Y. The graph has no directed cycles (and hence is a directed acyclic graph, or DAG).*
>
> *3. Each node Xi has a conditional probability distribution* $P(Xi|Parents(Xi))$ *that quantifies the effect of the parents on the node."* (Russell et al., 2010: p. 511)

The conditional independence for the Naive Bayes is a simplified model (hence the name "Naive Bayes"). Furthermore, the full joint distribution is defined by when a single cause influences a set of effects directly, and given the cause, all effects are *conditionally independent*. Russell et al. describe it as:

> "$P(Cause, Effect_1, Effect_2, ..., Effect_n) = P(Cause) \prod_i P(Effect_i|Cause)$" (Russell et al., 2010: p. 499)

The model is based on the distributions of words in documents, with a *bag-of-words* assumption (Aggarwal and Zhai, 2012a), meaning it does not consider the position of the words. With this basis of information, Naive Bayes has widely been applied for categorization of

texts. Given a document *d*, classify *d* to one of the classes that are defined based on the content of the text. In this context, the *effect* variables are regarding if a word exists or not, and the classes we want to assign documents to is the *cause*. Hence, the category the document belongs to (the cause), will deduce the words the document contains (the effect). It is assumed that words appear in documents independently with frequencies set by the class for the document (Russell et al., 2010). Since the basis of this methods is conditional independence, if this assumption about independence is not met it will not perform very well (Patel and Mistry, 2015).

3.3.3 Neural Network

Machine learning consists of many branches and one of them is Artificial Neural Network (ANN). ANN is used to solve different problems, and one of them is text classification. An ANN program is presented with data and then the program learns from these training examples. The program can recognize patterns such as handwritten digits or other data (Nielsen, 2015). A neural network is composed of levels with different neurons, and these neurons are connected by links. A neuron is a node with one or more input variables, and one output variable. The output arcs are often depicted with more than one arrow as in Figure 3.2, but only implicates that the output can be accessed as input into other trailing neurons. There are different types of neurons, two of them are called *perceptrons* and *sigmoid* neurons. Perceptrons have input and output variables that are binary, while sigmoid neurons take any values between 0 and 1. The sigmoid neuron is commonly used due to the difference in output values which can be more nuanced (Nielsen, 2015). With this description of single neurons, an ANN can be explained. The network consists of different layers, and the first layer is called *input layer*. This layer only serves to process the input, there are no calculations performed. The last layer is called *output layer* giving the final classification. The middle layer is called *hidden layer* with neurons as explained above, and these neurons are neither input nor output neurons (Nielsen, 2015). This layer has a more complex design, as it is this layer that performs all the calculations. The arcs are labeled with an associated weight, which describes the importance of the input values (not shown in the Figure 3.2).

Figure 3.2 is an example of a neural network from Nielsen (2015) and shows that there are no links between the nodes in the same layer only to the neurons in the forward layer. The figure shows a three-layer network. The neurons to the left forms the input layer and the rightmost neurons forms the output layer. This network contains only one hidden layer. A network can consist of several hidden layers, and it is considered a *deep neural network* if it contains more than one hidden layer (Nielsen, 2015). In this specific network each node in the first layer is connected to the second layer and each node in the second layer is connected to the last. This makes the network fully-connected. The hidden layer will perform calculations based on the value received from previous layer, in this case the input layer, together with the weight associated with the value. Each link has a weight associated with it, and this link determines the strength of the connection (Russell et al., 2010).

In the hidden layers it is applied an *activation function*, which is a formula computing the influence of the value and the weight for the particular neuron (Goodfellow et al., 2016).

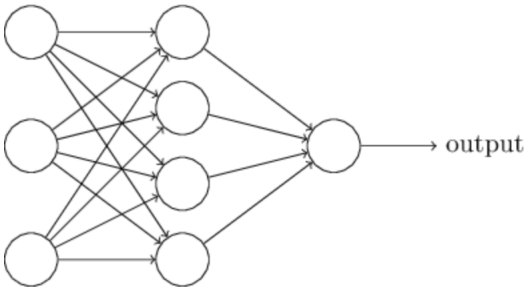


Figure 3.2: Neural network architecture example from Nielsen (2015)

There exist many different activation functions, depending on the use of the neural network (Russell et al., 2010). An example of an activation function for a Sigmoid neuron:

$$\frac{1}{1 + exp(-\sum_j w_j x_j - b)}$$

In this equation w_j is the weight and x_j is the value to neuron *j*. Bias *b* is a variable to decide how hard it is for the function to be activated (Nielsen, 2015). With this description in mind the objective of a neural network can be achieved, namely *learning*. Using supervised learning the AI can compare the final output value with the label of the correct classification. By adjusting weights and biases it can be trained to approach the correct solutions. One method to optimize the adjustment of the weights is by using the algorithm *Gradient Descent*. Gradient descent is an algorithm based on a cost function as a way of detecting the gradient of the function. Its common analogy is that it is a hill-climbing algorithm (Russell et al., 2010) where it is desirable to find the global minimum. The approach of this method is to minimize the loss until convergence (calculated by the cost function), and it is done by adjusting the weights of the weight space and going to the downhill neighbor coordinates to move towards the correct classification model (Russell et al., 2010). The cost function is often referred to as the objective or loss function (Nielsen, 2015)

In intricate domains the neural network methods may become useful and maintains continuous and discrete values well (Patel and Mistry, 2015). However, the training period may take some time (depending on the size of the training data and number of layers and neurons) (Patel and Mistry, 2015; Russell et al., 2010). Neural Network is also prone to overfitting, but there are several ways to reduce this. (Srivastava et al., 2014) described a method performing *dropout* on the neural net, resulting in a thinned model where some units are randomly dropped. The reason dropout might be ideal to use is because networks with many neurons and hidden layers may be slow, and combining results is difficult. Dropout drops a number of units at random, and by doing this it prevents the units from adapting too much, thus reducing the chance of overfitting (Srivastava et al., 2014). There are different types of neural networks, such as Feed-Forward Networks and Recurrent Neural Networks, which will be described in the following sections.

Feed-Forward Network

A Feed-Forward Network *(FFN)*, is a network that only have connections in one direction, it is a directed acyclic graph *(DAG)*. This means that there are no loops in the model (Russell et al., 2010; Nielsen, 2015), all data is going from the left side to the right as in Figure 3.2. There are different types of FFN, and a known model is the Convolutional Neural Network *(CNN)*. CNN have often been used for image processing and have achieved good results of identifying faces and traffic signs in pictures (Karn, 2016). In addition, it has accomplished good classification results when it comes to natural language processing and sentence classification. When a CNN processes an image, its main goal is to get features from the original image. Cong et al. (2017) states that:

> *"Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data."* (Cong et al., 2017: p. 191)

Britz (2015) addresses the use of CNN for natural language processing. Instead of having an image as input (e.g. presented as a matrix of grey scale values), imagine sentences represented as matrices, where each row is the word, and the number of columns is the dimension of the vector (typically word embeddings). Having a sentence with 10 words and an embedding of 300, our *image* will be a matrix of 10x300 (Britz, 2015). Further on, by using a *small square* on our matrix a *filter* can be used to create the desired feature map. Different features from an image could typically be feature maps where the edges or curves are detected (Karn, 2016). Some differences between text and image processing in CNN are that parts of phrases do not always have a close spatial relationship and may be several words apart, unlike an image where the parts of an object are relatively close (Britz, 2015). After the feature map is created, pooling may be applied to reduce the dimension of the map but still maintain the important information. Pooling subsamples the input (Britz, 2015; Karn, 2016), meaning it aggregates values (e.g. computing average or selecting the maximum score) to a matrix with a lower dimension. Britz (2015) state that CNN have been proven to be fast and efficient when processing the texts.

Recurrent Neural Network

A Recurrent Neural Network *(RNN)* differs from a FFN because it feeds the output back to the input, thus separating it from a DAG. This allows the information to persist (Russell et al., 2010). Nielsen (2015) state that a RNN is closer to how the human brain actually works and can solve problems more efficiently than a FFN. A RNN uses back propagation in the training process for updating the weights in each level (Georgios et al., 2018). A RNN can support *short-term memory*, giving the opportunity that previous inputs affect the computations. Russell et al. (2010) states that this is the main difference from RNN and FFN. A regular RNN does not capture long term dependencies, meaning output from an earlier time does not influence the current computations (Chung et al., 2014). This may be crucial to preserve semantics in text since semantically close words may occur several words apart.

Long Short-Term Memory *(LSTM)* is a special kind of RNN and is capable of learning long-term dependencies and remembering information for a long period of time (Graves, 2012). Chung et al. (2014) states that traditional RNN does not save the content more than one epoch, while LSTM manage to decide if it should preserve it for a longer time. If a unit in an LSTM model decides to keep a feature, it is capable to conserve it for a long sequence, resulting in a model which captures long-distance dependencies.

3.3.4 Discussion and Conclusion

The classification methods Decision Trees, Naive Bayes, and Neural Network Classifiers, as described in the previous sections, have earlier been used for categorizing text. Decision Trees applies a *hierarchical division* of the data based on the labeled features in order to partition the data into classes. It determines which class it most likely belongs to based on the attributes defined for the instance of the text. The dataset is divided into subsets according to the attributes, hence it requires feature engineering. The Bayes Classifier (Aggarwal and Zhai, 2012a; Patel and Mistry, 2015) is a probabilistic model based on features for the classes and it uses a bag-of-words representation. The content of the documents and the respective classes gives the posterior probabilities to classify the text. The third method was the Neural Network Classifiers, a network consisting of hidden layers with potential complex design to perform calculations. It has been used widely in different domains and is dependent of proper text representation.

Decision Trees and Neural Network are prone to overfitting as explained earlier, which means that they may be too adapted to the training set. Furthermore, if a subtree in Decision Tree is created wrongly it leads to trailing errors in the final model. Regarding continuous variables, Decision Trees does not handle it very well as opposed to Neural Network. When it comes to the Bayesian method it may not be useful once the assumption of conditional independence is not met but work well on textual and numeric data when the condition is satisfied. Neural Network may require some time when training the model, but once it is trained the testing can be done relatively fast (Patel and Mistry, 2015). A comparison done by Russell et al. is that although Naive Bayes learns well, the Decision Tree they modeled performed better. However, Naive Bayes does not need a very big dataset to be able to create the model, so the time complexity is not necessarily very large, and the model adapts well to scenarios with noise. This method, as well as the Decision Tree method, are dependent on feature engineering on the data to create the labeled dataset upon which to train the model.

The feature selection and how to represent the text before used as input to the classifier have been described by theory in Section 3.2. This section also reasons for the use of neural language models rather than the traditional methods. The main reason was the complex task of feature engineering versus what have been obtained as results, based on earlier work which achieved good results using neural networks without manual feature engineering. Both Decision Trees and Naive Bayes are traditional statistical methods, using features as term frequencies or bag-of-words models. This differs from neural networks as it considers the spatial relationship of words. Due to the neural language models chosen as text representation method, ANN was chosen as the classifier in this thesis. A challenge choos-

ing this approach is finding the optimal number of hidden layers and the number of neurons in each of these layers, which will be explored in Section 4.4.3. However, the traditional methods were not an unfavorable choice, but due to the text representation method chosen (neural language models) and our interest in exploring neural network to classify text for the Norwegian Labour Inspection Authority, it was interesting to research this approach.

3.4 Clustering

Clustering is a favorable task to apply when dealing with unstructured data, because clustering tries to solve the problem of finding groups with similar objects in the data and can be very useful when organizing documents to improve the retrieval (Aggarwal and Zhai, 2012b). The results are highly dependent on the noisiness of the input data (Aggarwal and Zhai, 2012b). An example is words like ”or” and ”the” and Norwegian words such as ”det” and ”at” is not very useful for the clustering algorithms. It is therefore important to remove such words to improve the clustering results.

There exists a large variety of clustering algorithms and the main approach is the partitioning-based method. Liu (2007) state that K-means is the best known partitional clustering algorithm, and this algorithm is one of the most widely used methods due to its effectiveness and simplicity. This algorithm iteratively partitions the data into k clusters based on a distance function. K-means is initialized by placing k centroids and calculating the distances to the other points in the dataset. Each unit is then assigned to the nearest cluster and new centroids are calculated by computing the average of the assigned points. This is done in each iteration, and ends when the clusters have no or minimal change in the division of the clusters. The computational complexity of the algorithm grows linearly with the number of cluster and points but is sufficient due to the small number of iterations necessary in order to converge (Aggarwal and Zhai, 2012b). When the algorithm converges it is finished. However, K-means are sensitive to the initial seed for the centroids, as they are placed at random, but by running the K-means several times, comparing the results and choosing the best outcome may reduce this problem (Paulsen and Ramampiaro, 2009).

Another disadvantage with this algorithm is the requirement to pre-set the number of clusters. The number of different topics from a dataset with natural language is not easy to estimate, this requires knowledge of the content and the number of data that are related. However, there exists an approach called the *elbow method*. This method is a visual way to decide the true K numbers of clusters. The approach is to start with K = 2 and keep increasing the number with 1, and in each step the method calculates the clusters and the cost of the clustering. A distance metric that can be used to compute the distortions for the cluster is *Hamming metric* (Norouzi et al., 2012). This metric calculates ”how many changes” necessary to transform one sequence (e.g. vector) to another vector.

