Investigating political speeches using machine learning

Master thesis

September 17th 2018

Sara Lee Naldal

Student No.: 88877
Cand.merc.it
Supervisor: Daniel Hardt
Number of characters: 145.004
Number of pages: 64

CBS COPENHAGEN BUSINESS SCHOOL
HANDELSHØJSKOLEN

Abstract

This thesis investigates the political affiliation of statements given by members of the Danish Parliament. It employs methods from computational linguistics which combines computer science methods of machine learning with linguistic knowledge to perform natural language processing. I specifically work within the framework of sentiment analysis and their concept of document sentiment analysis for the investigation.

The thesis attempts to automatically classify the statements according to their political affiliation using machine learning. The empirical material of the thesis consists of speeches from the Danish Parliament over an 11 year time period. I employ a support vector machine and a neural network for the task and test them on the full dataset as well as on temporal slices of the data depending on its year of origin. The support vector machine represent a typical machine learning method with proven good results in document classification tasks. The neural network was implemented to test a deeper learning method on the data set and compare its performance with a support vector machine.

In my testing I perform binary and multiclass classification and compare the overall performance of the two methods both on the full dataset as well as it partitions and discuss the shortcomings and strengths of the two methods compared to each other. Both methods achieve fairly good results compared to current research on both the binary and the multiclass task, however, with a much better performance on the binary learning task. Following this I investigate the generalizability of both methods by testing them on a dataset consisting of tweets and status updates from Twitter and Facebook. Following this I discuss the results and the level of domain dependence shown by both methods.

Evaluating the project as a whole there are definite options for future development including a deeper level of preprocessing the data prior to training and taking a more explorative approach to investigate the language use of the different parties both as a whole and temporally.

Finally I discuss possible broader use cases for automatic detection of political affiliation focusing especially on online texts including microtargeting and bias detection.

Index

Acknowledgements

Thank you Mikkel Nielsen who helped with the manual parts of an otherwise very time consuming data collection process.

Introduction

New technology is changing the world and the way humans interact with it. From improved ERP systems (Carlton, 2014) and chatbots taking over customer support roles (Flaiz, 2018) to IoT technologies such as smart fridges becoming increasingly normal (Lloyd, 2018). As both professional and personal aspects of life become digitized, technologies, which can help decipher, understand and relay information, have blossomed. An example of this is the field of language technology and research. The rapid growth of the web and the spread and increased use of social media have provided language technology research with an immense amount of data. Combined with progress in machine learning and big data analysis this has enabled research within natural language processing to improve computational understanding and production of language, which in turn can be used to improve user interaction and general usability of technology (Borin, 2018). On a larger scale the increased focus and possibilities within language technology is also reflected in large corporations such as Microsoft and Facebook, who both have invested in AI and language technology research and development in the last few years (Microsoft, 2018; Chaykowski, 2016).

One of the most active research fields within language technology and natural language processing since the start of the millenium is sentiment analysis (Liu, 2017). Sentiment analysis seek to understand the opinion of a given expression, and is a field of study which have greatly benefited from the rise of social media and other online methods of expression such as reviews and blogs as all of these have created a large online distribution of data, which can be mined and used for further analysis. As a result sentiment analysis is widely studied in for instance data mining and information retrieval as well in a variety of business fields and domains. Some of these fields and domains include customer sentiment analysis on products or brands and even political analysis to predict the success or failure of campaigns or candidates (Liu, 2017). Newer research within sentiment analysis have sought to go beyond a positive/negative analysis of opinions to gain a deeper understanding of for instance political opinion (Pla & Hurtado, 2014). The idea of being able to understand political discourse computationally intrigues me and I therefore wanted to delve further into this topic.

During my studies at Copenhagen Business School and the courses, which have involved big data analysis, I have been working with classifying texts using machine learning. During this I have mainly worked with smaller datasets and attempted to perform a binary classification of texts. For this thesis I want to expand upon that by increasing the dataset size and the number of classes being classified as well as the machine learning methods applied on the dataset. I want to classify Danish opinion documents. There are two main reasons for this.

Firstly, although modern technology is a strong tool, the research in the field is to a large extent mainly focused on tasks involving the English language. As a result so-called low resource languages with fewer native speakers and thus not as much available data fall behind

from a research point of view and do not develop as many tools or at the same rate as is the case for English language technology. In a 2012 analysis of the current state of language technology within the European languages, the vast majority of the languages assessed turned out to have only moderate or fragmentary technological support or coverage with English being the only language with a high score (Rehm & Uszkoreit, 2012).

Rehm & Uszkoreit (2012) suggest that a reason may be, that the methods for analyzing smaller languages belonging to the same language group as English are close to identical to those for English, and research have therefore focused on the latter. Danish is a Scandinavian low resource language, which is related to English. Danish differentiate from English in a few ways such as its word formation and word structure allowances, and I would therefore like to investigate how the methods, which have been employed for English will perform on a Danish data set.

Secondly, a large part of the research investigating political affiliation using machine learning use an American setting for their data. American politics have a two party system, which fits well with sentiment analysis methodology, which is often binary in its approach. This stems from typical sentiment analysis research, which have been focused on polarity research seeking to automatically classify texts as being either positive or negative (Liu, 2017). The Danish political system in comparison to the American system, although it can be considered binary on a general level (Larsen, 2015), consists of several smaller individual parties in a left-right wing dimensionality, who form coalitions depending on individual laws and propositions giving a less polarized environment for the machine learning task.

I will attempt to answer this question by using machine learning methods on a data set consisting of statements from the Danish parliament. I will employ a general machine learning method for the sentiment analysis task based on the literature review. Because I have done a similar task on a smaller data set before I also want to employ a neural network, which is a more complicated learning method, which is also used within sentiment analysis. In the results I will review and compare the performance of both methods. After testing and reviewing the results of the general test, I will apply both models on a secondary test set to investigate and compare their results on unknown texts. This gives me the following research questions.

Is it possible to automatically classify Danish texts according to their political affiliation using machine learning?

How will a deeper learning model perform compared to a typical machine learning method?

The layout of the project is as follows: I will first cover the general theory behind computational linguistics and sentiment analysis to gather an idea as to which terms and concepts that are needed to understand the current research. Based on this I will go through

the literature review, which the implementations and testing are based on. Afterwards follows the classification testing with an initial data description, methodology and discussion of the results. Following the classification I test the models on a new data set consisting of unknown texts in investigate, whether the model could be expanded for general use. .

Lastly I will discuss the broader aspects and possible use cases for automatic detection of political affiliation before summing up the results of the project and evaluate the thesis.

This project is not a political analysis of the Danish multiparty system. I will be using methods and theory from computational linguistics to classify the texts based purely on their intextual features, and I will therefore not be annotating the texts with various background variables such as gender, age, etc. I will be classifying the texts given the current descriptions of the political positioning of the Danish parties. This means that I will not be attempting to position the parties according to a given metric, but rather investigating whether machine learning methods based on language use by the speaker, can correctly sort political speeches into the right target values.

Theory

This section covers the theoretical aspects of the project. Below I will shortly describe computational linguistics and more specifically natural language processing, which makes the foundation of the project. I will then elaborate on key terms and expressions used within computational linguistics and machine learning when used to perform natural language processing.

This project specifically utilises methodology from a branch of natural language processing called sentiment analysis, which will be covered following the key terms and expressions. I will also write about Danish as a language in relation to the covered theoretical aspects of the assignment. The foundation of the section stems from previous courses on linguistics, machine learning and big data analysis and their related text books. The sections covering sentiment analysis is based mainly on Liu (2017), which were discovered during the literature review.

Computational Linguistics

Computational linguistics is an interdisciplinary field of studies, which combines the knowledge and methodology from linguistics and computer science with the aim of investigating language from a computational perspective (Jurafsky & Martin, 2012). The field stems from the 1950s, when researchers worked to perform automatic machine translation, but later developed to focus more generally on providing computational models to investigate general linguistic phenomena as well as harnessing linguistic knowledge to

improve computational understanding and parsing of natural written or spoken language (Jurafsky & Martin, 2012).

This project aims to classify transcribed speeches. I am therefore employing methods from the branch of computational linguistics commonly referred to as natural language processing. Natural language processing research investigates, how computers best understand and parse language and historically have two main approaches to do so. Early natural language processing had a rule based approach, which adhered to a rationalist perspective in its understanding of language. This meant that language was thought to be an innate ability. Early examples of this approach include formal language approaches such as Chomsky's context-free grammars and finite state models to represent language (Jurafsky & Martin, 2012).

This thesis, however, focuses on an empiricist perspective to natural language processing. Empiricism within linguistics considers language as learned and an acquired ability through the environment and culture. The general methodology of empiricists is to look at the actual use of language by for instance comparing large corpora with observed phenomenon. Empiricists within computational linguistics specificically utilise probabilistic models such as naive bayes and regression to perform natural language processing tasks. The empiricist approach only evolved further with the increased available amount of data sources and computing power to perform statistical modelling. This further caused previously ruled based methods like part of speech tagging to incorporate probabilities into their algorithm as well (Jurafsky & Martin, 2012).

The problem with natural data is that it is rarely normally distributed or regular in its distribution (Manning & Schütze, 2003). In a given corpus or data set there is most likely a lot of files or parts of the files that need to be filtered and sorted to remove junk content. Natural language processing consists of two important parts in its process. Firstly, the given data e.g. speech or text need to be converted into a computer accepted format. Once the data has been converted into an acceptable representation, it can be processed using machine learning for deeper learning purposes. The specific type of deep learning depends on both the data, the preprocessing and the chosen machine learning method. Below I discuss general considerations regarding the best representation of language for natural language processing purposes, when dealing with text based data. Afterwards I will describe the overall distinction and directions of machine learning methods and their general purpose.

Representing text in machine learning

Tokenization, lemmatization and stemming

The first part of preparing texts for computational work usually requires splitting the strings of text into tokens. This process is called tokenization. The tokens are usually words and

punctuation, however, a given string of text can just as well be split into syllables, sentences or individual characters (Manning & Schütze, 2003).

When tokenizing texts the model might have the problem, that the same word will appear in different settings conjugated differently and thus for the computer appear as a new word. This problem can be assessed by using lemmatizers or stemmers. Lemmatization and stemming are two similar methods of handling different conjugations of the same word. Lemmatization is the method of returning a given word to its base dictionary form i.e. its lemma (Croft et al, 2009). Similarly stemming wants to transform into their base forms, but a stemmer is a lot more crude in its methodology as it simply removes inflections at the end of the words rather than analyzing and finding the correct lemma (Croft et al, 2009).

Vector Space Models

Past tokenization and possible lemmatization variations of texts, vector space models are often employed to obtain a simple representation of the texts, which is easily comparable to the other instances in a data set. Vector space models stem from information retrieval theory. In this theory texts or data instances are represented as multidimensional vectors, where each word in the full data set of texts is a dimension in the joined vector space. In information theory the idea is here, that similar documents would have similar vectors, and thus would be 'closer' to each other in the vector space (Manning & Schütze, 2009).

A common simple vector space model is the Bag of Words model, which employ s the basic principles of a one hot encoded vector based on the overall dataset library of words. One hot encoding basically means that the vector is binary in its word representation. A one hot encoded vector would have the same length as the index of over overall library of words, and for each word have encoded whether they exist in this space model. This model, however simple, also strips the provided string of text of its original structure, punctuation and word order.

Weights, n-grams and continuous vector space

Although the bag of words model creates a simple framework for comparing documents it has some flaws and imprecisions. The representation is made under the assumption that having the same words in a documents means that they are similar in sentiment or meaning. This is a flawed assumption seen as the two sentences '*I like water but not beer'* and *'I like beer but not water'*, would have the same bag of words representation but not the same meaning. The issue with measuring the distance between the two sentences is beyond word order though. If the model only measured the distance based on word presence the sentences would still be similar as four out of six words would match regardless of how much informational value the words themselves provide to the sentence. The bag of words representation also does not take word ambiguity into account. Word ambiguity is the

problem of distinguishing between polysemes i.e. words with multiple meanings. The problem arises if for instance ‘a light’ is confused with the verb ‘light’.

There has been developed several tools to account for the imprecision of the bag of words representation by adding some complexity and weights to the basic representation.

The vector representation can be weighted. Instead of a binary annotation in the vector space of whether or not a word from the overall dictionary exists in a given text, the annotation can note the term frequency of the word in a text i.e. how many times does the document or sentence contain a specific word. This representation is also flawed because words do not appear according to importance. The most common words in a dataset or text are usually function words and pronouns (Manning & Schütze, 2003). In order to accurately weight the terms in order of importance, one can employ the principle of Zipf’s law on top of the term frequency. According to Zipf’s law the frequency of a word in one text is inversely proportional to its frequency in the overall data set of texts. This law can be used to weigh a vector representation according to both the term frequency in the text and the inverse document frequency of the whole dataset - or tf-idf for short. In doing this words or terms such as common auxiliary verbs like ‘be’ would be ranked lowly as they appear often in all documents, where words which appear a lot in one document but not overall in the the rest of the data set would be ranked higher (Manning & Schütze, 2003). Another simple approach to handle often occurring words with low informational value is to remove them. This can for instance be done using a list of common words.

As mentioned the bag of words with a one hot vector encoding approach is problematic. This is because the approach assumes the words in a sentence to be discretely distributed i.e. they appear independently of each other in the text (Provost & Fawcett, 2013). This poses a problem in cases such as multi-word expressions like ‘United States’, ‘at once’ or ‘machine learning’. One solution to this is to employ n-grams. N-grams are of a continuous sequence of *n* words from a given text. In the n-gram framework the basic bag of words model already have n-grams in the shape of unigrams, where only one word at the time is considered. The length of the sequence of items considered at once could be expanded to for example bi-grams or tri-grams to catch multiword expressions (Manning & Schütze, 2003).

Another problematic example concerning the discrete assumption of words is that semantically closely related words such as ‘red’ and ‘blue’ would not be considered similar or close by the simple vector space model even though both words are descriptive colors. The reason for this lies in the basic idea of a one hot encoded vector. Take a given a data set which has the three words ‘red’, ‘blue’ and ‘yellow’. They might in the overall dictionary of words be placed as [‘red’, ‘blue’, ‘yellow’]. In this the one hot encoding for the word ‘red’ would be [1,0,0] where the word ‘blue’ would be represented in the vector space as [0,1,0]. These are two very distinct vectors from which the semantic relation is not clear (Provost & Fawcett, 2013). Mikolov et al (2013a; 2013b) has been investigating this issue and proposed

the idea of continuous vector space models. The framework within continuous vectors assumes that semantically similar words would appear in similar environments i.e. around similar words. To analyse terms for similar environments, continuous vector space theorist for instance employ skip grams. Skip grams are an advanced form of n-grams, where the model analyzes the whole dataset and for each word considers the surrounding words within a defined range. The model then trains a neural network to predict the probability of each word to appear near the focus word (Mikolov et al, 2013a). This model can be used to map or embed words near each other which under the aforementioned assumption would be semantically similar yet in the one hot encoding represented as different.

Lastly, I mentioned the issue of word ambiguity in texts and how to represent this. Word ambiguity is often handled in text representation by tagging each word in the text with their so-called part of speech. This usually means annotating the word as either noun, verb, pronoun etc, but it can be more specific depending on how specific an annotation is needed. This kind of tagging is often referred to as POS-tagging (Croft et al, 2009).

Machine Learning

Supervised and unsupervised learning

Machine learning within computational linguistics is used to process and recognize patterns in text and more recently speech. Based on these patterns the machine, code or machine learning system should be able to create new rules or descriptions to better understand the given data and handle unseen data in the future. This process of creating new rules is considering the learning in machine learning. There are a variety of methods developed for machine learning with the clear distinction between what is known as supervised and unsupervised machine learning.

The main difference between supervised and unsupervised learning is whether the data has a target value i.e. is labelled (Provost & Fawcett, 2013). Supervised learning methods use data, which is labelled with a clear target value. The learning algorithms then attempt to sort the data according to the given target values. Within supervised learning there are two main subclasses, which are distinguished based on the type of target the machine is learning distinctive patterns for. If the target is categorical the task is usually a classification task. If the target value of the data is numeric the machine learning is usually a regression task (Provost & Fawcett, 2013). Typical supervised machine learning methods include decision trees, linear classifiers such as support vector machines or perceptrons and probabilistic classifiers like the Naive Bayes method.

Within unsupervised learning the data is unlabelled and the algorithm is used exploratively to find potential patterns within the data. This means that compared to supervised learning, where the machine learning method has been informed about the target values and thus looks

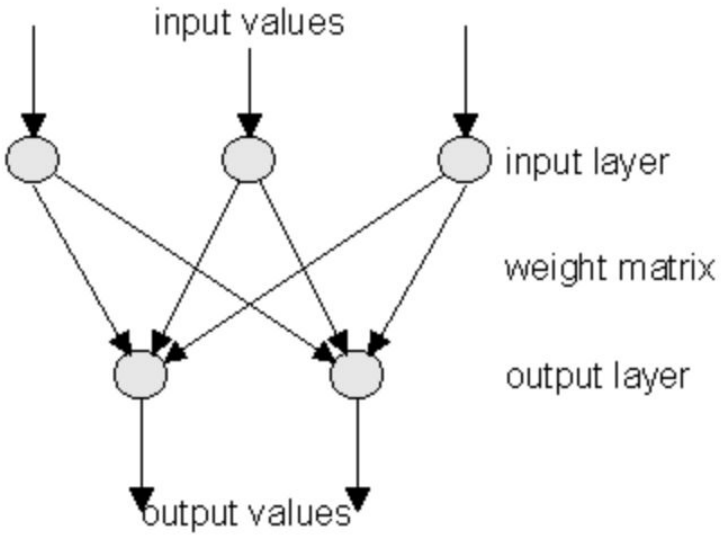
for pattern fitting these, the unsupervised methods attempt to create and sort its own groups or otherwise describe the data provided. Clustering methods such as k-means is a typical unsupervised learning method (Provost & Fawcett, 2013).
The task for this project is to classify speeches according to their political affiliation. The target values are known and therefore the machine learning task is a supervised classification task.

Neural Networks

Neural networks are machine learning methods, which draw their learning inspiration and name from the human brain (Bishop, 2006). The overall structure of a neural network usually consists of so-called layers of neurons. Each neuron then have weights between them of different strengths. These weights are considered the connections, which all together create the neural network structure. The learning of the model happens as the weights are dynamically shifted in each pass over the training data to strengthen and weaken connections between neurons in the network.

An early and comparatively simplistic version of a neural network is the machine learning method known as the perceptron (Rosenblatt, 1958). The perceptron is a linear classifier, which is often used for supervised learning. It is a binary classifier, which uses and optimizes linear prediction functions combined with feature vectors as input values to classify data. It assumes that the data points in a given data set are linearly separable and thus attempts to classify the data by finding the best dividing line between the data points (Bishop, 2006).

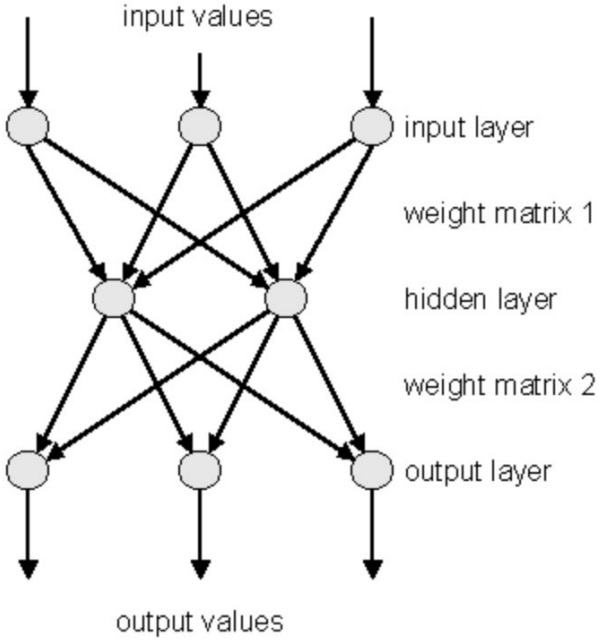
Model 1 - Perceptron (Fröhlich, 2018)



Model 1 is a depiction of a basic perceptron model, where data would be fed to the model in the nodes or neurons in the input layer. The input layer is then assessed and the weights are applied to pass the data to the output layer. By each iteration over the data points, the weights would then be corrected as the machine learned (Rosenblatt, 1958).

To more accurately depict the full scope of a neural network in its classic sense, the perceptron model needs to be extended into a basic neural network model called the multilayer perceptron. The multilayer perceptron has the same basic framework as the perceptron, but in addition to the input and output layer, it will have one or more neuron layers in between. These are commonly referred to as hidden layers (Bishop, 2006).