The number of true K is discovered when this is repeated until the cost drops dramatically and reaches a point where it does not converge any more (Kodinariya and Makwana, 2013), see Figure 3.3 for a visualization of the results of the elbow method. The elbow method is based on the principle of *Ockham's razor* (Skiena, 2017). It is based on choosing the

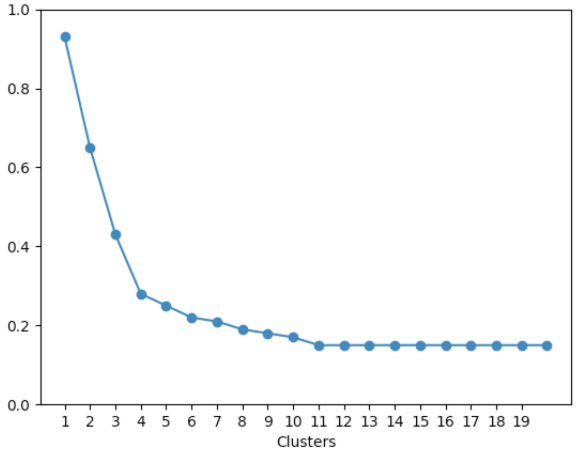


Figure 3.3: Elbow method graph for K-means example

assumption (e.g. graph) which is the simplest and is consistent with the data, as there are numerous possible consistent solutions and choosing the simplest eliminate the complex task of evaluating all the possibilities (Russell et al., 2010). In Figure 3.3 the number of K is presented at the horizontal axis, and as it appears from the graph the optimal amount of clusters is approximately 4, which is where the difference in the distortion converge (Norouzi et al., 2012).

3.5 Natural Language Processing

Natural Language Processing (NLP) is a computational approach towards analyzing text (Russell et al., 2010). NLP is an area of research that explores how computers can understand and manipulate speech or text generated by humans. NLP applications include a wide range of areas of study, such as machine translation and text summarization. (Chowdhury, 2003). Russell et al. (2010) state that almost all the pages on the world wide web are containing natural language, and for that information to be useful the machine needs to process the ambiguous and messy language that we use. Language is difficult, the number of generated sentences is infinite, and is constantly changing (Russell et al., 2010). Text classification is significant for NLP applications, such as information retrieval, email categorization and document classification (Hassan and Mahmood, 2018). Extracting information from unstructured text generated by humans is a complex task, and several methods have been developed to process language for further computational use. Methods such as part-of-speech, sentiment analysis, named entity recognition and automated translation services will be explained in the following sections.

3.5.1 Part-of-speech

One of the first processes in an NLP application can be to determine the morphological structure and nature of the words (Chowdhury, 2003). An example of a process doing this is part-of-speech (POS) tagging. A POS tagger is software that receive text as input, and the output is each word assigned with parts of speech, such as verb, nouns, adjectives, prepositions, etc. This process is very helpful when processing natural language in a text. An example of the result of a part-of-speech tagger for the sentence *"John likes the blue house"* is shown in Table 3.1.

Table 3.1: Part-of-speech tagging example

Word	**POS-tag**
John	*noun*
likes	*verb*
the	*determiner*
blue	*adjective*
house	*noun*

Some problems with part-of-speech is ambiguous words. Take a look at the sentence *"We can can the can"* (Güngör, 2010), where the word "can" is mentioned three times and should be assign three different word classes (auxiliary, verb, and noun). For a human eye it is easier to detect the differences with syntactic structure, domain knowledge, and common sense. The computer needs to learn how to deal with all this information. Language change, as mention earlier, and some of the words occurring may be unknown. This causes problems for the rule-based approaches, which will not know how to tag the new word (Güngör, 2010). These problems are crucial when creating a part-of-speech tagger, however, in this thesis we will aim to use an already trained model for POS-tagging.

3.5.2 Sentiment Analysis

Sentiment analysis is the field of study that analyzes e.g. people's opinions, attitudes and emotions towards certain entities such as products (Liu, B., 2012). Liu and Zhang (2012) state that with the explosive growth of social media, such as blogs and social networks, affects organizations and individuals to use opinions from these platforms to make their decisions. An opinion can be expressed about many different things, such as a topic or an organization. Further in the article they explain that for blogs and products, the opinion holders are usually the author. In relative to news articles, as these often state an organization or persons opinion. Liu and Zhang (2012) define opinion as a negative or positive sentiment about an entity or an aspect of the entity from an opinion holder.

Sentiment analysis were briefly investigated in this thesis, but although a sentiment analysis was a relevant field of study, it was decided not to proceed further with an analysis of the web documents. The main reason it was considered unnecessary, was because both positively and negatively charged text would be of interest for the Norwegian Labour Inspection

Authority. The only requirement was that the text contained relevant information. Another reason was the findings we did after the brief investigation. We were not able to find an existing tool, that did not require training, with good support for the Norwegian language. At last, the scope of this thesis did not include developing a sentiment analysis model.

3.5.3 Named Entity Recognition

Information Extraction (*IE*) is an area of research within NLP, which aim to extract structured information automatically when considering semi- and unstructured texts (Piskorski and Yangarber, 2013). Information Extraction contain the subtask Named Entity Recognition (*NER*) which is used to identify entities in a text. It may be persons, locations, and organizations, and extracting these objects from texts may give rise to further computational techniques (Jiang, 2012). Jiang (2012) presents two methods for named entity recognition; the rule-based and the statistical approach. The rule-based approach consists of a set of rules learned automatically or manually decided. The words in the text is represented by a feature set and are then compared to the rules. The statistical learning approach is based on machine learning, which aim to solve the problem of sequence labeling (Jiang, 2012). Given a set of observations (often represented as a vector), it is desirable to assign a label to each of these observations to determine the entity.

3.5.4 Automated Translation Service

The different information extraction methods and machine learning methods are well covered for the English language. However, these computational techniques for the Norwegian language have a long way to go to be competitive with the English support. Because of the scope of this thesis, which do not cover the development of a Norwegian Sentiment Analysis model (this have been a master thesis earlier (Øye, 2015)) nor a Named Entity Recognition system, we decided to examine the possibility to translate Norwegian texts in to English. Google Translate is a widely used automated translation service. There were some initial concerns regarding the use of Google Translate due to both grammatical errors and with which success rate it is capable of preserving the semantic meaning of a text, and based on academic papers that examined the translation from another languages to English, there were some concerns. The overall conclusion was that Google Translate was not grammatically error free and need to be used with cautioum (Balk et al., 2012; Groves and Mundt, 2015). Regardless, we conducted random sampling when testing Google Translate on the Norwegian language, and approximately 7 out of 10 sentences were well translated, i.e. the text was error free and the semantic was preserved. This will be further described in Section 4.5.3. The articles from Balk et al. and Groves and Mundt are somewhat *older* papers, respectively 2012 and 2015. As presented in a paper from 2016, Google Translate adopted the Neural Machine Translation *(NMT)* as explained in Wu et al. (2016), which is an automated translation service with an approach of end-to-end learning which improve the results.

3.6 Performance Measures

Evaluating the performance of the tasks in this thesis was necessary, and measures such as accuracy, precision, recall and F1-Score are commonly used (Sokolova and Lapalme, 2009). To be able to measure the performance of different experiments conducted in this research, *F1-Score* is an applicable measurement. The formula is shown in Equation 3.3. It was desirable to use this measure rather than plain accuracy (Equation 3.4) because it takes the number of false positives and false negatives into account (i.e. the test set is asymmetric).

The F1-formula is a harmonic mean and is based on values from *precision* (Equation 3.1) and *recall* (Equation 3.2), considering false negatives and false positives unlike the regular *accuracy* formula, which is the number of total correct labeled elements of the total set. The reason it was desirable to combine precision and recall was because separately they both evaluate important aspects of the retrieved elements. Precision focuses on how many of the retrieved elements that are actually correct, while recall considers how many of the retrieved elements that are relevant out of the total relevant elements in the collection. The Confusion Matrix (Table 3.2) describes the different possible scenarios whether or not the classification of two classes, X and Y, are correctly retrieved (Sokolova and Lapalme, 2009).

Table 3.2: Confusion matrix explanation

True Positive (TP) If an element is labeled to class X and is classified as X.	**False Positive (FP)** If an element is labeled as Y but it is classified as X.
False Negative (FN) If an element is labeled to a class X but is classified as Y.	**True Negative (TN)** If an element is not labeled to class X and neither classified as X.

Precision

Precision and recall are calculations in the field of information retrieval to measure how good a method retrieves the desired information (Ting, 2010). Precision takes the total number of relevant elements (true positives) that are retrieved out of the total elements that are retrieved (true positives and false positives), i.e. precision measures how many of the retrieved elements that are relevant.

$$\frac{TP}{TP + FP} \tag{3.1}$$

Recall

Recall is based on the total number of relevant elements (true positives) retrieved out of the total number of relevant elements in the data collection (true positives and false negatives), i.e. recall measures how many relevant elements that are retrieved of the total amount of relevant elements in the corpus.

$$\frac{TP}{TP + FN} \tag{3.2}$$

F1-Score

F1-Score considers both false negative and false positives because of the combination of precision and recall. It computes the weighted average, and therefor differs from the regular accuracy which only considers the correct classified elements of the total elements in the dataset.

$$2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.3}$$

Accuracy

As mentioned, accuracy take the correct labeled data (true positives and true negatives) out of the total amount of data (true positives, false positives, false negatives and true negatives). The formula for accuracy is shown in Equation 3.4.

$$\frac{TP + TN}{TP + FP + FN + TN} \tag{3.4}$$

Chapter 4

Approach

This chapter encompasses the approach for detecting deviations in work environments in enterprises. Section 4.1 will give an overview of the approach, and the specification of the requirements for the tasks from the Norwegian Labour Inspection Authority will be presented in Section 4.2. The source of data with the decision of the choice between social media or web documents is elaborated in Section 4.3. How we summarized, represented and classified the articles to detect relevant web documents is described in Section 4.4, along with the research of how to extract the topics with clustering. Our approach for named entity recognition and the problems we met with Norwegian texts will be described in Section 4.5, and the final system will be presented in Section 4.6.

4.1 Overview of Approach

To get an overall overview of the approach in this thesis, the task can be divided into four individual problems. The first problem is to extract information and create a data base with Norwegian texts. The second problem is to pre-process the data, a task that is performed to give the best possible input to the classification model and involves text representation and text summarization. There are also tested two different text summarization algorithms in Section 4.4.1. In order to choose the best text representation method, several methods are presented in Section 4.4.2, and further in the process they are tested with the different classification models. The third problem is to build a classification model and train the model on the Norwegian dataset, so it is able to classify the information as relevant for the Norwegian Labour Inspection Authority. To get the best possible solution for this problem, several neural network models were tested with different text representation methods. The model we propose is explained in Section 4.4.3. The final part of the approach is to cluster the relevant information and extract a topic from each cluster, and is presented in Section 4.4.4.

4.2 Specification of Requirements

The Norwegian Labour Inspection Authority presented several tasks they wanted explored in order to detect relevant articles, and it was therefore desirable to prepare an overview of the requirements. As a result, a requirement specification was created. For each requirement it was assigned a technical approach in order to get a better understanding of what methods that was applicable for each task. The requirements are presented in Table 4.1 along with the corresponding technical approach.

Table 4.1: Requirement specification overview

Requirement specification	**Task**	**Technical approach**
RS1	Investigate selected social media sources and online newspapers. Use these sources to look for interesting stories and useful information about companies.	No direct technical approach
RS2	Investigate and extract information from the sources that is found the most informative and usable.	Information retrieval
RS3	Perform search queries based on relevant terms within the Environment, Health and Safety (EHS) field, to extract more relevant information.	Information retrieval
RS4	Investigate methods to filter relevant and not relevant information	Classification
RS5	Identify Norwegian company names in the selected sources	NER
RS6	Identify the event in the source that was considered relevant e.g. work accidents	Clustering

The main goal was to investigate if the use of social media and online newspapers could be useful for the Norwegian Labour Inspection Authority. It was therefore important to start wide and exclude sources in the process that proved not to be accessible enough. As it emerges from Table 4.1, the specification requirements *RS1*, *RS2* and *RS3*, a big task in this thesis was to gather and extract data to be able to get a sufficient data base. Further on this data was used as foundation for testing several methods, among them machine learning methods for text classification (*RS4*). To detect the main content and extracting the event that caused the relevance of the text was also a desirable task (*RS6*) and the technical approach applicable was clustering. *RS5* addresses the task of extracting the name of the enterprise in unstructured text, and NER is a favorable task in this scenario. As an extension of *RS3*, we created a short list of highly relevant words together with the Norwegian Labour Inspection Authority which we used in our search after relevant documents, the list is presented below.

- Arbeidsulykke (*eng*: work accident)
- Sosial dumping (*eng*: social dumping)
- #metoo
- Arbeidsforhold (*eng:* working conditions)
- Arbeidsmiljø (*eng*: work environment)

The specification of requirement will be addressed in the approach proposed in the following sections.

4.3 Source of Data

Establishing a source of data was necessary for further processing. At the beginning of this research the Norwegian Labour Inspection Authority presented different types of open sources they considered interesting, such as news articles and social media. The first step was to investigate different types of sources, and to narrow the scope of the thesis we decided to begin with focusing on the social media platforms *Facebook* and *Twitter*, thereafter we examined online news papers.

4.3.1 Social Media and Online News Articles

Facebook is the social media platform with the most traffic in Norway and there are over 3,4 million registered Norwegian users[1]. Twitter is another frequently used platform in Norway, with over 1,1 million registered Norwegian users[2]. The number of registered users together with previous experience the Labour Inspection Authority have had, was the main reason of the choice to investigate Facebook and Twitter. Another interesting source of data was online news papers, and in Norway there exist a large selection of papers all over the country which publish new articles online every day. The police department also publish highly relevant updates, and there does also exists several online documentaries that addresses important events. The reason for researching news articles was due to some restrictions encountered with the social media, which will be explained in section 4.3.1. The different sources of data will be discussed further in the following sections, along with the final choice of data sources.