Model 2 - The Multilayer Perceptron (Fröhlich 2018)



Model 2 represents a basic neural network. Note that both the number of neurons as well as the amount of hidden layers may change depending on the nature and depth of the specific implementation.

Neural networks may be used for both supervised and unsupervised learning depending on the intention of the research and or nature of the learning task. Two typical current neural network models, which are relevant for computational linguistics, are the convolutional and recurrent neural networks, which I will briefly cover below.

Convolutional neural networks have primarily been used for image processing, but have also been applied within computational linguistics (Severyn et al, 2015; Poria et al, 2015; Kim, 2014). In this type of network the input data is sent through convolutional layers, which differs from normal layers in neural networks, as the nodes in the network are not necessarily all connected (LeCun et al, 1998). In convolutional layers neurons only concern themselves with the neighbouring neurons. As a result the deeper layers may shrink.

Recurrent neural networks are the type of neural network, which is usually associated with handling text and speech. Recurrent neural network are feedforward, which means that the

data moves in one direction from input to the output layer. The recurrent neural network have an extra feature to its feedforward methodology in that some information is retained in the neurons by each pass through of training data (Elman, 1990). This adds a temporal aspect to the network, where for instance the order of which the data is fed to the algorithm, may change the result of the training. The temporal aspect may create a problem of what is called a vanishing gradient, where the network loses information in especially the deeper layers (Elman, 1990). To combat this the recurrent neural networks have been combined with internal networks called Long Short Term Memory (LSTM). LSTMs create gates in the neurons, which help control the flow of information i.e. how much information is passed on between nodes and how much information should be retained or forgotten (Hochreiter & Schmidhuber, 1997). Recent research have also employed a variation of the LSTM called Gated Recurrent Units (GRU). GRU function similarly to the LSTMs but tends to be slightly faster but less expressive (Chung et al, 2014).

Sentiment Analysis

What is sentiment analysis

Sentiment analysis is a subfield of computational linguistics which uses semantic analysis. Sentiment analysis research use computational methods to study and extract the sentiment and opinion of an entity and their attribute from natural language texts (Liu, 2017). The terms opinion and sentiment are used interchangeably within the literature but in general sentiment analysis investigates the feeling towards a topic, where opinion mining investigates the writer's opinion regarding a topic.

Although the sentiment analysis as part of computational linguistics is an interdisciplinary field of research, the methodology and framework originates from computer science (Liu, 2017). This is apparent as research within sentiment analysis use the empiricist approach of probabilistic models to solve the problem of detecting sentiment. Linguistic theory is still applied but in a manner which seeks to understand and describe the underlying issue of the task and understand the structure and ambiguities of language. Liu (2017) further elaborates that the direction of sentiment analysis research in part is because current linguistic knowledge cannot be effectively operationalised to perform computational sentiment analysis. He theorizes that this is because current computational technology’s understanding capability still is not on level with that of a human and because most linguistic knowledge is not meant for computational use (Liu, 2017). Furthermore, sentiment analysis tasks have typically been binary in nature.

Sentiment analysis is as mentioned in the introduction section above a young research field which emerged in the early 2000s. The methodology consists of three main levels of analysis - document, sentence and aspect level analysis. In a document level sentiment analysis the model investigates the sentiment of a full document. An example of this is the process of

classifying positive and negative product reviews. Sentence level analysis is especially used within research on subjectivity in texts. It usually involves assessing the individual sentences as being positive, negative or neutral. Document and sentence level both investigates sentiment on a general level, however, neither level of analysis considers the target of the sentiment. Aspect level analysis investigates sentiments and attempts to map them with the associated target topic (Liu, 2017).

Assumptions and definitions

Below I will go through the terms within sentiment analysis as it is described by Liu (2017). Because the projects attempts to classify txt-documents according to political tendency, I will specifically describe Liu's (2017) definition of an opinion as well as the underlying assumptions in document level sentiment analysis.

Practical implementations of sentiment analysis have a simplistic framework wherein they have defined opinion as consisting of five parts. Firstly an opinion has a target entity or aspect of an entity upon which the opinion or sentiment is based, secondly the opinion has a sentiment. Within polarity research this could for instance be positive or negative. Then the opinion is owned or held by a person, and lastly there is a specific point in time, when this opinion was made or expressed. This gives the framework the five components entity, aspect, sentiment, opinion holder and time, which for short gives the expression (e,a,s,h,t). This simplified expression of an opinion covers a wide range of the different aspects in sentiment analysis research. Not all implementations may need to make use of all five components of the definition. For instance within brand management, the manager may only be interested in mining or finding opinions about specific products (entities), and therefore choose to assess opinion as a quadruple rather than a quintuple term.

Document sentiment classification seeks to classify opinion documents according to their expressed sentiment. This specifically means that the method considers the opinion document as a whole entity, when analysing its sentiment. This furthermore means that other components of the aforementioned quintuple definition of an opinion such as entity and aspect, are largely ignored in the sentiment analysis process (Liu, 2017). Document sentiment classification is considered the simplest task within sentiment analysis as it considers sentiment classification to be of the same nature as traditional text classification tasks. The target values or labels found in a regular text classification tasks are in document sentiment classification cases considered to be equal to sentiment orientations or polarities. This research therefore often employ the same machine learning models as seen in regular classification tasks. Because of its simplicity document sentiment classification is estimated to be the branch of sentiment analysis, which have been most extensively researched (Liu, 2017). To ensure that equating document sentiment classification with traditional classification tasks is meaningful in practical applications, research have assumed that a

given opinion document only expresses opinions on a single entity and from one opinion holder alone.

If we combine this assumption with the definition of an opinion as being made up of five components, we can determine the opinion of a document, where the entity is considered as a whole, with all aspects as given by (_,general, s, _, _) (Liu, 2017). Note that in this definition the temporal aspect of an opinion is also considered irrelevant.

Methods for sentiment analysis

Most existing techniques for sentiment classification use supervised learning models (Liu, 2017). Regular sentiment classification tasks usually consist of a polar framework with positive and negative labels.

Computationally sentiment analysis makes use of all of the methods mentioned above such as lemmatisers or vector space representations. Markov models which make use of n-grams is also common in for instance aspect and entity extraction (Liu, 2017). Because sentiment analysis traditionally have done extensive research in polarity tasks which involve classifying positive and negative data points, research have typically made use of for instance sentiment lexicons. Sentiment lexicons often consists of adjectives and adverbs as they usually denote relevant information in polarity tasks. As sentiment classification is a text classification task most regular tools for textual analysis and representation and machine learning, which was covered in the previous sections can be employed. A notable example of this is Pang et al (2002), who classified movie reviews using a unigram representation of the reviews and classified them using the probabilistic model Naive Bayes and a support vector machine.

Why sentiment analysis

Although sentiment analysis historically have focused on binary polarity tasks, the field have branched beyond this topic and engaged in for instance opinion summarization, intention mining, detecting fake sentiment expressions as well as analysing online debates (Liu, 2017). These tasks are in some respects more complex than regular sentiment classification of positive and negative texts. One of the reasons for this is that in regular polarity tasks, the learning method can to some extent rely on typical markers of sentiment polarity such as positive or negative adjectives. Such markers of polarity are not as widely present in e.g. political speech and thus makes the task more complicated Diermeier et al (2012). Pla & Hurtado (2014) approach this complexity in classifying political sentiment. They attempt to identify the political identity of twitter users under the assumption that political statements are expression of sentiments and that opinion holders of similar political sentiment would share the same sentiment on the same topics.

Danish and language technology

The Danish population is highly connected on the internet. According to the Digital Economy and Society Index Denmark has an estimated internet penetration of 95% as per 2017 and is the most digitised country in the European Union (DESI, 2016). The index further shows that 86% of Danes consume their news online and 78% use social media. In a recent media development analysis of Denmark it was noted that although traditional media such as TV still remained the main source of information and news, traditional media was in decline and especially young adults under 35 used Social Media as their main source of information (Wandsøe-Isaksen et al, 2018). Furthermore, Denmark has compared to other nations a media landscape with a low rate of polarisation (Wandsøe-Isaksen et al, 2018).

Interestingly, even though Denmark is ranked as the most digitised country in the European Union by DESI (2018), a recent study made by the Multilingual European Technology Alliance (META) of the available language technology resources for the European languages, showed that Danish language technology within most categories only has fragmentary support, as research within language technology for the last 50 years has mainly been focused on English (Rehm et al, 2012).

Danish is the official language spoken in Denmark and part of the mainland scandinavian languages (Rehm & Uszkoreit, 2012). There are approximately five million native speakers and Danish can therefore be considered a relatively small language compared to e.g. English or French. Despite Denmark being a very digitized nation (DESI 2018), there are few available tools for automatically analysing Danish compared to the research done analysing English. There are, however, traits and features, wherein Danish is different from English.

First, Danish has a large flexibility in creating compound nouns (Rehm & Uszkoreit, 2012). This would for instance make unigram one hot vector encoded implementations such as simple bag of word representations vulnerable to new words created by compounding nouns together and possible inconsistencies in when and to what extent the nouns have been compounded. Danish furthemore uses semi-lexicalised particles (Rehm & Uszkoreit, 2012), whose semantic meaning and syntactic use also could be lost in a unigram vector representation. Lastly, Danish, like the other scandinavian languages, allow a lot of syntactic movement on a sentence level (Rehm & Uszkoreit, 2012). This could be problematic for models, whose representation depends on word order or sentence structure. The bag of words representation would not struggle handling the free syntactic structure of Danish, as the linguistic information such as sentence structure is removed in the bag of word model's vector representations.

Literature review

The following section covers the literature review, which creates the basis of the project in terms methodology and scope. The review is split in three parts: A review of the current research within sentiment analysis using neural networks, the research within Danish textual analysis and lastly the available machine learning frameworks available using Python.

Sentiment analysis using neural networks

The literature review for sentiment analysis and neural networks was done via a targeted search on Google Scholar. The search was conducted using term such as *Sentiment analysis*, *Neural networks, Opinion Mining* and *Political affiliation*, and combining the queries. The initial search results were assessed based on their title and abstract as well as their origins. This means that papers from top ranked publishers and conferences according to Google Scholars h5-index (Google, 2018) were prioritized. Because the research field of sentiment is only 20 years old (Liu, 2017) newer articles were prioritised. This process yielded initially 67 interesting papers, whereof 34 were relevant for the literature review after being screened. The overall intention of the review was to gain an insight into current research methodology and tasks within sentiment analysis when mining for political opinion.

In the papers investigating automatic detection or classification political affiliation or ideology in texts, there is a clear tendency to utilise support vector machines (Yu et al, 2008; Pang et al, 2006; Dahllöf, 2012) or logistic regression (Recasens et al, 2013; Diermeier et al, 2011). The majority of the studies favor a binary distribution of instances. This may in part be due to the research being conducted within the field of sentiment analysis. As mentioned in the section above sentiment analysis to a large extent research polarity tasks, wherein they classify documents as either positive or negative. Considering this, it is natural that sentiment analysis studies within politics also would gravitate towards a polar data distribution, where the polarity lies in the political spectrum rather than a positive/negative distribution. Furthermore, the binary focus in the research may also be due to the research being in English, as two major English speaking countries, namely the United States and the United Kingdom, both have bipartisan parliaments. Almost all papers used simple majority of instances in their datasets as their baseline, and papers using support vector machines generally gained an accuracy between 70-90% on their data set. The other used methods achieved a score below 70%.

As I wanted to investigate how a deeper learning method such as a neural network would perform compared to the regular machine learning methods, the second part of the literature review focused on research within sentiment analysis using neural networks. As described in the theoretical section above recurrent neural networks are the most commonly used neural network for automatic textual analysis (Irsoy & Cardie, 2014; Iyyer et al, 2014; Liu et al,

2015), however, research using convolutional neural networks have also been utilised for word generation and sentence level sentiment analysis (Poria et al, 2015; Kim, 2014; Johnson & Zhang, 2015). All the research using neural networks emphasised the importance and value of preprocessing the textual input and almost all papers employed some variation of word embeddings. The investigations mainly used regular machine learning methods such as support vector machines as baselines (Johnson & Zhang, 2015). The investigations had an average baseline accuracy between 60-70% and managed to improve their results between 5-25% .

Collectively looking at the full literature review of sentiment analysis using neural networks with the intention of mining for opinion, it should be noted that only two of the relevant papers had specifically applied neural networks for detecting political tendency (Iyyer et al, 2014; Plat & Hurtado, 2014) and the most accurate results for sentiment analysis were achieved by regular supervised classification methods regardless of whether the task is for regular sentiment analysis or political tendency.

Furthermore, the studies all noted the difficulty in accurately classifying shorter documents, as sentiment analysis like other machine learning methods perform better with longer documents, which contains more informational value. A few papers specifically address the problem regarding the loss of word order structure in bag of word representations with Le & Mikolov (2014) specifically emphasizing, that having the same words in a documents does not mean, the opinion holders are of similar sentiment.

To summarize all of the literature reviewed most research use simple majority in the data set as a baseline. Furthermore, almost all of the reviewed papers were doing binary classification or assessment of the data using mainly support vector machines or logistic regression. Two articles in all of the literature reviewed used neural networks for investigation political affiliation.

Danish

The literature review exploring Danish as a research target for computational linguistics was initially performed using broad key terms on Google scholar. This returned limited relevant results and none which fully satisfied the original screening process of papers, which was used when conducting the literature review in the section above.

The immediate impression of the few relevant papers, that conducted Danish machine learning tasks, was a general usage of the text analysis tools and representations originally tested and developed for English machine learning tasks. Research has been done to create Danish versions of known English natural language processing tools such as DanNet and AFINN (Nielsen, 2018). Various other basic preprocessing level tools have been developed

with various degrees of success and prerequisites but only limited resources and research is available for Danish.

Some of the available literature note the limited resources available for Danish both in terms of available tools and the sparse amount of training data. An interview with the Danish Centre of Language Technology point out that Danish as such is a low resource language, because the underlying resources for language processing are lacking on a larger scale and unlike the other Scandinavian languages, Norwegian and Swedish, Danish have not invested in creating so-called language banks with resources for developing natural language processing (Ammitzbøl, 2013).

Most of the reviewed literature doing natural language processing of English actively employ previously tested datasets, frameworks and methods. An example of this are the pre trained word embeddings described and developed by Mikolov et al (2013a), which are employed in some variation or to some extent in the vast majority of the reviewed literature regarding neural networks, rather than each paper developing their own pre embeddings. Similarly English has several linguistic resources such as spaCy (Honnibal & Montani, 2017) and NLTK (Loper & Bird, 2002), who amongst others provide readily available tools for smaller, simple natural language processing tasks. Nielsen (2018) has an extensive list of the available resources for Danish textual analysis. In this the two libraries above are mentioned as having tools for Danish, however, when tested these tools only provided partial or crude support in Danish and lacked many of the functionalities available for English language processing. The Centre for Language Technology at the University of Copenhagen also has a range of available resources but not in a format, which is easily applicable to a large data set.

The two following sections describe the dataset used for the project as well as how and what machine learning methods were implemented in order to investigate the initial research question. To reiterate, this project seeks to investigate whether its possible to automatically classify texts according to their political affiliation using machine learning. Furthermore, I wanted to test how a deeper learning method such as a neural network would perform compared to more simpler natural language processing tools. Below follows the dataset description of the instances, distribution and features and how well the data set is in line with the basic assumptions set by the document sentiment analysis framework. Following the dataset description I will outline the methodology of my thesis by first defining the baseline of the project based on the literature review. I will then cover the my choice of machine learning methods as well as how the data was processed and prepared for training the algorithm.

Data

As previously mentioned Danish is a low resource language and has relatively little available, suitable data for the scope of this project. For a classification task depending on political

affiliation the optimal data set would contain several opinionated texts with a clear political affiliation. I therefore chose to collect speeches from meetings in the Danish Parliament. The Danish parliament contains 179 members and has 107 meetings annually on average, where the members discuss and debate legislation. As all members represent their individual party the correct label for a given speech should therefore represent the opinion and political affiliation of the individual speaker's party membership.

All meetings in the parliament are transcribed and publically available on the official website of the Danish Parliament, www.ft.dk. All speakers appear in the transcriptions with an annotation of which political party the speaker represents. The Parliament has been working on digitizing all their documents from 1850 until now, however, as per the time of writing the only available digitised minutes of the meetings in a usable format go back to 2007 (Varder, n.d.). The data has been collected by downloading the minutes from each meeting since 2007 as txt-files locally. This process will be described in further detail in the methodology section following the data set description below.

Instances and target value

The minutes of the meetings held in the Danish Parliament since 2007 were collected through www.ft.dk, and saved locally as txt files. Using a python script all 1183 meetings from 2007-2018 were then split into individual speeches. Procedural statements and statements from the members representing Faroese and Greenlandic parties were then removed. The latter was done as the four members from the Unity of the Realm represented their home nation as a whole rather than a party political opinion. This provided a data set consisting of 248.998 individual speeches. Each speech contains 210 words on average and the whole data set contains 52.401.911 words across 2.659.725 sentences.

All speeches were sorted into folders according to their individual party affiliation using a python script. This was in part done because the datasets package in sklearn, when loading the data for the classification task can assign the folder names as the target value for the speech automatically.

The target values for the classification tasks consist of the parties in the Danish Parliament. The format of the transcriptions of the meetings enabled me to automatically assign the label for each instance in the data set, rather than manually assessing all speeches and from them determine the political affiliation individually prior to the classification task.

When a speaker has been taken into office as a minister, their name appears without the normal notation of party affiliation. I chose to sort those speeches according the party, they belonged to when they became ministers. A full list of ministers in government in the Danish Parliament is available through the website of the prime minister's office (Statsministeriet, n.d.).

Since 2007 there have been ten parties elected to the Danish Parliament, giving the following target values for the classes listed below, with their corresponding abbreviations in the transcriptions. The parties will henceforth be referred to using their associated abbreviation.

Table 1

ALT	Alternativet	DF	Dansk Folkeparti
EL	Enhedslisten	KD	Kristendemokraterne
RV	Radikale Venstre	KF	Konservative Folkeparti
S	Socialdemokraterne	LA	Liberal Alliance
SF	Socialistisk Folkeparti	V	Vestre

As most of the available research was performing binary classification, I wanted to divide the parties into two reasonable groupings to test the machine learning methods in a binary environment as well as a multiclass environment. Traditionally in public Danish media the parties are referred to as belonging to either 'red' or 'blue' block (Larsen, 2015). In this discourse the red block refers to left wing socialist parties, and blue block harbours liberal and conservative parties.

The Danish data analysis firm Buhl & Rasmussen use machine learning to structure and visualize political data from the Danish Parliament, European Union, Danish municipalities and more. One of their projects collect voting data from the website of the Danish Parliament and use it to analyze and afterwards visualize data about the different Danish parties based on their voting statistics in the more than 1.5 million vote casts in the Danish Parliament since 2001 (hvemstemmerhvad A, n.d.).

One of the visualizations analyse the individual position of the Danish parties relative to each other based on their cast votes in the Danish Parliament in the time period 2001-2011. The individual position of each party is calculated using a statistical model called the Markov Chain Monte Carlo. Using this parties who vote similarly will be placed close to each other and vice versa if they do not agree in their votes (hvemstemmerhvad B, n.d). This created a division of the parties, which supported the general notion of a left and right wing.

I combined this with which candidate the parties supported as prime ministers after each election since 2007. In the period 2007-2018 the prime ministers have either been from S, who belong to the left wing of Danish Politics, or V who traditionally belong to the right wing of the scale (Statsministeriet, n.d.). This provides the following distribution of parties within the target values Left and Right.

Table 2

LEFT WING					RIGHT WING				
ALT	EL	RV	S	SF	DF	KD	KF	LA	V
5.258	29.236	16.547	50.193	24.041	36.670	136	19.993	14.581	54.343
2,11%	11,74%	6,65%	20,16%	9,66%	13,92%	0,05%	8,03%	5,86%	21,82%
125.275					123.723				

In an ideal situation all parties within the Danish Parliament would have an equal participation rate and thus an equal amount of instance in the data set. However, because the number of members in parliament for each party vary, so does their participation in the debates. The overall distribution of instance can be seen in table 2. For a more detailed overview of the distribution of instances, please see the appendix.