Social Media

The main micro blogging services we examined were Facebook and Twitter. Facebook provides a Developers API[3] that makes it possible for people to extract and use some of the data

[1] `https://www.ipsos.com/nb-no/ipsos-some-tracker-q118`
[2] `https://www.ipsos.com/nb-no/ipsos-some-tracker-q118`
[3] `https://developers.facebook.com/`

from Facebook. Connecting to this source is expensive, which makes it difficult for students. Without this access, data from Facebook is hard to extract. Twitter also provides an API for developers. They offer different types of data streams; Search and Realtime API[4]. The Realtime API offers realtime streaming of tweets and was the most interesting Twitter source. During the test period we discovered that also this source required payment as well, which forced us to change API. The Search API did not require payment for a small amount of data per minute and was therefore suitable for gathering data as they allowed 180 calls every 15 minutes[5].

Norway is a small country, and the amount of available data on social media was less than expected. Twitter was the platform with the most available data, but it is not as much used in Norway as Facebook[6]. Facebook is a more closed platform, and one reason to this is the fact that user can set restrictions to who can see their data. Data from social media also have various quality, and one of the challenges we met were excessive use of abbreviations. For instance, a tweet has a limit of 140 characters[7], and this may lead to several word contractions in a tweet to express as much as possible. Irony and ambiguity are difficulties when processing natural language; equal words and sentences may have different definition in different circumstances (Georgios et al., 2018). Another drawback of using social media as a source of information was spelling mistakes. These difficulties can lead to noise and sparsity in the data, which lead to need of more complex calculations to be able to process the natural language.

Online News Papers

It was desirable to investigate online web documents as there have been earlier scenarios were online news papers have published articles of interest for the Norwegian Labour Inspection Authority. Online news articles are one kind of web documents with publications related to current events[8]. For instance, it was an article regarding a fatal accident at a work place[9] and another article where the Norwegian Labour Inspection Authority took action after a news paper revealed a fraud[10]. The news papers respective websites may be updated regularly, and with journalists and editors creating and updating articles makes room for a potentially major area of platforms for data retrieval. The various news papers in Norway have open websites but some of the papers requires payment for certain articles, creating some complications for the retrieval. Another type of web documents available are updates from the police department in Norway. An advantage by investigating the publications from the police department is hopefully minor use of clickbait titles (titles that are misleading or

[4] https://developer.twitter.com/en/docs
[5] https://developer.twitter.com/en/docs/basics/rate-limiting
[6] https://www.ipsos.com/nb-no/ipsos-some-tracker-q118
[7] https://developer.twitter.com/en/docs/basics/counting-characters
[8] https://en.oxforddictionaries.com/definition/online_newspaper
[9] https://www.nrk.no/trondelag/omkom-i-dykkerulykke-pa-froya-1.6441011
[10] https://www.vg.no/nyheter/innenriks/i/MgRlzK/arbeidstilsynet-slo-til-mot-det-norske-kartselskapet-etter-vg-avsloering

provocative[11]). Other interesting documents are documentaries that covers a subject with factual records[12].

Discussion

As explained in the above, Facebook was dismissed as a data source due to the restricted access. Twitter, on the other side, seemed to be a relatively open platform. Twitter was therefore explored further as a source of data, and we experienced both advantages and disadvantages with this platform. The main advantage we experienced was the easy extraction of data. The only main restriction we discovered early in the process were the lack of access to the Realtime API, which forced us to gather data based on search queries. We therefore chose to use the top 200 most used Norwegian words as individual search terms, inspired by Kvamme (2015). The idea was that this would cover a wide range of tweets and extract a representative amount of data with different themes and from several different users. The data were gathered over a small period of time, and all tweets were Norwegian. After the data gathering period we started to annotate the data together with the Norwegian Labour Inspection Authority. During the annotation we discovered that most of the Twitter data were not relevant. In order to improve the data we retrieved, we tried to search with only relevant queries, containing words like ”Arbeidsulykke” (*eng:* working accident) from the list of words presented in Section 4.2. This approach resulted in several hits, but the results were mainly from online news papers, referring to their article about the theme, and not individual users referring to a workplace. We experienced that few users wrote something where they openly commented their workplace or anything other of interest to the Norwegian Labour Inspection Authority. Since we needed a large amount of both relevant and not relevant data for training and testing of the model, it was decided to not proceed with Twitter as a source.

Since we experienced that different online news papers posted about their articles on Twitter, and several of these articles were relevant for the Norwegian Labour Inspection Authority, the exploration of Twitter did lead us on to another path. Due to the results from the relevant queries on Twitter and already proposed as a relevant source from the Norwegian Labour Inspection Authority, we decided to explore online news papers. News articles have the possibility to contain more relevant information than a tweet, since there is no limitation of the length as opposite to tweets with the limit of 140 characters. Tweets do not need to be news or concerning anything special, but an online paper's main purpose is to provide recent news to the readers. So, if a significant accident happen at a workplace and a journalist discovers it, chances are high that the situation will be published. However, the online news papers are not imposed to write about cases concerning workplaces, so if something else happens of even higher value for the papers, these cases will most likely be prioritized. Another challenge with online news papers is that the information is spread on several different sources, and not only on one platform.

To be able to get enough data to train the model, we decided to proceed with online news papers as a source. The gathering of data from online news papers is presented in the next

[11] `https://www.urbandictionary.com/define.php?term=clickbait`
[12] `https://en.oxforddictionaries.com/definition/documentary`

section.

4.3.2 Data Gathering

Social media had little accessible data, as described earlier, and this led to the retrieval of information from online news papers. Articles from several news papers was gathered, resulting in extraction from multiple sites. Neither of these pages had an API we could use, and this required us to extract the data separately from the individual pages before adding it to one database. This resulted in the use of a tool named Parsehub[13], which is a web scraping tool that were useful for the data mining phase. The format and semantic structure of these pages were varied; text length, batch of text versus several paragraphs, the use of videos in the article and the various use of preamble. This led to a phase to assemble the data into one format and make it ready for annotation. The different sources we decided to gather information from can be seen in Table 4.2.

Politilogg was considered as a web document and not a social media source even though the text was retrieved from Twitter. This was because the public saying from the police may be more official than any other post from regular people. In addition, we examined the articles on the official page for the Police Department in Norway, as the police articles is considered as highly relevant source of interesting data in addition to news articles. To decide which online newspapers to extract articles from we based the choice on the top five towns in Norway with the highest population in 2017[14]. This resulted in news papers from Oslo, Bergen, Stavanger, Trondheim, and Drammen. To expand the number of sources, we also decided to extract information from NRK, which is a Norwegian state-owned broadcasting company[15]. NRK also provides a page called NRK Dokumentar, consisting of documentaries with more depth and information than regular news. Aftenposten posted a series of interesting articles related to corruption and economic crime. The journalists behind the work was Einar Haakaas and Siri Gedde Dahl, and the series of articles are called Grå Økonomi[16]. This series of articles was considered relevant for the Norwegian Labour Inspection Authority and have earlier revealed big criminal networks and illegal work in Norwegian work life.

A challenge using online newspapers was the payment restriction. Due to the lesser use of paper versions, the newspapers had to start requiring payment for some of the articles to maintain the income for editors and journalists[17]. This resulted in only extracting the title successfully without the main content for some of the articles, as it was unavailable due to the payment restrictions. Another challenge was the gathering of data from the news articles resulted in a major part of irrelevant articles. It was therefore necessary to collect some additional data to increase the number of relevant articles, resulting in articles from

[13] https://www.parsehub.com/

[14] https://www.ssb.no/beftett/9

[15] https://www.nrk.no/informasjon/veien-til-norsk-rikskringkasting-1.11093657

[16] https://www.aftenposten.no/emne/Gr%C3%A5_%C3%B8konomi

[17] https://journalisten.no/2016/03/Aatte-av-ti-nettaviser-i-usa-tar-betaling-fra-leserne

Table 4.2: Sources of web documents overview

Site	Content Type	URL	No. of articles
Aftenposten	News articles	*https://www.aftenposten.no*	139
Bergens Tidende	News articles	*https://www.bt.no/*	94
Stavanger Aftenblad	News articles	*https://www.aftenbladet.no/*	68
Adressa	News articles	*https://www.adressa.no*	75
Drammens Tidende	News articles	*https://www.dt.no*	46
NRK	News articles	*https://www.nrk.no*	181
NRK Dokumentar	Documentaries	*https://www.nrk.no/dokumentar*	84
Politiet	Articles from the police department	*https://www.politiet.no/aktuelt-tall-og-fakta/aktuelt/*	155
Politilogg	Tweets gathered from police departments posted on Twitter (unofficial)	*https://www.politilogg.no*	56
HMS-portalen	Collected EHS-related articles	*https://www.hms-portalen.no*	114
Fri fagbevegelse	Collected EHS-related articles	*https://www.frifagbevegelse.no/*	8
Other	Different relevant articles retrieved by manual search on Google	Different sources	28

HMS-portalen, Fri Fagbeveglse, and several articles from singular sites (Other in Table 4.2) retrieved by Google search using the list of words presented in Section 4.2.

Annotation

When the dataset was gathered it was desirable to evaluate if an article was relevant or irrelevant for the Norwegian Labour Inspection Authority. With a total of 1048 articles in the dataset, 80% of the data were labeled by us, and the remaining 20% were annotated by the Norwegian Labour Inspection Authority. The first dataset was used as the training set for the classification, and the last part annotated by external people was to be used as a test set to avoid a biased test result. The labels for the binary classes were *relevant* or *irrelevant*, annotated as *R* and *I* respectively.

The criteria set for annotating an article as relevant was whether the text mentions any irregularities at a workplace and unfair behavior for the employees. Articles concerning accidents at work, insecure work places, poorly paid employees, #metoo situations, unreasonable methods concerning resignation or other aspects of employees' well-being, were annotated as relevant. Examples of themes and words that was considered relevant are shown in the list in Section 4.2. An article was not necessarily annotated as relevant even though it was concerning an enterprise, e.g. articles about stock exchanges, updates about companies and general information which do not affect the working conditions. The relevant articles were approximately 20% of the dataset which is relatively low, signifying that the dataset is imbalanced.

4.4 Detecting Relevant Web Documents

Section 4.3.2 explains how web documents was retrieved. With a proper dataset established, the approach of detecting relevant web documents could be researched. The approach of summarization and representation of the text will be done before used as input to a classifier. Which classification model we want to propose will be researched, and the clustering of the relevant documents and topic extraction for each cluster will be presented. The different phases of detecting relevant web documents will be described in the following sections.

4.4.1 Text Summarization

To be able to input articles with long text in the neural network classifier, it had to be summarized into a brief review. Cases were the text is shorter have occurred in earlier work with classifiers for tweets, where a maximum length of the text based on number of words in a general tweet (Georgios et al., 2018). With the restriction of 140 characters in a Twitter Status Update it is limited how many meaningful words in a sentence that can be identified (on average 10 to 11 words[18]). However, since we were dealing with articles there were no limit on how many words it may contain. Therefore, text summarization was applied on the data to get a fixed length of input features for the classifier. If the original text consisted of 50 words or less, it was not summarized.

We tested two graph-based text summarization methods; TextRank and LexRank as described by theory in Section 3.1. Both of the methods existed as libraries in Python; *summa*[19] (TextRank) and *sumy*[20] (LexRank). The different summary methods supported the choice of how long the summary should be. We sat a limit of 50 words, because in addition to longer text we dealt with shorter text containing at the minimum four words. This was to not get to big leap from the longest to shortest texts. How we padded the shorter sentences will be explained in Section 4.4.2. LexRank had an option to set a given number of sentences desired in the summary extracted, and in TextRank it was possible to specify the number of

[18] `https://blog.oxforddictionaries.com/2011/05/17/short-and-tweet/`
[19] `https://pypi.python.org/pypi/summa/0.1.0`
[20] `https://pypi.python.org/pypi/sumy`

words desired in the summary output. If the result of the summary exceeded 50 words, it was trimmed at exactly 50 words.

To decide which text summarizer method to use, the training set for the classifier were summarized by the two methods and then evaluated. We sampled 11.9% of the training set and afterwards we compared the results manually to see which one who preserved the semantics best. Of the sample, TextRank and LexRank summarized 24% of the data equally good, LexRank outperformed TextRank on 55% of the data and TextRank were best on other 13%. Both of the summary methods were not sufficient for the remaining 8% of the sample. Some of the reasons that a bad summary was created was when the original text was poorly written, obviously creating a bad extraction. Another general scenario appearing in summaries that were not sufficient was when the original text contained quotations. The quotes got heavily weighted and the summaries extracted these sentences without any context. In total, LexRank had 79% and TextRank had 37% sufficient summaries of the sample. The evaluations were performed by us, focusing on the semantics of the summary and if the summary managed to reproduce the main topics in the full text.

There have been compared different text summary methods in Python earlier[21]. A typical comparison measure of summary methods is the ROUGE-N measure, which is a set of metrics to evaluate automatic generated summary in comparison to a golden summary[22]. Based on our manual labeling and comparison of the training set and earlier comparison with the ROGUE-1 measure, LexRank was chosen to summarize the text.

4.4.2 Text Representation

The results from the summarizer described in Section 4.4.1 created texts with a maximum of 50 words. If a text was of shorter length than 50 words, it was padded with the value *0.0* (Georgios et al., 2018). In addition, to be able to represent the text we investigated three neural language text representation methods, as explained in Section 3.2; Word2Vec, GloVe, and FastText. The model trained by the FastText and Word2Vec were a pre-trained model retrieved from Github, while the last model, GloVe, were trained on our dataset extracted from online web documents. The reason why we chose to test two pre-trained models, was because we wanted to keep the vocabulary as general as possible. By training the model on our own, the model can become too adapted. The three text representation models were then used as a dictionary to access the vectors. For each word in a text, the vector for the respective word was retrieved and added to the list with the other word vectors for the given text. This resulted in three datasets that was to be experimented with by the classifier. Each text was represented as a list of vectors, and each vector with a dimension of 300. If the word did not exist in a dictionary, it was replaced with a vector of 300 dimensions with the value 0.0.