Figure 1

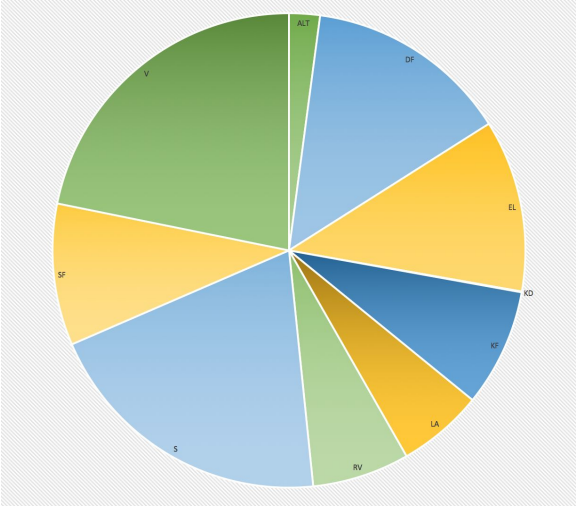
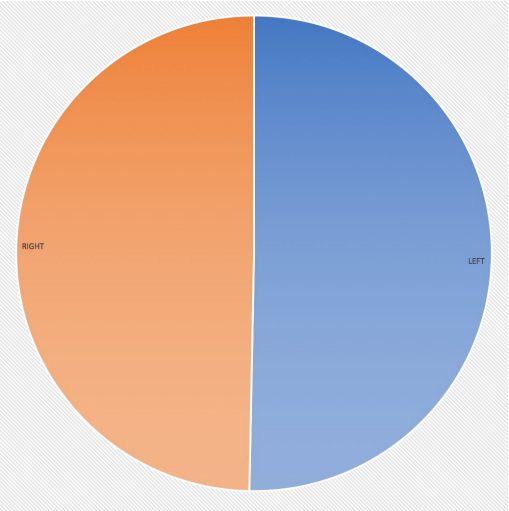


Figure 2



Considering both table 2 and figure 1 it is clear, that there is an uneven distribution of instance in the data set with parties such as V and S, who both have ten times as many instances as ALT and 420 times as many instances as KD. For both ALT and KD their small numbers are to be expected seen as ALT only got elected into parliament in 2014 and KD had one member in parliament in 2009-2010.

The data set as a whole however seems to have an almost perfect distribution if the dataset is split according to left and right, and the distribution of instances would therefore need to be assessed independently for binary and multiclass classification, which I will do below in the entropy assessment.

Entropy and information gain

The data set is not evenly distributed and would most likely benefit by removing at least two of the possible classes from the data set. This can be calculated using the terms entropy and information gain. Within machine learning entropy and information gain is used to asses the randomness and balance of the dataset and potential information gain in partitioning the data set. Entropy is given by the expression

$$Entropy = \sum_{i=1} p_i \log_2(p_i)$$

Where p is the number of instances in the data set divided by the full number of instances. The expression for entropy can be used to calculate the randomness and purity of a given dataset. Information gain relates to entropy in the sense that, information gain measures how informative a split of a data set is concerning the target value. It is calculated by finding the entropy of the partitions themselves and subtract their accumulated amount from the entropy of the full data set.

A dataset where all instances belong to the same group, would for instance have an entropy of zero as it would be a completely homogeneous dataset. In my partitions into the ten political parties, I assume all speeches belong and represent their allocated party and as such, the entropy of the individual party partition would be equal to zero and thus information gain would be the same as the entropy of the full data set.

In order to assess the balance of the data set and compare the entropy of the data set with and without the two smaller classes ALT and KD, their entropy was compared to what the entropy of the data set had been, had the classes been evenly distributed in either case and compared the numbers.

Table 3

Distribution	Calculation	Entropy	Difference
Ideal 10 class	$-\left(\frac{1}{10}\log_2\left(\frac{1}{10}\right)+\frac{1}{10}\log_2\left(\frac{1}{10}\right)+\ldots+\frac{1}{10}\log_2\left(\frac{1}{10}\right)\right)$	3,3219	0,3771
Actual 10 class	$-\left(\frac{5258}{248998}\log_2\left(\frac{5258}{248998}\right)+\frac{34670}{248998}\log_2\left(\frac{34670}{248998}\right)+\ldots+\frac{54343}{247998}\log_2\left(\frac{54343}{248998}\right)\right)$	2,9448	10,4%
Ideal 8 class	$-\left(\frac{1}{8}\log_2\left(\frac{1}{8}\right)+\frac{1}{8}\log_2\left(\frac{1}{8}\right)+\ldots+\frac{1}{8}\log_2\left(\frac{1}{8}\right)\right)$	3	0,1478
Actual 8 class	$-\left(\frac{34670}{243604}\log_2\left(\frac{34670}{243604}\right)+\frac{29236}{243604}\log_2\left(\frac{29236}{243604}\right)+\ldots+\frac{54343}{243604}\log_2\left(\frac{54343}{243604}\right)\right)$	2,8522	4,9%

As can be seen from the calculations above the difference in entropy is more than halved by removing ALT and KD from the full data set. As this thesis also investigates a binary

distribution, the difference in entropy has also been calculated for a binary distribution before and after removing the two parties.

Table 4

Distribution	Calculation	Entropy	Difference
Ideal	$-\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right)+\frac{1}{2}\log_2\left(\frac{1}{2}\right)\right)$	1	
Binary before	$-\left(\frac{125275}{248998}\log_2\left(\frac{125275}{248998}\right)+\frac{123723}{248998}\log_2\left(\frac{123723}{248998}\right)\right)$	0,99997	0,00003 0,003%
Binary after	$-\left(\frac{120017}{243604}\log_2\left(\frac{120017}{243604}\right)+\frac{123587}{243604}\log_2\left(\frac{123587}{243604}\right)\right)$	0,99985	0,00015 0,015%

Interestingly the original dataset is almost perfectly balanced before removing ALT and KD, and the difference from the ideal entropy is five times larger, when removing the two parties. It would therefore based on the balanced calculated using entropy be optimal to perform binary classification on the full data set, but multiclass classification on a data set where ALT and KD have been removed.

Document sentiment classification assumptions

As was described in the theory section regarding sentiment analysis, the general theory of document sentiment classification consists of a general underlying assumption for the task to be meaningful: The document in question have to be of a singular opinion (i.e. only belonging to one class or label) and from a single opinion holder. This is reflected in the current format of the data set in the following way. Each document consists of one individual speech. Thus the requirement of the individual document being of a singular opinion holder is fulfilled. In this project I choose the assume that each speaker fully represents the party they are a member of, and thus that their speech is of a singular opinion and representative of the party of the speaker.

Features

The features of the data set are the words and sentences in the speeches. Machine learning methods cannot handle text directly. The txt-documents must therefore be transformed into a different format. As described in the theory section above the common way to do this is to transform the sentences and words into vector representations, where the individual words, sentences or characters have been changed into numbers according to an overall document library dictionary.

In this project each vector regardless of machine learning implementation will represent a full document, where the words have been tokenized. As such each word become a feature of the vector. I tested some of the tools mentioned by Nielsen (2018) for the vector representation. These tools include stemming and other early preprocessing tools such as PyStemmer (Boulton, 2006) and spaCy (Honnibal & Montani, 2017), however, all were only partially implemented for Danish or required a lot of extra prerequisites for the tools to work effectively. I therefore decided not to stem or lemmatize the data.

Methodology

Collecting the data

As mentioned in the data set description above the data set was collected by downloading the minutes from meeting in the Danish Parliament since 2007. After collecting all minutes I use a python script (clean_up.py) to go through all the minutes, change all words to lowercase, remove noise and split the minutes into individual documents for each speaker. The code works by looking at each line and the individual characters in that line. When the code discovers a line with a name of a speaker, it creates a new file named after the speaker and writes his or her speech into the file. After having split all the minutes into individual speeches, I use a python script to sort all the files into folders according to the party name in the file title (sort.py). The python script also contains special information for sorting ministers seen as their party no longer appears in the minutes after taking office. In the sorting all files containing moderators and members from the Faroese and Greenlandic were moved to a separate folder. Lastly, I discovered a few mistakes in the way the original clean_up.py split some of the minutes. I therefore created a new script to find said files so they could be deleted (slet.py). All python scripts have been included in the appendix.

In the testing I also investigate different partitions of the data set depending on the year the speech was held. To create the corresponding data set, the above process was repeated for eleven iterations with the minutes partitioned after the year they belong to. This provided four general data sets.

1. All speeches divided according to party affiliation (multiclass)
2. All speeches divided according to wing affiliation (binary)
3. All speeches divided into separate years and according to party affiliation (multiclass)
4. All speeches divided into separate years and according to wing affiliation (binary)

Baseline

In order to reflect the research assessed in the literature review, I have chosen to set the baseline according to the largest segmentation of data points in the data set. As almost all the

relevant papers were doing binary machine learning tasks, I also chose to split the learning task:

1. Machine learning on a binary data set
2. Machine learning on a multiclass data set

In the first task the data set will be split into a binary partition according to whether the speaker belong to the left or right wing of the Danish political scale. This partition was also described in the data description section. The aim of the binary task is to be able to compare the results on Danish sentiment analysis with current research.

The second task keeps the parties separate of each other. The learning task therefore becomes an multiclass task with eight target values. The target values here correspond to the parties described in the previous section without ALT and KD. The intention of the eight split partition is to see how well the learning algorithm will perform on classes, which are possibly not as clearly separable as in the binary task.

As stated in the beginning, the target value which represents the majority of the instances in the data set will be used as the baseline for the learning tasks. This principle provide a 21.82% baseline for the multiclass learning task and 50.31% for the binary task.

Machine learning methods

This section describes the machine learning methods chosen for the project. The methods have been chosen in line with the findings of the literature review to enable a comparison between the results of this project and current research. As the target values of the data set are known, this is a supervised learning task.

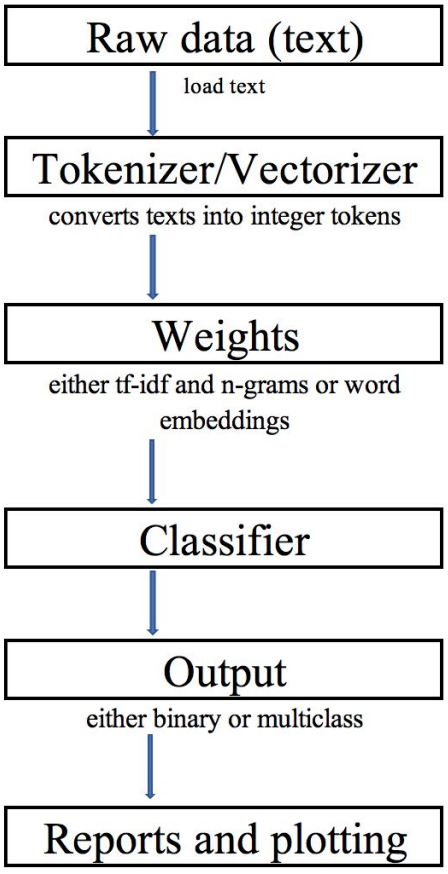
Reflecting the papers from the literature review, I will be implementing a support vector machine as this was the most prominently represented learning method used in the papers investigating political affiliation recognition. Furthermore, to test a deeper learning method I chose to implement a neural network using LSTM. As only one of the papers in the literature covered used neural networks for identifying political affiliation, the implementation was inspired by the literature performing general sentiment analysis with neural networks. As I am investigating whether or not it is possible to do this type of classification on Danish texts and not as such improving an existing method or setup, I have implemented both the support vector machine and the neural network in a simplistic, general manner. This enables me to investigate and compare the performance of both models without too many specifications, that could affect performance either negatively or positively.