[21] https://rare-technologies.com/text-summarization-in-python-extractive-vs-abstractive-techniques-revisited/

[22] http://www.rxnlp.com/how-rouge-works-for-evaluation-of-summarization-tasks/#.WqEss5M-d-U

To decide which representation model to be used we tested the three methods as input for the classifier, the results will be shown evaluated in Sections 5.1.1 and 5.2.1. How the different models were obtained will be explained in the following sections.

Word2Vec

Word2Vec is a method trained by a unsupervised feed-forward network, as described in Section 3.2.1. We retrieved a file of pre-trained word embeddings for the Norwegian language trained by Word2Vec from a Github account named *Kyubyong*[23]. This model contained 50 209 words and each word have a dimension of 300, trained on data from Wikipedia. These vectors were made by using the skip-gram model.

GloVe

As explained in Section 3.2.2 Glove is an unsupervised learning algorithm used for obtaining vector representations of words. Pre-trained word vectors for Glove is only available in English, and it was therefore necessary for us to train an own model for vector representation in Norwegian. Stanford NLP Group had published the code for training a model[24] in any preferable language. The training of this model was done on the training part of the dataset, which means that our GloVe model contained 32285 words available as 300-dimensional vectors. We chose to have the vectors represented with 300 dimensions due to both Word2Vec and FastText had pre-trained vectors with a dimension of 300.

FastText

FastText is a method using word embeddings represented by vectors as described in Section 3.2.3. Facebook's FastText github site supplied pre-trained vectors for many languages using the skip-gram model described in Bojanowski et al. (2016), including Norwegian[25]. They provided files with pre-trained word embeddings (Bojanowski et al., 2016) containing 515 788 words. The model was trained on text from Wikipedia, and each word embedding had a dimension of 300 as well.

4.4.3 The Classifier

The task of the classifier was a binary classification of web documents as either relevant or irrelevant, which we aimed to solve by deep learning. Keras API[26] was an applicable framework for neural network modeling, and it was used to research the different models described in Section 3.3.3; Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN),

[23] https://github.com/Kyubyong/wordvectors

[24] https://github.com/stanfordnlp/GloVe

[25] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

[26] https://keras.io/

and regular Feed-Forward Network (FFN) with Dense[27] layers. We investigated models consisting of layers of the same type (models with pure CNN, LSTM and Dense layers), and the combined model (LSTM layers and CNN layers in the same model). Based on earlier work (Agarwal et al., 2017; Hassan and Mahmood, 2018) we propose a joint CNN-LSTM model as the final model. This model will be presented in the next section, and compared with other models in Sections 5.1.1 and 5.2.1. The proposed neural network model was also tested without the pooling layer (Hassan and Mahmood, 2018), but the results did not improve from the model with the pooling layer. It was therefore decided to proceed with a pooling layer.

Architecture

The neural network model we propose is a joint CNN-LSTM neural network, a deep learning model consisting of four layers. The architecture of the neural network we propose are:

- *The Input Layer:* The data encoded with the embeddings trained by Word2Vec, Glove and FastText is managed by this layer. The input shape was the number of words in a text which was 50, and the number of dimension for each word vector were 300. No activation function was used for the input layer.
- *The Hidden CNN Layer:* The filter size for the convolved feature map was 4, and the number of neurons was set to 100 for the dimension output (i.e. the 100 filters with a feature map of size 4). The activation function for this layer was ReLu. We used max pooling with a *pool size* of 4 on the generated feature map.
- *The Hidden LSTM Layer:* The number of neurons for the output dimension was set to 200 and the activation function selected was Tanh.
- *The Output Layer:* This layer consisted of 1 neuron to classify the data as relevant or irrelevant, and Sigmoid was chosen as the activation function.

These layers constituted the final model. The number of neurons and which activation function to be selected for each layer was chosen by experimenting with different combinations and selecting the ones achieving the best results. The model is illustrated in Figure 4.1.

Hyperparameters

To create the model for the neural network the hyper parameters was defined. Hyperparameters are variables which may provide optimizations of result when training and evaluating the model[28]. Parameters that needed to be set for the models were the optimizer, loss function, batch size and number of epochs. When selecting the optimizer, the choice was based on the related work done by Georgios et al. (2018). Gradient descent is a common algorithm used

[27] Keras define Dense layers as "just your regular densely-connected Neural Network layer" `https://keras.io/layers/core/`

[28] `http://colinraffel.com/wiki/neural_network_hyperparameters`

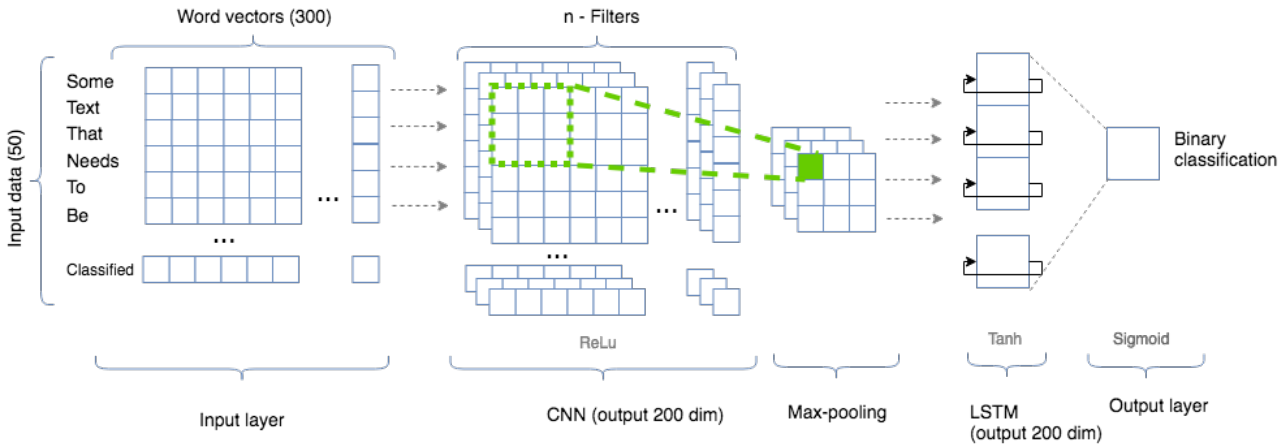


Figure 4.1: CNN-LSTM architecture.

to optimize neural networks, as described in Section 3.3.3. *ADAM* is an extension of gradient descent algorithm (Kingma and Ba, 2014) and was chosen as the optimizer. The loss function (also known as objective function) was set to *binary crossentropy* since we were only working with a binary classification problem (relevant or irrelevant). The loss function calculates the difference between output the model generates with the correct label in the training set. The batch size is the number of data of the input that should be propagated through the network, and the batch size for this network was set to 32. When experimenting with number of epochs the training should perform, we experienced the same behavior as Georgios et al. (2018). It normally stabilized after 40 to 50 epochs but ran maximum 100 (as Georgios et al. (2018)) epochs to avoid overfitting when experimenting with the model.

Training

With the architecture of the model and the given hyperparameteres as presented above, the training part began. Even though the number of epochs was set, an early termination criterion was specified to avoid overfitting. This criterion stops the training of the model if there are no or minimum change after a number of epochs, i.e; when the quantity being monitored has stopped improving the training stops. We sat the criteria for stopping to if the accuracy value did not change with a maximum of 0.0001 within 15 epochs, the training would terminate as the improvement has stagnated. When training the model to get the final results, the number of epochs was set to 50 and the early termination criteria was paused to get a consistently number of epochs for all the models tested. Another way to try to improve the model before the termination criteria was met was to reduce the learning rate when the accuracy stagnated. If the accuracy score does not change over a period of five epochs, the learning rate is reduced by a factor of 0.01.

The model was saved after each epoch under the training phase to be able to retrieve the best model. When the training was done, the epoch with the highest F1-Score was retrieved as the best model and tested with the test set. It was desirable to choose the best model

after training on a given number of epochs to be able to get a F1-Score as close as possible to a global maximum. We chose not to decide the best model based on evaluating the best F1-Score after each run due the risk of only getting a local maximum. To reduce overfitting we applied Dropout layers to the model (Dropout is described in Section 3.3.3). We tried to reduce overfitting even more by applying regularizers in the model, Keras supported weight regularizers which penalize the adjustment of weights during the training period. However, even though this reduced the overfitting, it produced worse results for the testset, so it was not included in the final model.

Summary

When the experimental phase to create the neural network model was done, we came to a conclusion regarding the number of neurons and activation functions applicable for our model. In addition, the hyperparameters for optimizing the model was set. With the final model specified, the testing of the three word-embeddings models Word2Vec, GloVe and FastText could be done. The F1-Scores were computed to compare the results of the three neural language models with the four deep learning models we wanted to research. In addition, it was interesting to observe the distribution of the F1-Score for the two classes. As described earlier, the dataset was imbalanced; the Relevant-class proportion in the dataset is smaller than the Irrelevant-class. This may be reflected in the results. The results and evaluation of the testing period of the classifier will be presented in Section 5.1.

4.4.4 Clustering the Relevant Data

Grouping together the relevant data could make it easier for the Norwegian Labour Inspection Authority to detect similar violations in work places and extracting the themes of the documents could give a superior overview over which articles dealing with specific topics. Before any clustering could be performed, the text had to be represented with features upon which the grouping could be based on. Then the number of clusters had to be decided for the K-means algorithm. Finally, the topics were extracted using a naive traditional method based on frequent terms. This approach will be described in the following sections.

Text Preprocessing

Before vectorizing the documents, the sentences were preprocessed. The full article was used as input to the clustering algorithm, not the summarized text that was used in the classifier since the input for the clustering did not need to be fixed.The first step was to remove stopwords, which is frequent common words in a language and it typically is connectives and prepositions (*and*, *but*, *in*, *to*, etc.)[29] as these words are frequently used in a text but is not explanatory for the content. It does not exist a formal list of stopwords, so an arbitrary list

[29] `https://en.oxforddictionaries.com/grammar/word-classes-or-parts-of-speech`

was retrieved from github[30]. Then each word in a sentence was labeled with the respective part of speech tag (adverb, verb, noun, preposition, etc.). This was done by a library called *RippleTagger*[31], which supported several languages including Norwegian. Since we wanted to cluster articles describing different scenarios and events, we filtered the words which was tagged as a *noun* and *proper noun*. By doing this we excluded words as verbs among other things, because words as *"doing"* was not an interesting term to cluster by. Finally, each word got stemmed to be able to cluster the morphology of a word as one term. This was done by a library from NLTK called *NorwegianStemmer*[32]. After the stopword removal, part-of-speech (POS) tagging and stemming was done, the text could be vectorized using TF-IDF values. *Scikit learn*[33] is a python library which had a TF-IDF vectorizer, creating vectors for each document in the corpus.

Choosing the Number of Clusters

When the corpus had been preprocessed the number of cluster for the *K-means clustering* had to be decided, as explained in Section 3.4. Therefore a method called the *elbow method* was performed do be able to decide the number of clusters for our dataset. The elbow method was done by performing the K-means clustering on the dataset of a range of clusters from 2 to 20. Then, by computing the distance between the clusters and finding where the reduction of the distortion stagnates, gave a good indication for how many clusters were needed. The metric used for computing the distortion was the *Hamming* measure, as explained in Section 3.4. The elbow method was run ten times, computing the average "elbow". This was done because the results differed somewhat for each run due to the random initial values for the centroids. By running the elbow method ten times we achieved an average number of six clusters (see Figure 4.2a and 4.2b). Our dataset did not have obvious groups of data and some deviations occurred resulting in the elbow method did not produce a perfect elbow every time. The elbow method is based on the principle of Ockham's razor (as explained in Section 3.4), and it was applied on the elbow method graph to get the simplest solution, making it easier to find an elbow by computing the exponential regression for the graph. This is also shown in Figure 4.2a and Figure 4.2b.

K-means Clustering

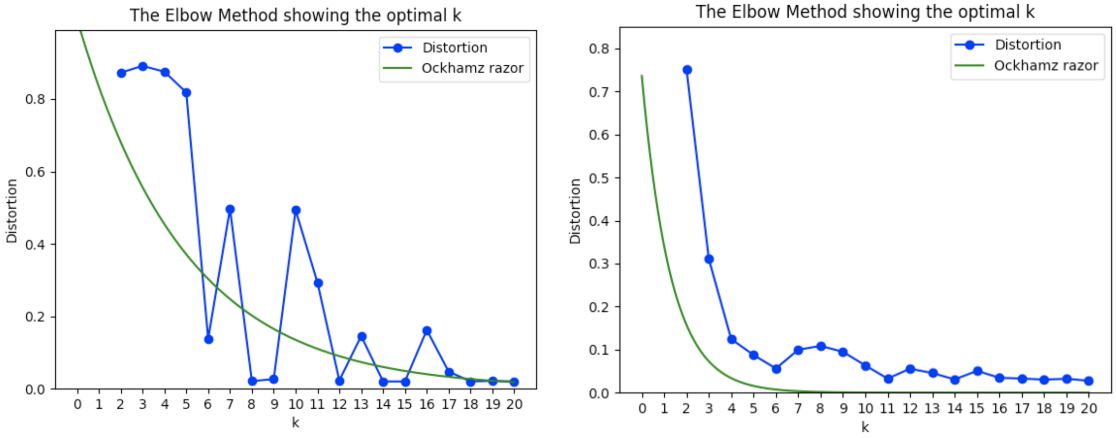
Tripathy et al. (2014), the article presented in Section 2.3, presented a clustering approach using a set of topics from Wikipedia combined with the K-means method. This approach achieved better results than the TF-IDF method, but our conditions differs from theirs, since we want to use topics only of interest for the Norwegian Labour Inspection Authority. We

[30] https://github.com/stopwords-iso/stopwords-no/blob/master/stopwords-no.txt

[31] https://github.com/EmilStenstrom/rippletagger

[32] https://www.nltk.org/_modules/nltk/stem/snowball.html

[33] http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html



(a) K-means elbow on training set.