Coding

Both the support vector machine and the neural network is trained and tested on the same data although the preprocessing will depend slightly on the individual implementation. All results was processed and presented using the metrics library from Scikit Learn.

For the initial testing I have three main python codes which represent the implementation of the support vector machine, the neural network for binary classification and the neural network for multiclass classification respectively (see appendix for codes named mySVM.py, my5thRNN.py and mymultiRNN.py respectively). All three implementations have the same basic overlying structure.

Model 3 - General code structure



After importing the relevant libraries depending on the implementation (either Scikit Learn or Keras), the first part of either code loads the data and create the training and testing data set as well as the target names. All data was initially supposed to be loaded using Scikit Learn's load_files functionality, which provides a framework to easily handle the data points and their corresponding target value. There did however turn out to be a problem with the load_files functionality as the programming was done on a Mac. In the Apple operating

system all folders contain a file called .DS_Store, which is short for Desktop Services Store. The file contains attributes of its containing folder and cannot be deleted. The problem with including this file, other than the obvious problem of it becoming part of the data set and a directory file being used for classification, was that it contains characters which could not easily be read using the UTF-8 encoding the rest of the files were saved in. This is a problem in general for Mac users, and I was therefore able to find a revised version of Scikit Learn's load_files method on Github, which I used instead for loading the data into the code. The revised load_files.py method was used to read data for all machine learning implementations and have been included in the coding appendices.

Example of loading the data set from the multiclass dataset 'taler'

```
print('Getting the data...')
folders = glob.glob('taler')

for f in folders:
    Pol_data = load_files.load_files(f,
                                     ignore_files='.DS_Store',
                                     encoding='utf-8',
                                     shuffle=True)
#Preparing dataset
x = Pol_data.data
y = Pol_data.target

X_train, X_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.1,
                                                    random_state=42)
```

After being saved as a variable the data set was split into a testing and training set at a 90/10 split for testing using the scikit.datasets library.

Support Vector Machine

The support vector machine is commonly referred to as an SVM is a linear classifier (Provost & Fawcett, 2013). The classification is performed by creating linear discriminants to separate the classes. The best performing line has the highest margin to the classes.

The support vector machine for the project is implemented using the framework provided by Scikit Learn. Due to the size of the dataset I am specifically using the linear support vector classifier, as it has a reasonable amount of flexibility when training on the dataset compared to Scikit's regular support vector classifier. The classifier has been implemented using initially balanced weights and Scikit Learn's pipeline function (see mySVM.py lines 35-42).

In the literature the SVM implementations used bag of word representations using unigrams or bigrams to represent the texts in the data set. There was no apparent discrepancy in whether or not the papers had chosen to calculate the inverse document frequency in the

preprocessing of the data points. For the SVM implementation the data set was changed into bag of words with bigrams using Scikit's tfidfVectorizer. The features are thus tf idf-weighted bigram vector representations of the words in the individual documents. After setting up the pipeline the method was fitted on the training data partition of the dataset.

Example from setting up the SVM

```
print('\nSetting up pipeline for SVM')
from sklearn.svm import LinearSVC as SVC
clf = Pipeline([
                    ('vect', TfidfVectorizer(ngram_range=(1,2))),
                    ('clf2', SVC(multi_class='ovr',
                    loss='hinge',
                    class_weight='balanced')),])


print('Fitting the data..')
clf.fit(X_train, y_train)
```

As can be seen the classifier is set up using 'hinge' as its loss function, which according to the Scikit Learn documentation (Pedregosa et al, 2011) is the standard for support vector machines. The balanced class weight uses the target values to automatically adjust weights inversely proportional to their class frequencies in the input data.

Neural Network

As covered in the theoretical section above regarding machine learning and neural networks, there are two typical implementations of neural networks within natural language processing - the convolutional neural network and the recurrent neural network. The literature review showed an even distribution of research using either type, however, the research employing a convolutional neural network mainly investigated sentiment analysis tasks on a sentence level or on ultra short texts such as tweets.

Comparatively, research involving recurrent neural networks are more focused on regular document sentiment analysis. Furthermore, the papers investigating the effect of for instance word embeddings (Liu et al, 2015) or combining a neural network with binary dependency trees (Nguyen & Shirai, 2015) all use recurrent neural networks. For those reasons and because recurrent networks, as mentioned, has been the type of neural network usually associated with natural language processing, I chose to build a recurrent neural network as well. To combat the risk of losing information in the hidden layers due to the possible vanishing gradient problems, there has to be implemented internal gates for the information flows in the neurons. Literature suggest either using LSTMs or GRU for this as they should wield similar results. Because almost all of the assessed literature combined their recurrent neural network with LSTM, I have chosen to do the same.

I have initially worked with the machine learning framework Scikit Learn (Pedregosa et al, 2011) for automatic text classification. Scikit Learn does not provide the tools for an efficient neural network implementation. The one neural network Scikit Learn does provide is the multilayer perceptron classifier, which after 12 hours still had not completed its 5 epochs of training on the available computer. Furthermore I wanted more control of the implementation of the neural network, than what was available from the Scikit Learn framework, but still remained simple in its API, as I have never worked with neural networks before.

I therefore googled frameworks for neural network and compared them according to the amount of available documentation and the simplicity of use. The two possible frameworks I found, which could replace Scikit Learn were Theano (Theano, 2016) and Tensorflow (Abadi et al, 2015). Both these frameworks had extensive documentation and an active Github environment, but in addition to this Tensorflow has two wrappers namely Keras (Chollet, 2015) and TFLearn (TFLearn, 2016). Both these wrappers provides a high level application programming interface and builds on top of the Tensorflow framework. Keras is, however, a more general purpose application programming interface, which to some extent also functions with Theano.

I chose to make an implementation with both wrappers initially and tested them both with a smaller portion of the full data set. Both the implementation using TFLearn and Keras contained an initial embedding layer, where pretrained embeddings were loaded as weights. They then had two hidden layers with LSTM with a dropout rate of 0.5 to avoid overfitting (Srivastava et al, 2014). Overfitting is when the machine learning method fits its weights and parameters too closely to the training data, so it fails to generalize enough to accurately classify unknown data.

Layer (type)	Output Shape	Param #
layer_embedding (Embedding)	(None, 664, 300)	15000000
lstm_1 (LSTM)	(None, 664, 100)	120300
dropout_1 (Dropout)	(None, 664, 100)	0
lstm_2 (LSTM)	(None, 664, 100)	60300
flatten_1 (Flatten)	(None, 66400)	0
dense_1 (Dense)	(None, 2)	132802

Total params: 15,313,402
Trainable params: 15,313,402
Non-trainable params: 0

Because the training of the neural networks was very time consuming and took several hours, the initial testings of the implementations were done on a sample consisting of 5% of the full data set. Early trials showed that the network past six epochs would start to lose accuracy and the validated loss functions would increase rather than decrease. The implementation of both neural networks were therefore implemented using 5 epochs. The implementation using Keras outperformed the TFlearn implementation on both the binary and the multiclass task, wherefore Keras was used for the testing on the full data set.

As with the support vector machine the raw texts had to be transformed into a format, which could be processed by the neural network. Rather than creating bigram bag of word representations for the neural network, the Keras tokenizer was utilised to create a vocabulary of the 50.000 most popular words in the data set. The number was set at this level to ensure that all words in the data set were represented. The tokenizer functionality then provided mapping of all words in the vocabulary to individual integers. The speeches were then transformed into sequences of numbers corresponding to the mapped words in the dictionary.

Example from my5thRNN.py

```
print('Tokenizing text.. ')
#Tokenizing the text
num_words=50000
tokenizer = Tokenizer(num_words=num_words)
tokenizer.fit_on_texts(data_text)

x_train_tokens = tokenizer.texts_to_sequences(x_train)
x_test_tokens = tokenizer.texts_to_sequences(x_test)
```

Because the neural network require the input data to be of the same length the data points the texts had to be either padded or truncated to a set length. As mentioned in the data description the average length of each speech is 210 words, however, the range in speech length varies from 1-10.199, so simply padding all speeches to have the same length as the one large address to the parliament would not be reasonable. To decide upon a reasonable data size for the individual data points, the average size was multiplied with two standard deviations using numpy. This calculation suggested to equalize the size of the data points to 672 words, which would allow for 95% to be padded and the remaining five percent to be truncated without making the data points unnecessarily large.

All research in the literature review of neural networks emphasized the importance of word vectors, which aim to map semantic meaning into a geometric space. Unfortunately the size of the data set is not extensive enough to properly create word embeddings on it own. I have therefore chosen to add pre-trained word vector embeddings for Danish to the data set after tokenizing the data. The pre trained word vectors are 300 dimensions matrices and have been trained on Danish wikipedia articles using the fastText library (Bojanowski et al, 2017). The

fastText library is a skip gram based version of Mikolov's word2vec framework (Mikolov et al, 2013a).

After having tokenized the texts and padded or truncated the data, the embeddings are loaded in. When reading the embeddings file, the code also creates an index of the embeddings, which afterwards is combined with an index of words in the data set created by the tokenizer. This is done to create an weighted embedding matrix for the first layer of the neural network.