(b) K-means elbow on on test set.

Figure 4.2: Elbow method

therefore decided to proceed with the TF-IDF approach, in lack of a domain specific knowledge base for the Norwegian Labour Inspection Authority. This approach will on the other hand be interesting for future work, and will be discussed further in Section 6.2.

When the input data was preprocessed, and the number of clusters was decided, the K-means clustering algorithm was performed on the dataset. The K-means method we used was from the *Sklearn*[34] library for Python. The method performs the K-means algorithm by initializing k centroids (defined by the elbow method) and calculating the distance for every TF-IDF vector for an article to the centroids. Then the centroids were adjusted to in the middle of the associated clusters and then calculating the new distances for the data. This was set to be done 300 times. For the algorithm it was set max of 10 runs, which is the number of times the K-means methods is run with different centroid seeds. This is the default setting for the Sklearn K-means method. The results with the best output was chosen (this was supported by the Python library). Since the data was clustered by TF-IDF vectors for each document, the vectors were high dimensional. Plotting this in vector space was a complicated task and hard to visualize for the human eye, therefore the vectors was reduced to two- and three- dimensional vectors by using Sklearn decomposition method *Principal Component Analysis*[35]. It transforms the high dimensional vectors to a vector with lower dimensionality by using Singular Values to reduce the dimensionality linearly. The result of the plotting will be shown in Section 5.1.

[34] `http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`

[35] `http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`

Finding a Topic for each Cluster

Grouping the data into different clusters was done successfully, but it was also desirable to give each cluster a topic to define what was the most relevant terms for each cluster. We investigated two different approaches, one by the values from the K-means method from Sklearn using the TF-IDF values, and another one we developed using document frequencies. We chose top three terms with highest score for each of the methods proposed, and the results will be shown and discussed in Section 5.1.

4.5 Named Entity Recognition

Named Entity Recognition (NER) is used to recognize entities in text, as described in Section 3.5.3, such as company names. This information extraction could be useful when classifying web documents, as well as increase the quality of the documents marked as interesting. We therefore wanted to research the possibility to extract names from the Norwegian and English languages. The register of enterprises in Norway is described in Section 4.5.1. Section 4.5.2 discussed which NER method for English texts that achieved the highest F1-Score and Section 4.5.3 explains the two approaches tested for extracting names in Norwegian texts. A summary is given in Section 4.5.4.

4.5.1 Brønnøysundregistrene

Brønnøysundregistrene strives to trustworthy maintain the digital registers in Norway, in addition to be a dependable source of information[36]. They offer free and open source registers with information about the different enterprises in Norway. They also provide a tool that allows search in *Enhetsregisteret*[37,38], which is a register with recorded enterprises. This tool is however a pilot and is not a complete solution. Another way to access the information from the register is to download the complete dataset. This dataset gives access to all registered companies with accompanying information such as organization number and number of employees. A lookup in *Enhetsregisteret* contains much information about Norwegian enterprises.

4.5.2 English Text

For named entity extraction in English texts there were three tools we decided to explore; NLTK, Stanford CoreNLP, and Polyglot. *NLTK* (Bird et al., 2009) is a Python library that performs part-of-speech tagging and entity detection[39] for the English language. *The Stan-*

[36] https://www.brreg.no/

[37] http://data.brreg.no/oppslag/enhetsregisteret/enheter.xhtml

[38] https://www.brreg.no/om-oss/oppgavene-vare/alle-registrene-vare/om-enhetsregisteret/

[39] http://www.nltk.org/index.html/

ford Core Natural Processing Language (Manning et al., 2014)[40] also performs named entity recognition for English texts. *Polyglot* (Al-Rfou et al., 2015) is a natural language pipeline with methods for named entity recognition. These tools aim to detect names of persons, locations and organizations, and their ability to do this will be discussed in the following sections.

Approach

To decide which Named Entity Recognizer that should be used, we tested a dataset with the three NER tools described above. The dataset was a collection of English texts retrieved from kaggle.com[41] where the named entities were already labeled. Every row in the dataset represented a word with a POS-tag, and it was given a respective tag whether it was named entity or not. The tags that were interesting in this research were *geo* - Geographical Entity, *org* - Organization, *per* - Person, and *gpe* - Geopolitical Entity. The remaining tags in the dataset, *time, event, artifact, and natural phenomenon*, were not considered relevant. An assumption that was made when processing the dataset were about adjacent words. A name consisting of multiple words were split over several lines, for instance, the part with ”President Bill Clinton” in the dataset were represented as three lines: (President; NNP; B-per), (Bill; NNP; I-per), and (Clinton; NNP; I-per). Therefore, the directly adjacent named entity tagged rows were concatenated to one named entity. The ”NNP” tag after each name is the part-of-speech tag which indicates if it is a proper noun (singular)[42]. This narrowed the possibility for a high accuracy score due to ”North America” will be one concatenated word instead of ”North” and ”America” when *north* is not a unique name in the English language, e.g. ”North America”, ”North Africa”, ”North England”, etc.

Results

We focused on which tool that acquired the highest performance score (*F1-Score* as described in Section 3.6) when extracting named entities from the dataset. To show the differences of the extraction results among the NER tools, examine the following sentence:

> *WH chief of staff John Kelly says he will ”absolutely not” apologize for comments about Rep. Frederica Wilson*

This text contains the named entities *”WH”* which is an abbreviation for the *White House*, furthermore it mentions *John Kelly*, and *Fredrica Wilson*. With a total of three named entities, NLTK scored 0.5, Stanford scored 1.0, and Polyglot scored 0.67. All entities were considered, not only the name of enterprises as this will be covered later in this thesis, i.e. it was not taken into account whether a named entity was labeled as a person, location, nor an organization. If there were no entities in a text, and the tools confirmed that there were no

[40] https://stanfordnlp.github.io/CoreNLP/

[41] https://www.kaggle.com/abhinavwalia95/entity-annotated-corpus/data

[42] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

entities the result would be a perfect score. If one part of an entity was recognized by one of the tools, it was considered as successfully recognition of an entity. If the correct answer in the annotated dataset was ”President Bill Clinton”, the extracted bigrams ”President Bill” and ”Bill Clinton” will be approved. Otherwise, if ”Bill Clinton” is the correct answer both ”Bill” and ”Clinton” must be extracted to be approved. The average scores for the NLTK, Stanford, and Polyglot for the English dataset from Kaggle can be shown in Table 4.3.

Table 4.3: Performance measures for named entity recognition on English text

NER method	Precision	Recall	F1-Score
NLTK	0.936	0.890	**0.906**
Stanford CoreNLP	0.915	0.801	0.839
Polyglot	0.841	0.694	0.742

NLTK scored the highest average level of accuracy on the English dataset. With the highest values for precision, recall, and F1-Scores (0.936, 0.890, 0.906 respectively) NLTK was considered as the best tool. With some difference Stanford comes second with a F1-Score of 0.839. Polyglot was the tool with the poorest results of the tools mentioned. The precision score of Polyglot was fine, but with a recall score under 0.7 the results were not optimal.

4.5.3 Norwegian Text

Section 4.5.2 concludes that it was NLTK which achieved the highest F1-Score for name extraction for the English language. In addition to extract names from English text, we wanted to research a method to extract company names from Norwegian texts as well. Since all the data relevant for the scope of this thesis was Norwegian, it was important to investigate extraction of named entities from Norwegian texts. Named Entity Recognition is a useful computational technique, but to the best of our knowledge, the existing NER tools we investigated were not sufficiently trained for Norwegian texts. As the main goal for this thesis was to find interesting information for the Norwegian Labour Inspection Authority, a method to extract registered company names were worth exploring. We investigated two different approaches to extract company names from Norwegian texts and in the sections below the approaches and challenges of the process is explained.

Approach Using Enhetsregisteret

The first approach we tested were lookups towards the Enhetsregisteret. The main idea was to see if the article contained words with match in the Enhetsregisteret, because then we would know if the text mentioned a Norwegian registered company. We decided to download the whole dataset from Enhetsregisteret, explained in Section 4.5.1, and use the .csv file instead of the pilot. The first step of the process was to makes sure the data was compatible with utf-8. After splitting the words, each word in the web document was compared to the names

in the .csv file. The comparison of the names was done in two different ways, using partial and exact match:

> **Partial match** means that the name in the article does not need to have a full match in the .csv file. If a company name is *"Bear Toys AS"*, the only matches will be of the words "bear", "toys" and "AS" separately in the web document or combined. This means that the number of matches most likely will be very high because the word "bear" can be written in any other context then a company name. There may also be several other companies registered with matching words in the web document.
>
> **Exact match** only match words when they are exactly the same. This require the full name of the company to be mentioned in the text and not only parts of it.

Using partial match resulted in varying result. To the best of our knowledge, Enhetsregisteret do not have any strict name policy, which means that a registered name can be almost anything. We experienced that words like "skal" (eng: "shall") was used in several registered company names, and this word is also often used in news articles. This means that for each article, several companies were extracted as a match. Adding words like "skal" to the stopword was not an option, because such words can be used in companies that may be of interest for the Norwegian Labour Inspection Authority. The partial match did provide some advantages, because a web document does not always mention the fully registered company name. For instance, the "AS" part of the name can be excluded. However, due to our experiences we realized that a partial match was not going to give us any good results, as it gave us hundreds of names. At the same time, exact match gave us the challenge that the whole company name had to be mentioned. We therefore decided to test another approach using Google Translate and already existing NER methods for the English language.

Approach Using Google Translate

As explained in Section 3.5.4 Google Translate is an automated translating service with an approach of end-to-end learning. It is easy to use, but the result is very varying as it may not be 100% correct translations, losing important semantics for the sentence for some scenarios. We wanted to explore this translating service to get English text, mainly to explore if using an already existing NER tool was possible.

In this approach, all the web documents from the dataset were run through Google Translate and then the translations were evaluated manually. The results of the translations were diverse; some web documents were translated properly with all content and semantics intact, while others were missing context and semantics. An overall opinion of the translation was that the results were inadequate and not sufficient enough to be used as a trustworthy source to translate full texts. However, for our purpose the only words that was necessary to investigate was the company names. We therefore extracted all the web documents that contained company names and translated these. The main purpose of this task was to investigate if the company names were preserved and not translated. Translation of the company names resulted in 81% good translations, which means that in 81% of the situations the translator

did not translate the company name. After this, NLTK were chosen as tool, because this tool scored the highest level of accuracy in the test conducted in Section 4.5.2. The satisfying results after running the NLTK method on the translated texts was less than 70%, which may be because the syntactic structure of the sentences have been compromised and the named entities were more complicated to extract for NLTK.

4.5.4 Summary

To be able to recognize a company name in unstructured text is of great value for the Norwegian Labour Inspection Authority. The approaches presented in the previous sections ended with varying results. The main challenge was the language, as Norwegian is not a sufficiently supported language among the existing NER tools we explored in this thesis. The tool that provided the highest level of performance measures, on the English dataset from kaggle.com, were NLTK with a F1-Score of 0.906. Based on our test, if the texts were written in English, NLTK would be the best choice. Even though the main language for the data in this thesis was Norwegian, it was desirable to test English texts as well due the possibility to use Google Translate to translate Norwegian texts into English texts. This approach used an Automated Translation Service to translate the Norwegian text, and it was desirable to use the NER tool with the highest F1-Score. The results from the approaches tested on Norwegian texts were not satisfying, but it proved possible to extract some of the company names. Due to the results from the approach we tested with Enhetsregisteret and Google Translate were not sufficient enough, and with the scope of the thesis in mind, it was decided not to proceed with NER in the final solution.

4.6 Final System

After presenting the methods which composes the approach, it is time to present the final system that filter out relevant news articles for the Norwegian Labour Inspection Authority. The complete approach is presented in Figure 4.3, and a brief explanation of how the components works will be described in the following sections. Note that the system is not connected, as the scope of this thesis did not include developing the final decision support system. The components work individually. The Figure 4.3 show how each individual component could work together in a final system.

4.6.1 Source of Data

The input data is the dataset consisting of articles from different Norwegian online news papers, as explained in Section 4.3. The data was extracted by using Parsehub and saved in a TSV-format with the columns title and the text. The whole dataset consists of 1048 articles which is annotated as relevant or irrelevant for the Norwegian Labour Inspection Authority. The dataset was imbalanced, with approximately 20% Relevant-labeled and 80% labeled as Irrelevant.