Example from my5thRNN.py

```
print('\nLoading embeddings...')
embeddings_index = {}
f = codecs.open('cc.da.300.vec', encoding='utf-8')
for line in tqdm(f):
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
print('found %s word vectors' % len(embeddings_index))

#Creating embedding matrix
print('preparing embedding matrix...')
embedding_size = 300
embedding_matrix = np.zeros((num_words, embedding_size))

for word, i in tqdm(idx.items()):
    if i >= max_tokens:
        continue
    embedding_vector = embeddings_index.get(word)
    if (embedding_vector is not None) and len(embedding_vector) > 0:
        #words not found in embedding index will be all-zeros
        embedding_matrix[i] = embedding_vector
print('number of null word embeddings: %d' % np.sum(np.sum(embedding_matrix,
axis=1) == 0))
```

Keras' Sequential model is used to create the recurrent neural network. The main difference between the binary (my5thRNN.py) and the multiclass (mymultiRNN.py) is in the final layers of the method. In the dense layer the difference is in the activation function which for the binary task performed the best with a sigmoid function and with a softmax activation for the multiclass classification.

Likewise in the choice of loss function i.e. the function we want to minimize while maximizing the accuracy, which uses binary cross entropy and categorical cross entropy for the two tasks respectively.

Example from my5thRNN.py

```
#CREATING THE RNN
print('\nStarting the RNN... ')
model = Sequential()
#Danish embedding
model.add(Embedding(input_dim=embedding_matrix.shape[0],
                    output_dim=embedding_matrix.shape[1],
                    weights=[embedding_matrix],
                    input_length=max_tokens,
                    name='layer_embedding'))

model.add(LSTM(units=100, return_sequences=True))
model.add(Dropout(0.5))
model.add(LSTM(units=100, return_sequences=True))
model.add(Flatten())
model.add(Dense(2, activation='sigmoid'))

#Different kinds of optimizers
optimizer = Adam(lr=1e-3)  #suggested by guide
optimizer2 = RMSprop(lr=1e-3)  #suggested by Keras documentation
optimizer3 = SGD(lr=1e-3, clipvalue=0.5) #suggested by github bc adam suffers
from diminishing gradients

model.compile(loss='binary_crossentropy',
              optimizer=optimizer,
              metrics=['accuracy'])

print(model.summary())
#TRAIN THE RNN
model.fit(x_train_pad, categorical_y_train,
            validation_split=0.1, epochs=5, batch_size=500)
```

Reporting the results

After both methods have been fitted to the training data, they are evaluated on the test data. In order to evaluate and visualize the performance Scikit's metric library as well as the pyplot function of matplotlib is used to create a classification report of the precision, recall and f1-scores as well as plotting the confusion matrix. For plotting the results into a confusion matrix I found a guide in Scikit Learn's documentation (Pedregosa et al, 2011) which provided a general method for pyplot, which I could use with my results.

Results

Below I will present and discuss the results and performance of both the support vector machine and the neural network. The results have been split into two parts. A general overview where the algorithm has been trained and tested on the full data set and a temporal analysis, where the data set have been split according to which year the speeches belong to.

For all results I have also created a confusion matrix of the predicted classification compared the true class of the data points in order to better understand, when and where the algorithms struggle to correctly classify the data. All figures and graphs have been created using the metrics package from the Scikit Learn framework as well as Microsoft Excel. I will discuss both method individually before making a comparison between the two.

Following the discussion of the results I will review the potential sources of error of this project in terms of the basic assumptions and limitations of the project. Lastly I will review the results of the algorithms compared the expectations from the literature review by comparing the method, data and results with a few selected papers.

Overall results

Table 5 - Overall results

	Binary classification	Multi-class classification
Majority baseline	50.2%	21.82%
Baseline (SVM)	**79.48%**	**63.64%**
Keras RNN (LSTM)	72.11%	56%

From the overall results in the table in above it is apparent that both the support vector machine and the neural network implementation manage to beat the set majority baseline regardless of whether it is a binary or a multiclass learning problem. When comparing the two learning methods, the support vector machine outperforms the neural network with around 4% and 12% for the binary and multiclass classification task respectively.

Support vector machine

As seen in table 5 the support vector machine achieved an accuracy of 79.48% on the binary classification task and a score of 63.64% on the multiclass task. The two tables below provide more detail to the performance of the learning algorithm by showing the performance on the individual classes. Precision, recall and F1-score are terms usually associated with information retrieval (Croft et al, 2009). Precision is a measure of the true positives in a class and is found using the following calculation:

$$Precision = \frac{True\ positive}{True\ positive\ +\ false\ positive}$$

In terms of terminology a true positive, is a data point, where the predicted target value is equal to the actual target value. Similarly a false positive is a data point which have been

incorrectly labelled as the given target value. Consider a machine looking for data points belonging to the class 'left'. If the machine finds a data point, where the true label is 'right', this would be a false positive. Comparatively a false negative would refer to a true label 'left' data point, which has been discarded and a true negative, would be a data point belonging to the class 'right', which correctly has not been labelled as 'left'.

To explain the function above further we can use the example again. Given the class with the label 'left', precision calculates how many of the files labeled 'left' are correctly labelled. Recall is a measure of the machines ability to find all the instances belong to a class. Continuing the previous example it provides a measure of how many of all the data points, which are supposed to be labelled as 'left', that have been labelled as so. Recall is calculated using the following function.

$$Recall = \frac{True\ positive}{True\ positive + false\ negative}$$

The F1-score is a function of precision and recall and is a measure which seeks to balance precision and recall measurements to a joined score. It is given by the following:

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Table 6 - Support Vector Machine for binary classification

	Precision	Recall	F1-score
Left	0.76	0.71	0.68
Right	0.82	0.81	0.82
Average	**0.80**	**0.79**	**0.79**

Assessing table 6 given the above definition, it appears that the algorithm is doing relatively well in its precision compared to its recall. This means that although the support vector machine has found a method, which enables it to correctly label true positives it is not comprehensive enough to find all true positives i.e. all speeches belonging to the left wing. This thereby create an increased amount of false negatives. This is most apparent in the label 'left', which in turn causes the overall score to decrease for the binary classification task.

Table 7 - Support Vector Machine for multiclass classification

	Precision	Recall	F1-score
DF	0.65	0.71	0.68
EL	0.64	0.77	0.70
KF	0.54	0.63	0.58
LA	0.50	0.64	0.56
RV	0.49	0.59	0.53
S	0.74	0.58	0.65
SF	0.59	0.57	0.58
V	0.72	0.61	0.66
Average	**0.65**	**0.64**	**0.64**

In table 7 the highest and lowest score for each measurement has been highlighted in blue and red respectively. The struggle observed in the binary classification task to left wing parties is also apparent in the multiclass task, where three out of four left wing parties have a recall of less than 0.6, and are the lowest scores on the board.

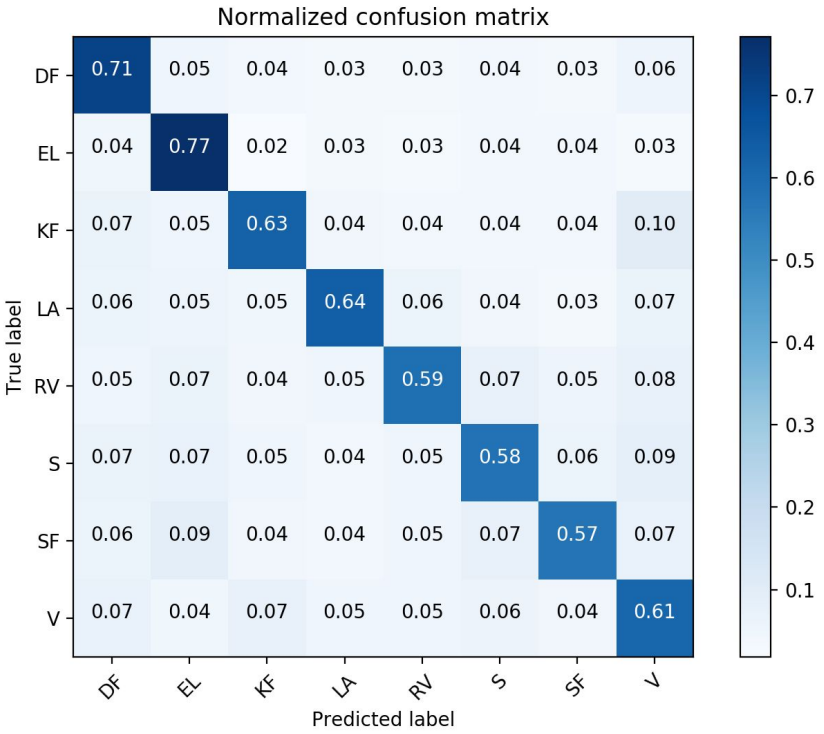
Interestingly S has the highest precision, but the second lowest recall of all the parties, meaning that the support vector machine may have traded off its recall ability to increase the precision. It is worth noting that the overall highest scorers are DF and EL. Neither of those two parties have been in government throughout the 11 year time period the data set covers.

In order to investigate the recall rates further, a confusion matrix have been computed. Furthermore, the numbers in the matrix have been normalised to appropriately compare the classes.

As V is the party with the highest distribution in the data set, there is an expectedly comparatively high amount of false positive V predictions in the data set. The highest of these is for KF, where 10% of the KF speeches were misclassified as being a speech made by V.

Seen as KF in six of the eleven covered years in the data set has been forming government with V, this is not entirely unexpected. Interestingly though, V has been falsely predicted to a larger degree with almost all other parties in the parliament other than EL regardless of whether the party officially is considered left or right wing.

Model 4 - Support vector machine confusion matrix



As mentioned above the three parties with the lowest recall values are RV, S and SF, where for all three of them, the support vector machine was only able to correctly label less than 60% of the or the possible correct data points. Considering that S is about as well represented in the data set as V, where both parties individually contribute with around a fifth of the dataset individually, the support vector machine does not have the same tendency to misclassify speeches as S compared to V as only RV, SF and V have their false negatives classified instances being more often classified as S compared to other parties. Furthermore, S is most often confused with V, DF and EL. This is odd seen as both V and DF in the described division between right and left wing parties should be considered right wing parties, where S would be a typical left wing party. The similarities between V and S might stem from that fact, that both these parties traditionally run for government with the intention of obtaining the prime minister title as well, which would cause them to moderate their statements as to appeal more broadly compared to the smaller parties, thus making them harder to distinguish. The low recall for RV and SF unlike S might originate from their low representation in the dataset although EL has a 20% larger distribution in the data set its recall i 35% higher. As with the general overview of the performance of the precision and recall, the confusion matrix show that the support vector machine is relatively good at classifying both DF and EL. That might be because the machine to a relatively large degree predicts both true positives and false negatives as DF and EL and thus raise the recall rate for the two categories, however as seen in table 7 the two parties precision values are also relatively high compared to the other parties.