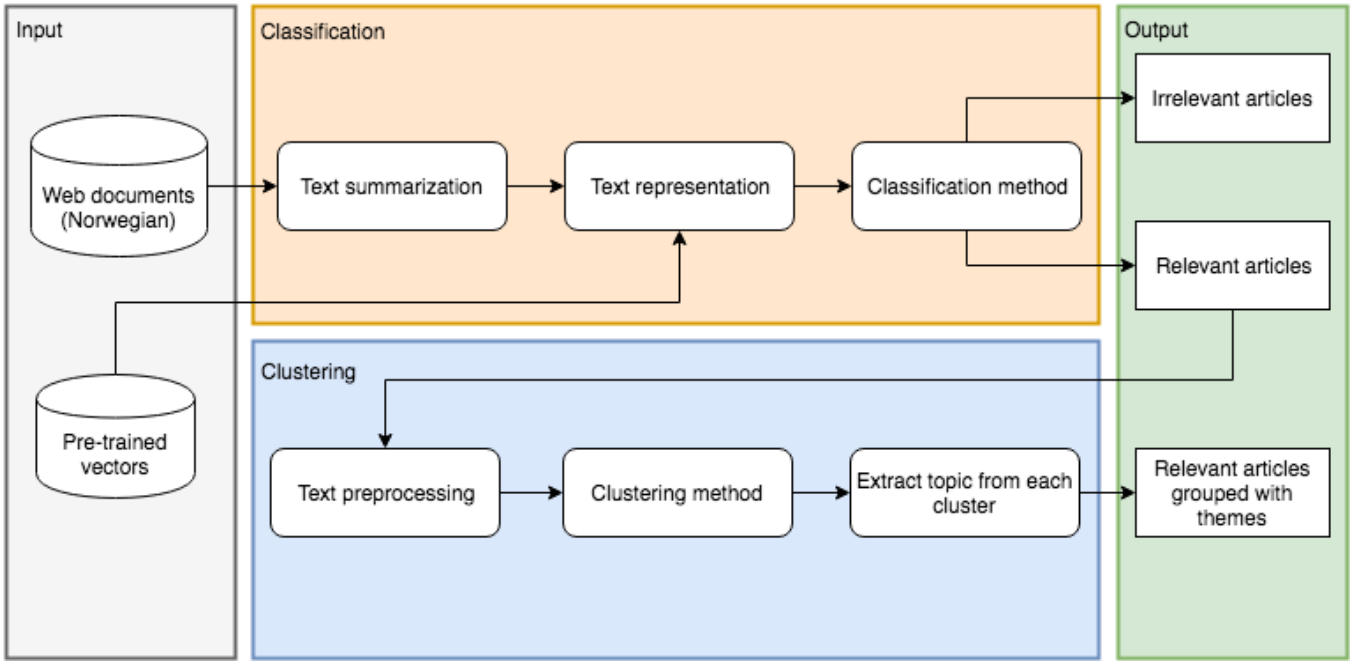


Figure 4.3: Final system overview

4.6.2 Detecting Relevant Web Documents

To perform classification of the data, it was necessary to preprossess and present the data, so the input was suitable for the model. In order to get a fixed size of 50 words or less as input, the python library *sumy* implementing LexRank were chosen to summarize the articles in the dataset containing more than 50 words. If the article consisted of less than 50 words, the input was padded with the value 0.0. The next step was to represent the input as vectors, each with a dimension of 300. *Word2Vec*, *GloVe* and *FastText*, which provided pre-trained vectors for the Norwegian language, was the neural language models used to convert the input to vectors. The next step was to use these inputs in the classifier. The classifier was a deep learning model, built by the Keras API for neural network modelling, and was a combined CNN-LSTM model consisting of 4 layers. The model was explained step by step in Section 4.4.3. The language models and the different neural network models was combined to compare the results. The model classified the input as relevant or irrelevant for the Norwegian Labour Inspection Authority.

4.6.3 Clustering the Relevant Data

The final step in the system was the clustering. The stopwords were removed, then tagged by a library called *RippeTagger* and the input was stemmed by a library from NLTK called *NorwegianStemmer*. The input was then vectorized by using TF-IDF values. The number of clusters was decided by performing the elbow method. The K-Means algorithm, provided by the *Sklearn* library, then received the preprocessed input data and grouped the data into

different clusters and assigned each cluster with topics.

Chapter 5

Results and Discussion

This chapter presents the results and discussion of the research we have conducted during this thesis. The outcome from the classifier (the task of detecting relevant documents) is presented along with the result of clustering of the relevant data. The results from the classifier and clustering method are shown in Section 5.1, and the discussion is encompassed in Section 5.2.

5.1 Results

The results from the classifier, encompassing the performance measure with the four neural network models and the three text representation methods, will be presented in Section 5.1.1, along with the development of the training of the model. The clustering results with the K-means algorithm, the plotting of the clusters and the extraction of the topics from the clusters, are shown in Section 5.1.2.

5.1.1 Classification

The main goal of this thesis was to research if it was possible to classify text as relevant or irrelevant for the Norwegian Labour Inspection Authority, and the classifier was therefore a big part of the testing phase. With the deep learning models described in Section 3.3, the testing and comparison of the different models were performed. The model with the plain Dense, Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) layers was compared with the combined LSTM-CNN model. The combined model with the CNN and LSTM layers and the set hyperparameters was described in the previous chapter, and it was desirable to see if this could outperform the models with plain layers. In addition, we wanted to see which text representation model achieved the best classification results of Word2Vec, GloVe and FastText. To be able to discover the best text representation method and to detect the best model, a comprehensive systematic testing was performed to crosscheck which combinations achieved the best results. Therefore, the three text represen-

tation methods were used as input separately for the four different neural network models we wanted to examine. This way we could research if there were any correlation between which text representation method gave the highest score for the different models. The results are shown in Table 5.1 and will be further discussed in Section 5.2.

Table 5.1: Performance measures for the neural network classifiers

Model	Text Rep.	Class	Precision	Recall	F1-Score	Total F1-Score
Dense	Word2Vec	Relevant	0.7407	0.4444	0.5556	0.8325
		Irrelevant	0.8634	0.9576	0.9080	
	GloVe	Relevant	0.6071	0.3778	0.4658	0.7972
		Irrelevant	0.8462	0.9333	0.8876	
	FastText	Relevant	0.6744	0.6444	0.6591	0.8559
		Irrelevant	0.9042	0.9152	0.9096	
LSTM	Word2Vec	Relevant	0.7368	0.6222	0.6747	0.8674
		Irrelevant	0.9012	0.9394	0.9199	
	GloVe	Relevant	0.6053	0.5111	0.5542	0.8182
		Irrelevant	0.8721	0.9091	0.8902	
	FastText	Relevant	0.7436	0.6444	0.6905	**0.8729**
		Irrelevant	0.9064	0.9394	0.9226	
CNN	Word2Vec	Relevant	0.5000	0.3333	0.4000	0.6833
		Irrelevant	0.8333	0.9091	0.8696	
	GloVe	Relevant	0.6364	0.4667	0.5385	0.8184
		Irrelevant	0.8644	0.9273	0.8947	
	FastText	Relevant	0.7407	0.4444	0.5556	0.8325
		Irrelevant	0.8634	0.9576	0.9080	
CNN-LSTM	Word2Vec	Relevant	0.6889	0.6889	0.6889	0.8667
		Irrelevant	0.9152	0.9152	0.9152	
	GloVe	Relevant	0.7097	0.4889	0.5789	0.8367
		Irrelevant	0.8715	0.9455	0.9070	
	FastText	Relevant	0.6957	0.7111	**0.7033**	0.8719
		Irrelevant	0.9207	0.9152	**0.9179**	

The Accuracy and F1-Scores for the training and test data were plotted to track the development of the training period, both to see the differences for each epoch and to observe the behavior of the model. This way we could possibly detect overfitting by examining if the F1-Score for the training set is considerable better than the test set scores. By plotting the score for each epoch, we saw when the model peaked and could research the number of epochs necessary to create a good model. The plotting of the accuracy and F1-Score was to compare the different performance measures and see how they differ from each other. It was also done to see how the model classifies data to the correct labels versus which one are classified wrong (Accuracy only considers correct labeled data, while F1-Score encom-

passes false positives and false negatives as well, as described in Section 3.6). However, the F1-Score was the performance measure most interesting to track and the final results were based on this metric. The performance measures for each epoch for the CNN-LSTM model for one run is shown in Figure 5.1, the training will be discussed in Section 5.2.

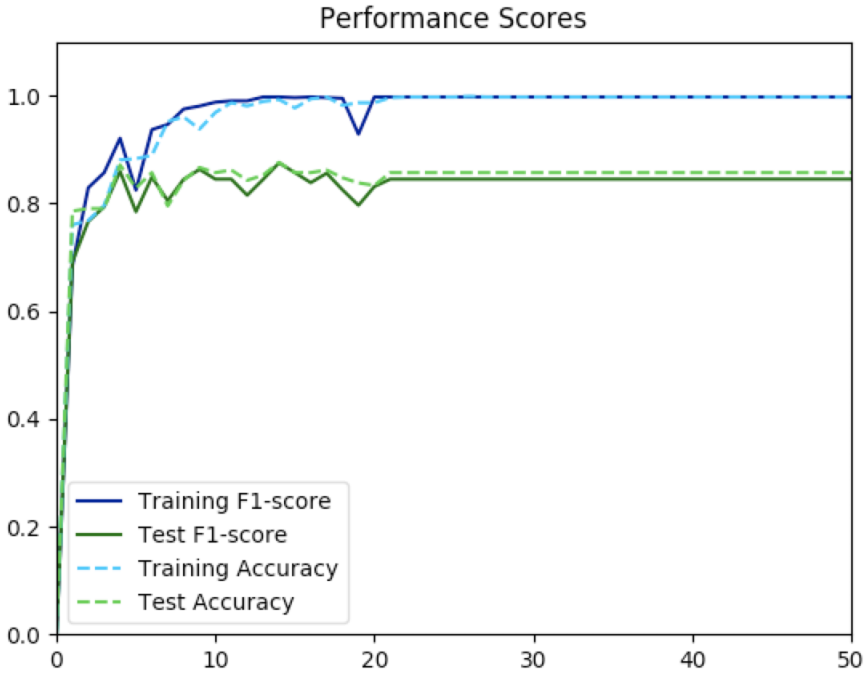


Figure 5.1: Performance measures of CNN-LSTM model

5.1.2 Clustering

For clustering of the relevant data, the partition based clustering algorithm K-means was applied. The number of clusters was decided by using the elbow method resulting in six cluster, as described in Section 4.4.4. The results from the K-means clustering method was plotted and shown in both two-dimensionality and three-dimensionality. The vectors was originally high-dimensional TF-IDF vectors, but as explained in Section 4.4.4 the dimensions was reduced to make it visualizable for the human eye. The two-dimensional plotting of the clusters can be shown in Figure 5.2 and the three-dimensional plotting can be shown in Figure 5.3.

To increase the value of the clusters it was desirable to extract topics from each cluster, making it easier for the Norwegian Labour Inspection Authority to see the superior theme for the relevant data. To select the terms that should be used as topic for the different clusters, two information retrieval approaches were tested. One with the TF-IDF which the clustering is based on, and the other one based on document frequencies. After performing the K-means algorithm with six clusters on the relevant data, the topic extraction was performed.

The results of the topics extracted by TF-IDF and Document Frequency for each cluster are shown in Table 5.2, and will be discussed in Section 5.2.

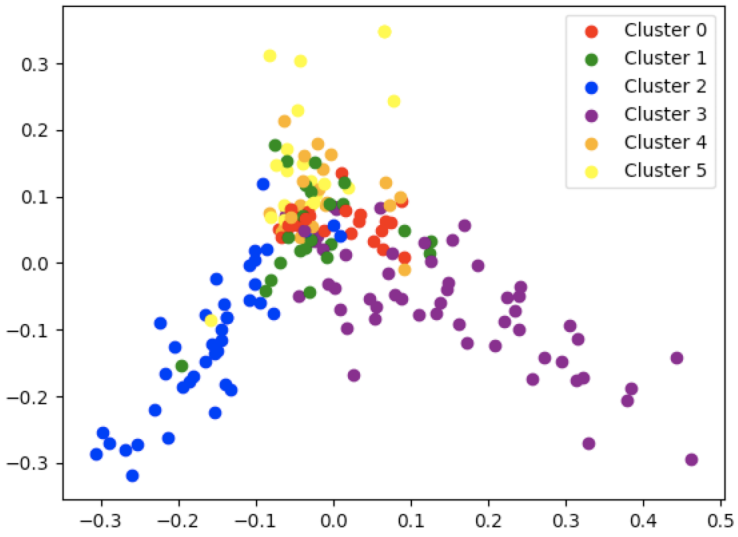


Figure 5.2: Clustering 2D

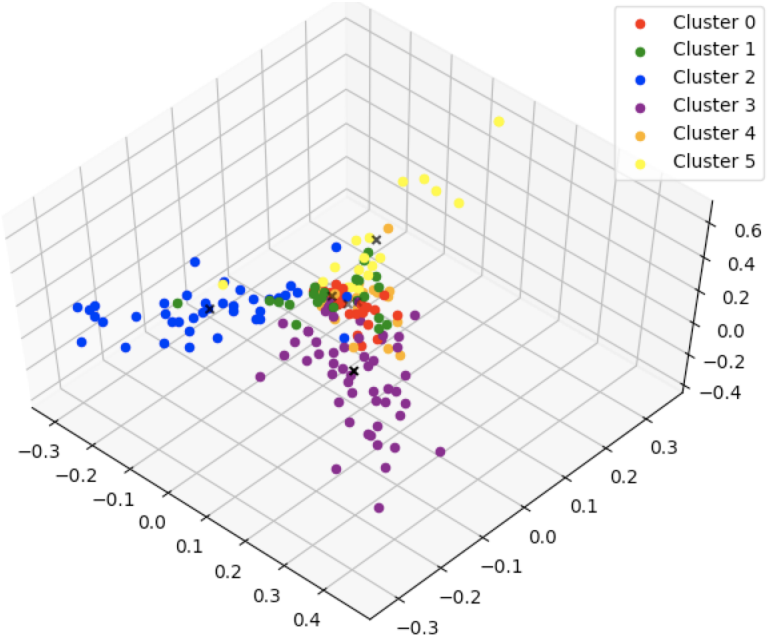


Figure 5.3: Clustering 3D

Table 5.2: Results for topic extraction with TF-IDF and Document Frequency

Cluster	TF-IDF	Document Frequency
0	jobb, inntekt, selskap	led, dag, jobb
1	arbeidsmiljø, bot, kron	brudd, arbeidsmiljø, kron
2	mann, arbeidsulykk, ulykk	ulykk, arbeidsulykk, mann
3	politi, sak, selskap	oslo, politi, sak
4	arbeidstilsyn, tim, tilsyn	arbeidstilsyn, brudd, tim
5	kommun, helsepersonell, sykepleier	sak, arbeidsmiljø, kommun

5.2 Discussion

The evaluation of the results from the classifiers is presented in Section 5.1.1, with a discussion of the neural network models achieving the best results in Section 5.2.1. In this section the behavior of the model during the training period will also be analyzed as well. The evaluation of the clusters and the plotting along with the research of the best topic extraction method, shown in Section 5.1.2, will be discussed in Section 5.2.2.

5.2.1 Classification

Section 4.4.3 presented four neural networks worth investigating; a plain Dense neural network, LSTM, CNN, and a joint LSTM-CNN model. In addition to finding the optimal deep learning model we investigated three neural language text representation methods, namely Word2Vec, GloVe and FastText. It was desirable to decide which combination of text representation method and deep learning model that achieved the best results. The development of the training of the model was worth investigating as well. This will be discussed in the following sections.

Best Text Representation Method

Table 5.1 includes the different text representation methods for the different deep learning architectures. FastText is the text representation which gave the best overall performance measures when used by the different models. One of the reasons FastText is better may be due to the larger vocabulary, which was significantly bigger than GloVe and Word2Vec, at over 500.000 words versus approximately 50.000 (Word2Vec) and 30.000 (GloVe). The size of Word2Vec is 1/10 of the size of FastText, and performs second best on the Dense model, LSTM, and CNN-LSTM. Word2Vec has a F1-Score of 0.4000 for the Relevant class for CNN, which was the worst F1-Score allover. GloVe is the text representation method performing the worst with Dense, LSTM and CNN-LSTM models, but performing better than Word2Vec on the CNN model. The reason GloVe performs worse is probably due to the language model for GloVe is trained on the training set we gathered for this thesis. The

dataset is somewhat limited, as the model created from this set had the smallest number of words, increasing the probability of out-of-vocabulary words.

Of the three text representation methods, FastText did not achieve a F1-Score under 0.55 for the Relevant-class for the four models. Word2Vec gets somewhat good scores for three out of the four models but get a critical deviation of the CNN model achieving only 0.40 as F1-Score for the Relevant-class. GloVe keeps to an average mediocre score, in a range from 0.46 to approximately 0.57 (Relevant-class), but although it has the weakest scores it does not deviate as much as Word2Vec with a range from 0.40 to 0.68 (Relevant-class). For the Irrelevant-class all the text representation methods scored well, the poorest score is 0.8696 for Word2Vec on the CNN model. On the other side, Word2Vec was also the method that achieved the best results on the Irrelevant-class with 0.9199 on the LSTM model. Based on the overall results as presented in Table 5.1 and the highest total F1-Score of 0.8729, FastText was concluded to be the best text representation method for the neural network classifiers.

Best Neural Network Model

In Table 5.1, there are some differences in the F1-Scores for the two classes *Irrelevant* and *Relevant*. There can be different reasons for this, and one reason may be the imbalanced dataset where 20% of the data is labeled Relevant while the rest is labeled as Irrelevant. This may have affected the model during the training, since a major part of the dataset is labeled irrelevant which gives this part a better basis of training than for the relevant part of the dataset. Another reason for the difference in the F1-Score may also be because the amount of data is not sufficient. With approximately 1.000 lines of data, the model may not get the necessary amount of data for training to be able to label the test set more correctly. The scores for the Relevant-class is not perfect (or close to perfect), and one reason can be because of the data annotation. The test set was annotated by the Norwegian Labour Inspection Authority, while the training set was labeled by us, creating some differences in the evaluation of the data as relevant or irrelevant. The amount of relevant information available were very limited, and the irrelevant information dominated the dataset. Because of this we chose to select the model with focus on the highest score for the Relevant class, together with good results on the Irrelevant class.

It emerges from Table 5.1 that the model with the joint CNN-LSTM architecture achieved the best results when using FastText as the text representation method. This model achieved a F-Score of *0.9179* on the Irrelevant class, *0.7033* on the Relevant class and *0.8719* in total F1-Score. This is the model achieving the best results on the relevant documents, but there is another model achieving a little better on the irrelevant documents and the total F1-Score. The LSTM model with FastText provided the best irrelevant results and total F1-Score, with an F1-Score of *0.9226* on the Irrelevant-class and *0.8729* as total F1-Score. The LSTM model with Word2Vec also achieved an F1-Score of *0.9199* on the irrelevant class. This shows that the LSTM model is highly competitive, and only performs poorer on the relevant class but still achieving the highest total F1-Score. For us the performance score for the relevant class is the most important metric, which is also one of the reasons why we concluded that the CNN-LSTM was the best model. The LSTM model also performs

slower than the CNN-LSTM model, most likely because of the pooling layer the joint model has. This layer helps the CNN-LSTM model perform slightly faster because it reduces the dimensionality of the data, which is an performance advantage. The model that performs poorest is CNN with Word2Vec, with an F1-Score on the relevant class of *0.400*. The overall CNN results are worse than the rest of the models. This may be because CNN focuses on data which is ”close” and this works well in images, but the semantically close words in a text may be several words apart (as described in Section 3.3.3).

The different results on the two classes is, as explained earlier, most likely a reflection of the imbalanced and somewhat small dataset. It is therefore interesting to look further into both LSTM and CNN-LSTM models, since they achieve very similar results. On the dataset we have available, we chose the CNN-LSTM model as the best one because of the results on the relevant class and the performance.

Training the Model

Figure 5.1 shows the development of the model during training. The performance metrics (accuracy and F1-Score) are plotted both for the training set and test for each epoch. Based on the Figure 5.1 it is seen that the testing of the model gets a F1-Score of 0.70 already in the first epoch. The scores increase quickly, achieving the best F1-Score for the test set in epoch 14 approximately close to 0.88. The training set scores converge and reach a perfect score (1.0) after epoch 22 or 23 and no changes happen after this, the performance metrics for the test stops increasing as well at the same epoch. There is a significant gap between the training set and test set scores. Some overfitting occurred, but this was the best F1-Score for the test set we achieved. As explained in Section 4.4.3 we examined the use of regularizers but that resulted in worse scores of the test set although it reduced the overfitting slightly. We applied dropout layers in the model to reduce the overfitting which led to better test results and decreased the gap to some extent.

In the experimental phase of developing the model, we observed that the neural network achieved good results quickly (before 10 epochs have been run). This may be because major parts of the dataset (80%) is composed of data labeled as Irrelevant, and when initializing the model, it was ”easy” to get a good score immediately. The results of the test set do not increase in parallel with the training set scores but is able to reach a peak getting the best performance metrics in a later epoch (typically around 20 to 40 epochs). This may be because during training of the model the computation of the weights experiments somewhat with the adjustments, allowing it to discover possible better scores. Another reason the test set does not evolve at the same time as the training set, may be because the annotation of the data is different. The Norwegian Labour Inspection Authority may have labeled some web documents to one class in the test set, while we have labeled the same web documents to another class, creating ambiguity in the dataset. One possible way to solve this problem is conducting the annotation of the training set together with the Norwegian Labour Inspection Authority, creating a consensus of the labeling of the web documents.

5.2.2 Clustering

To evaluate the clustering of the relevant documents there were two aspects interesting to investigate; the visualization of the clustering, and the extraction of topics for each cluster. The performance of the K-means clustering and which topic detecting method used will be described in the following sections.

Evaluating the Clusters

We wanted to see which articles that were clustered and why they were grouped together. Figure 5.2 and 5.3 shows the plotting of the data based on our TF-IDF vectors, both two-dimensional and three-dimensional respectively. There were no clearly divided clusters in Figure 5.2 or Figure 5.3, and to verify the correct number of clusters was complicated. The reason for no clearly separated clusters is because to cluster natural language is a challenging task. It may also be because of our limited dataset, and expanding the dataset may result in more distinct clusters. In addition, the visualization of the cluster is not necessary completely correct because of the original high dimensional TF-IDF vectors was reduced to two and three dimensions, and reduction can have lost important dimensions of the vectors. It may be that the vectors in ten-dimensional (or higher dimensions) may result in more distinctive visual clusters but is harder to visualize for the human eye.

In Figure 5.2 and Figure 5.3 there were three clusters which were somewhat easy to spot, namely cluster 2, 3 and (barely) 5. Combining this with the topics given in Table 5.2 it can be seen that cluster 2 is about working accidents which is highly represented in the dataset. Cluster 3 contains the terms ”politi”, ”selskap” and ”sak” (*eng:* police, case, and company) which is not obvious terms for the topics for the cluster but gives you a hint that the police is involved with some kind of police reports. Cluster number five contains ”kommun” (*eng:* municipality) which is somewhat descriptive since these documents contains information about several municipalities in Norway and their routines about warnings of fraud among other things. Clusters 0, 1, and 4 are harder clusters to verify out of Figure 5.2 and Figure 5.3. This is the reason for the three-dimensional visualization to be able to see the clustering from another perspective. However, some of the topics from these three clusters were somewhat descriptive, for instance for cluster 1 is ”arbeidsmiljø” and ”bot” (*eng:* work environment and fine), giving an indication of enterprises may have gotten a fine from poor working environments or other similar cases. Cluster 0 have the terms ”jobb”, ”inntekt”, ”selskap” (*eng:* job, income, enterprise), and since the clustering has been performed on the relevant documents for the Norwegian Labour Inspection Authority the chances are that these articles may be about suspicious enterprises and their income. This is for the Norwegian Labour Inspection Authority to judge.

A limitation with the use of a static number of clusters is if the dataset is expanded. As mentioned earlier the articles domination in the media is changing, considering for instance the #metoo campaign. One possible solution is to perform the K-means algorithm again after new data have been gathered to see if creating new clusters is necessary.

The clustering using TF-IDF vectors was not perfect, but it managed to create reasonable

clusters with fairly relevant terms. Using the K-means algorithm works well and by deciding the number of clusters with the elbow method as described in Section 4.4.4 it provided sufficient results.

Selecting the Best Topic Method

To evaluate the topic of the clusters we performed the two methods based on TF-IDF and Document Frequency on each cluster. Table 5.2 shows the main terms in each of the clusters for two approaches, one with the use of TF-IDF and the other one using Document Frequency. TF-IDF and Document Frequency are both traditional information retrieval methods and works well in this scenario since we were dealing with Norwegian text and these methods are language independent. Previously, the terms retrieved for the clusters have been discussed overall for each cluster, and now we want to compare the two methods and see which one is best. Some of the terms were not very descriptive, e.g. ”lead”, ”day”, or ”job” (translated from Norwegian) and was not much of value as topic for a cluster (Document Frequency on cluster 0), see Figure 5.5 for a word cloud of the text. On the other side, for cluster 2 TF-IDF and Document Frequency mentions the word ”arbeidsulykke” (*eng:* work accident), which is a very descriptive word giving the Norwegian Labour Inspection Authority an overview of occurring accidents at work places worth investigating. This is also one of the relevant words mentioned in the list in Section 4.2.

The two approaches for detecting topics for a cluster was very similar, they retrieved many equal terms which was descriptive of the content, but some terms were not sufficient. Some differences occurred where one of the approaches did not retrieve important terms which the other approach managed to detect. In cluster 5 TF-IDF retrieves the word ”helsepersonell” (*eng:* health-care provider) which indicates that this cluster may be about the health sector. However, in this cluster the Document Frequency method returned the term ”arbeidsmiljø” (*eng:* work environment) which indicates that there may be something about the Work Environment Act. Both methods retrieve important words but the TF-IDF method gives better unique results referring to the health sectors than the overall working environment. Furthermore, in cluster 0 the TF-IDF method return the words ”jobb”, ”inntekt” and ”selskap” (*eng:* work, income and enterprise), where the document frequency retrieved ”led”, ”dag” and ”jobb” (*eng:* lead, day and work) The terms ”dag” and ”led” are not descriptive enough, so the Document Frequency retrieved poorly topics in this scenario. The terms from the TF-IDF approach gives slightly better results retrieving the words ”income” and ”enterprise” narrowing the theme of the cluster to some extent.

It was interesting to see which words occurred the most, and as a verification of the topic choice methods, a *word cloud* was generated by a Python library[1]. An example of the word cloud of cluster 2 can be shown in Figure 5.4, and of cluster 0 can be shown in Figure 5.5. Figure 5.4 gives a visualization of the terms occurring frequent in the documents for cluster 2. The word cloud highlights the terms such as ”arbeidsulykke” and ”ulykke” (*eng:* work accident and accident) which is interesting terms for the Norwegian Labour Inspection

[1] `https://github.com/amueller/word_cloud`

Figure 5.4: Word cloud for cluster 2

Figure 5.5: Word cloud for cluster 0

Authority. This verifies that both the TF-IDF and Document Frequency approaches works well on this cluster. However, the Figure 5.5 for cluster 0 shows that the most frequent words are more abstract and will not give the Norwegian Labour Inspection Authority any indication of what has happened. However, the TF-IDF method was able to get the most unique terms for cluster 0.

Based on these results we conclude that the TF-IDF approach was the most informative and gave better results. When evaluating the clusters, six clusters were a good division of the texts. Even though some of the articles grouped together was noisy and could be considered as outliers the most relevant documents were grouped together (e.g. considering work accidents in cluster 2, and health sector in cluster 5).

Chapter 6

Conclusion and Future Work

This chapter will present the conclusion of the research of detecting relevant web documents for the Norwegian Labour Inspection Authority, as well as future work to improve and extend our approach. In Section 6.1 we present the conclusion for the work in this thesis, where we explain the contributions and goal achievements related to the research questions. In Section 6.2 we present a proposal for future work.

6.1 Conclusion

This section encompasses the contributions we have obtained during this research; the dataset retrieved, and the machine learning classification used for detecting relevant documents. This will be presented in Section 6.1.1, but first our goal achievements answering the research questions from Section 1.3 will be presented. How the questions have been answered in this thesis is explained and elaborated below:

RQ 1: *Where can we extract the most usable data from open sources to create a valuable dataset?*
The research presented in Section 4.3 shows that the social media platforms Twitter and Facebook did not provide the necessary access or enough relevant data. The discussion in Section 4.3.1 led us to create a dataset consisting of valuable information from a broad specter of different online Norwegian news papers. The sources is presented in Table 4.2.

RQ 2: *How can the data be processed and represented for further processing?*
In this thesis it was found beneficial to summarize the text as part of the preprossessing of the data. The main argument was to be able to create a standard size of the input to the classifier, since articles vary a great deal in size and number of words. This is explained in Section 4.4.1, together with the research that resulted in choosing LexRank as the summarization algorithm. The experiments presented in Section 4.4.2 explained several possible text representation methods, and the method which achieved the best results with the classifier was

FastText, as described in Section 5.2.1.

RQ 3: *How can we develop a suitable machine learning method that is able to find relevant data?*

The classification task was an important part of this thesis, and several machine learning methods have been presented in Section 3.3. In Section 3.3.4 these methods was discussed, and neural network was chosen as the machine learning approach. More specific a joint Convolutional Neural Network and Long Short-Term Memory (CNN-LSTM) model was proven to be the best model for the task of detecting relevant data and achieved good result.

RQ 4: *How can we present the relevant data making it more applicable for the Norwegian Labour Inspection Authority?*

Clustering the relevant data using the partition-based clustering algorithm K-means sorted the data into six groups. Extracting the topics for each group made it easier to get an overview of the different themes in the news and made the results more applicable for the Norwegian Labour Inspection Authority. Getting a superior overview rather than just present the results from the classifier as one collection of data, was more useful and easier to comprehend.

These four sub questions have all contributed to answering the main research question, which was:

RQ: *How can relevant information for the Norwegian Labour Inspection Authority be detected in open data sources?*

The research in this thesis show that, among the sources considered interesting by the Norwegian Labour Inspection Authority as presented in Section 4.3, online news articles was the web documents containing most available information. By summarizing and representing the text, the data was applicable for input to a classifier. A neural net classification approach using a combined CNN-LSTM model was one method to detect relevant information in open data sources and achieved sufficient results. Combined with the K-Means clustering algorithm, the relevant information from open data sources was presented in a usable way for further investigation and processing.

6.1.1 Contributions

The main contributions of this thesis are directly related to the classification of online new articles for the Norwegian Labour Inspection Authority. The main contributions are mentioned briefly in Section 1.3, and can be divided further into the following three main topics; exploration of data sources, data gathering and machine learning classification. The respective contributions will be summarized and explained in the following sections.

Exploration of Data Sources

This section is an elaboration of the first contribution in Section 1.3; *Exploration of new sources of data for the Norwegian Labour Inspection Authority*. Since there is no previous work directly related to this thesis, a thorough exploration of the different data sources was necessary. The results of the experimentation with different sources for information showed that online news papers were the most accessible, and contained relevant information, since it report recent news every day. In Table 4.2, an overview of the different news papers applicable for text retrieval is presented.

Norwegian Dataset

The second contribution presented in Section 1.3 is; *A Norwegian dataset consisting of news articles annotated as relevant or irrelevant for the Norwegian Labour Inspection Authority*. After the exploration of different data sources, the information was retrieved and extracted into a dataset. This dataset was annotated by us and the Norwegian Labour Inspection Authority. Creating a dataset was necessary to train the classification model.

Machine Learning Classification

The third contribution is; *A deep learning architecture for text classification of news articles*. We researched different neural network models to classify the dataset of Norwegian text. By summarizing the texts and using FastText as text representation we proposed a joint CNN-LSTM architecture to classify data as relevant or irrelevant. The model achieved good results, with the performance scores *0.7033* for the Relevant class and *0.9179* for the Irrelevant class. Since there were no directly competitive earlier work, we claim to achieve state-of-the-art results in this domain for the Norwegian Labour Inspection Authority on Norwegian texts.

6.1.2 Main Findings

The main findings in this thesis are mostly related to the Norwegian Labour Inspection Authority and their future work. We conclude that our main findings are about the sources of data. When we explored social media, we did not discover much relevant information. Even though social media is an interesting platform to extract information from, we have contributed to a deeper understanding of what is required from such platforms to be used as sources, such as access and an expansion of the search conditions. During the work with this thesis, we found that there were little support for the Norwegian language in e.g. the already existing tools we explored.

6.2 Future Work

This thesis proposed a decision support system for the Norwegian Labour Inspection Authority, and some improvements and extensions we suggest that can be applied to achieve better results will be presented in the following sections.

Gather More Data

Our dataset is somewhat limited when it comes to the number of articles gathered. The total dataset contains 1048 articles with approximately 20% marked as relevant. The training set consists of 80% of the data gathered, while the test set is 20%. This leads to a small set used for training, and with more data the model can be fitted to classify the test set more correctly. In addition, making the dataset more balanced can also reduce the differences of performance scores for the Relevant and Irrelevant classes.

Support for the Norwegian Language

To the best of our knowledge, one of the existing restrictions is the support for Norwegian language. Named Entity Recognition (NER) and sentiment analysis was relevant features for this system, but due to the lack of support for the Norwegian language in the tools we explored, the scope of this thesis stopped the further investigation. It can be interesting for further work to investigate how to use them as additional features in the classifier as well.

Clustering Based on Neural Language Word Vectors

The clustering method we applied was based on TF-IDF vectors, but it is interesting to see if it is possible to cluster based on list of word vectors with 300 dimensions from FastText. By using neural language models as FastText may give the opportunity to cluster based on semantically close words, given more significant clusters. We did not manage to cluster the data using 50x300 vectors for each text in the dataset due to limited existing methods in Python for doing this.

Dynamic Number of Clusters

After the articles were classified and they were clustered into groups, we assigned a topic which is named in each cluster. These topics were based on the frequent news at a given time, and this approach is not dynamic when it comes to changes and new concepts in the news. For instance, the flood of #metoo articles had a rise in 2017, and there are a chance that similar trends may occur in later news.

Another approach to using clustering and extracting the topics, is to use a database of relevant words from the Norwegian Labour Inspection Authority to extract domain specific topics.

Social Media

In this thesis different sources for information have been explored, and social media were eventually not included as a source. This platform is still interesting for the Norwegian Labour Inspection Authority as it contains user-generated text, and with the right access the amount of data can be enough, given sufficient quantity of relevant data. For future work it can also be interesting to investigate different languages, because foreign employees may also produce text about working conditions and similar cases on social media. Text from these employees together with geo-location on e.g. tweets can contribute to even more valuable information for the Norwegian Labour Inspection Authority. E.g. a polish tweet with geo-location in Norway can be data containing relevant information for the Norwegian Labour Inspection Authority.

Bibliography

Agarwal, B., Ramampiaro, H., Langseth, H., and Ruocco, M. (2017). A deep network model for paraphrase detection in short text messages. *arXiv preprint arXiv:1712.02820*.

Aggarwal, C. C. and Zhai, C. (2012a). *A Survey of Text Classification Algorithms*, pages 163–222. Springer US, Boston, MA.

Aggarwal, C. C. and Zhai, C. (2012b). *A Survey of Text Clustering Algorithms*, pages 77–128. Springer US, Boston, MA.

Al-Rfou, R., Kulkarni, V., Perozzi, B., and Skiena, S. (2015). Polyglot-NER: Massive multilingual named entity recognition. *Proceedings of the 2015 SIAM International Conference on Data Mining, Vancouver, British Columbia, Canada, April 30 - May 2, 2015*.

Balk, E. M., Chung, M., Hadar, N., Patel, K., Winifred, W. Y., Trikalinos, T. A., and Chang, L. K. W. (2012). Accuracy of data extraction of non-english language trials with google translate. .

Balog, K. and Ramampiaro, H. (2013). Cumulative citation recommendation: Classification vs. ranking. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 941–944. ACM.

Balog, K., Ramampiaro, H., Takhirov, N., and Nørvåg, K. (2013). Multi-step classification approaches to cumulative citation recommendation. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, pages 121–128. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

Becker, S. and Plumbley, M. (1996). Unsupervised neural network learning procedures for feature extraction and classification. *Applied Intelligence*, 6(3):185–203.

Bird, S., Loper, E., and Klein, E. (2009). *Natural Language Processing with Python*. O'Reilly Media Inc.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Britz, D. (2015). Understanding convolutional neural networks for nlp. *URL: http://www. wildml. com/2015/11/understanding-convolutional-neuralnetworks-for-nlp/(visited on 11/07/2015)*.

Chowdhury, G. G. (2003). Natural language processing. *Annual review of information science and technology*, 37(1):51–89.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Cong, F., Leung, A., and Wei, Q. (2017). *Advances in Neural Networks-ISNN 2017: 14th International Symposium, ISNN 2017, Sapporo, Hakodate, and Muroran, Hokkaido, Japan, June 21–26, 2017, Proceedings*, volume 10262. Springer.

Erkan, G. and Radev, D. R. (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Georgios, P. K., Ramampiaro, H., and Langseth, H. (2018). Detecting offensive language in tweets using deep learning. *Department of Computer Science Norwegian University of Science and Technology*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Graves, A. (2012). Long short-term memory. In *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 37–45. Springer.

Groves, M. and Mundt, K. (2015). Friend or foe? google translate in language for academic purposes. *English for Specific Purposes*, 37:112–121.

Güngör, T. (2010). Part-of-speech tagging.

Hassan, A. and Mahmood, A. (2018). Convolutional recurrent deep learning model for sentence classification. *IEEE Access*, 6:13949–13957.

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

Karn, U. (2016). An intuitive explanation of convolutional neural networks. *ujjwalkarn, August*.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kodinariya, T. M. and Makwana, P. R. (2013). Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95.

Ku, C.-H. and Leroy, G. (2014). A decision support system: Automated crime report analysis and classification for e-government. *Government Information Quarterly*, 31(4):534–544.

Kvamme, H. (2015). *Gender prediction on Norwegian Twitter accounts*. Norwegian University of Science and Technology.

Liaw, A., Wiener, M., et al. (2002). Classification and regression by randomforest. *R news*, 2(3):18–22.

Liu, B. (2007). *Web data mining: exploring hyperlinks, contents, and usage data*. Springer Science & Business Media.

Liu, B. and Zhang, L. (2012). *A Survey of Opinion Mining and Sentiment Analysis*, pages 415–463. Springer US, Boston, MA.

Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. Morgan and Claypool Publishers.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

McMinn, A. J., Moshfeghi, Y., and Jose, J. M. (2013). Building a large-scale corpus for evaluating event detection on twitter. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 409–418. ACM.

Mihalcea, R. (2004). Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 20. Association for Computational Linguistics.

Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Nenkova, A. and McKeown, K. (2012). A survey of text summarization techniques. In *Mining text data*, pages 43–76. Springer.

Nielsen, M. A. (2015). Neural networks and deep learning. *http://neuralnetworksanddeeplearning.com/chap1.html*.

Norouzi, M., Fleet, D. J., and Salakhutdinov, R. R. (2012). Hamming distance metric learning. In *Advances in neural information processing systems*, pages 1061–1069.

Øye, J. A. (2015). *Sentiment Analysis of Norwegian Twitter Messages*. Norwegian University of Science and Technology.

Patel, P. and Mistry, K. (2015). A review: Text classification on social media data. *IOSR Journal of Computer Engineering*, 17(1):80–84.

Paulsen, J. R. and Ramampiaro, H. (2009). Combining latent semantic indexing and clustering to retrieve and cluster biomedical information: A 2-step approach. *Norsk informatikkonferanse (NIK)*.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Piskorski, J. and Yangarber, R. (2013). *Information Extraction: Past, Present and Future*, pages 23–49. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ramos, J. et al. (2003). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142.

Repp, Ø. K. (2016). *Event Detection in Social Media*. Norwegian University of Science and Technology.

Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., and Edwards, D. D. (2010). *Artificial intelligence: a modern approach*. Prentice hall Upper Saddle River, third edition.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Skiena, S. S. (2017). *The Data Science Design Manual*. Springer.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Tang, B., He, H., Baggenstoss, P. M., and Kay, S. (2016). A bayesian classification approach using class-specific features for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1602–1606.

Ting, K. M. (2010). *Precision and Recall*, pages 781–781. Springer US, Boston, MA.

Tripathy, R. M., Sharma, S., Joshi, S., Mehta, S., and Bagchi, A. (2014). Theme based clustering of tweets. In *Proceedings of the 1st IKDD Conference on Data Sciences*, pages 1–5. ACM.

Warner, W. and Hirschberg, J. (2012). Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yang, J. and Leskovec, J. (2011). Patterns of temporal variation in online media. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 177–186. ACM.