Neural Network

On a general note the neural network was not able to perform as well as the support vector machine. Furthermore the training took several hours before the machine was able to make predictions. Therefore, the code for the neural network therefore is not as often or well-tested as the support vector machine.

Table 8 - Neural network binary task (full)

	Precision	Recall	F1-score
Left	0.68	0.67	0.68
Right	0.75	0.76	0.75
Average	**0.72**	**0.72**	**0.72**

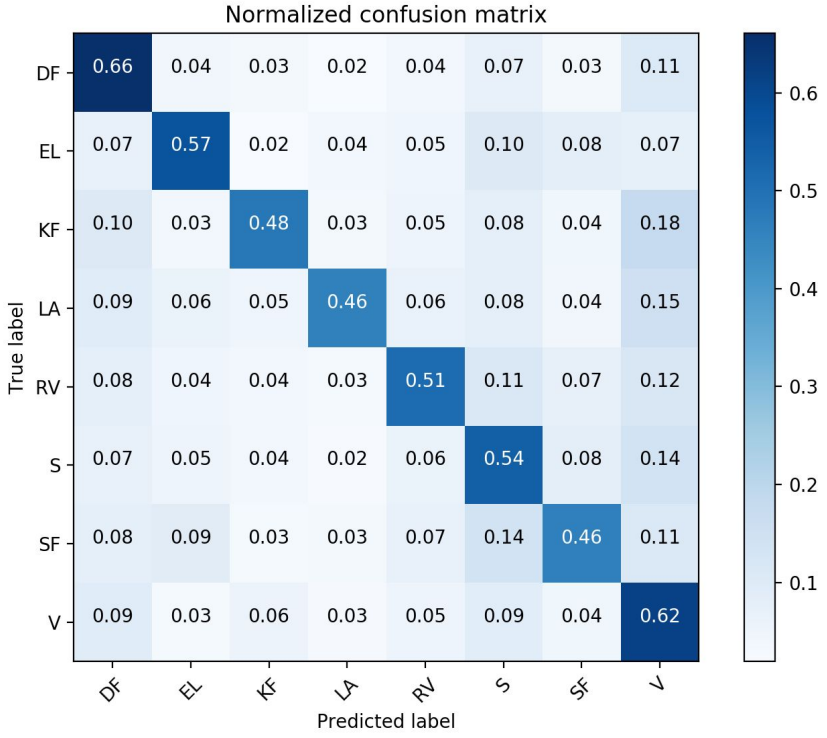
Like the support vector machine, the neural network also struggled more to recognize and thereby correctly classify speeches from the left wing parties than the right wing parties. Unlike the support vector machine, however, the precision and recall values are relatively similar within the same class. Considering the binary task in conjunction with the results for the multiclass task, it is clear that the target values with relatively low representation in the data set suffer in both precision and recall. As with the support vector machine report of the precision, recall and F1-score, the highest and lowest values have been marked in blue and red respectively below.

Table 9 - Neural network multiclass task (full)

	Precision	Recall	F1-score
DF	0.57	0.66	0.61
EL	0.63	0.57	0.60
KF	0.53	0.48	0.50
LA	0.52	0.46	0.49
RV	0.41	0.51	0.45
S	0.60	0.54	0.57
SF	0.46	0.46	0.46
V	0.60	0.62	0.61
Average	**0.56**	**0.56**	**0.56**

None of the four parties, EL, KF, LA and RV, who have less than a ten percent representation of the data set, have a precision, recall or F1-score above the average, and vice versa for the four parties with a high representation in the data set, apart from the recall of S. This may suggest that the neural network did not have sufficient data about the smaller parties to correctly classify them. There does not seem to be a pattern as to the value of the precision compared to the recall of the neural network. As with the support vector machine the confusion matrix may help investigate, which parties are often confused with each other and misclassified.

Model 5 - Neural network multiclass (full)



Looking at the distribution of predicted labels for the data set, it is clear that the neural network struggled a lot more than the support vector machine with much higher distributions of false negatives. As with the support vector machine V is often a predicted label, but in the case of the neural network the two other largely represented parties in the data set, S and DF, are also often predicted as the correct label for false negatives.

As mentioned above the parties with the lowest recall values are also the parties with the worst representation in the data set. You can however see a general tendency to be able to associate the speeches with parties belonging to the same political leaning as seen with SF, who to a large degree more often is classified as S rather than V or DF. This could explain why the neural network perform so much better on the binary classification task where it is only 4% worse in performance than the support vector machine compared to the 13% poorer performance in the multiclass classification. Although DF still has a high recall rate, the

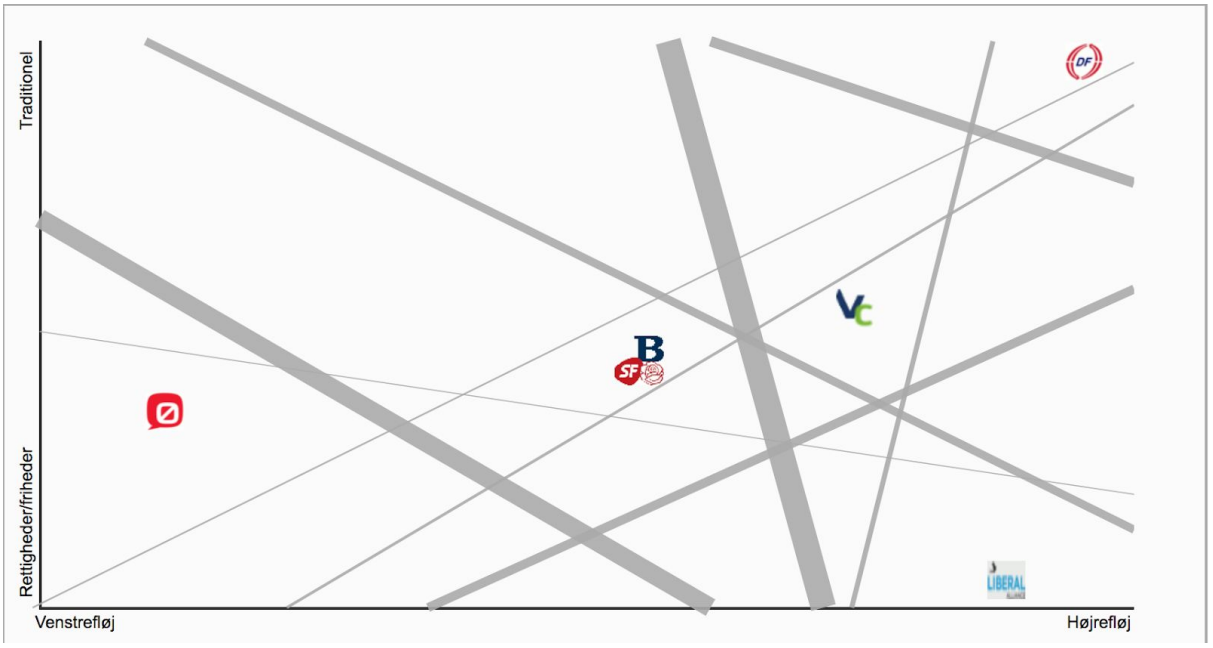
recall value for EL has fallen below that of S and V, which might stem from the machine often predicting false negatives to be to the three major parties causing the recall to increase.

Overall it seems that the neural network is more dependent on the distribution of instances in the data set compared to the support vector machine, when classifying texts. This is seen in the distribution of predictions on false negatives. One may wonder if this issue would lessen with a larger dataset, where the neural network would have more information to learn from, however, one may consider whether the added processing time is worth the potential increase in F1-score, when the much faster support vector machine already outperforms the machine on a smaller data set.

Distribution of the Danish parties

In the description of the data set I decided to split the dataset in two in order to do binary classification of the documents. In doing so I used the visualizations provided by Buhl & Rasmussen (Hvemstemmerhvad B, n.d.) of the parties positioned according to each other based on their voting history in the time period 2001-2011 combined with whom the party supported for prime minister. Buhl and Rasmussen expanded on their original one dimensional distribution of parties by expanding the original statistical model to add a second dimension to the distribution of parties using the voting statistics in 2011-2012 (Hvemstemmerhvad, C n.d).

Model 6 - Party position in two dimensions 2011-12 (hvemstemmerhvad C, n.d.)



In model 6 the parties are positioned according the the previously established left-right wing dimensionality as well as an added dimensionality between so-called tradition and rights/freedom. The grey lines represent votes held in the Danish Parliament, where parties

on the same side of a given line have voted together. The broadness of the line represent how the commonality of the division (hvemstemmerhvad C, n.d). Although the reason for whether or not the individual party vote in favor or against a given proposal may differ the representation above may help understand the accuracy difficulties of both the support vector machine and the neural network.

In both model 6 above and the one-dimensional model (Hvemstemmerhvad B, n.d.) the three left wing parties, S, SF and RV, appear closely together, which would suggest a high degree of voting agreement between the three parties. Comparatively the remaining left wing party, EL, is far further to the left of the other parties and separate of the other three parties. The S-SF-RV grouping upon inspection is actually closer to the two right wing parties V and KF than to EL.

Comparing this to the machine learning results above both the support vector machine and the neural network had relatively good score for recognizing and finding speeches from EL party members, while they struggled more to differentiate between the other three left wing parties. In the neural network implementation it is especially apparent that although the network is able to recognize and differentiate between left and right wing speeches at almost the same level as the support vector machine, it had performance problems in the multiclass task and among others struggled to distinguish between for instance SF and S.

Both learning algorithms showed a preference for V as the main classification label. Considering model 6 above this is not entirely unreasonable seen as the party is relatively close to four other parties in terms of voting habits across the wings, which may have confused the learning algorithms.

Following the argument that both learning algorithms perform well when classifying EL speeches, because EL in model 6 does not position itself closely to any of the other parties, it would be expected that the machine learning method would be equally successful in classifying speeches from either DF or LA, who both according to model 6 have positioned themselves in a unique position. It is the case for DF, which like EL has relatively high precision score for both the support vector machine and the neural network, however, LA has not. Although the support vector machine has a relatively good score for LA compared to the other parties, the neural network severely struggles in finding all the relevant LA documents, which results in LA having the lowest recall value of all the classes. A reason for this might be that LA has the lowest amount of instances of all the classes, as seen in table 2 from the data description, where V for instance has four times as many data points as LA does, making it harder for either machine learning method to make predictions for LA due to the lack of data.

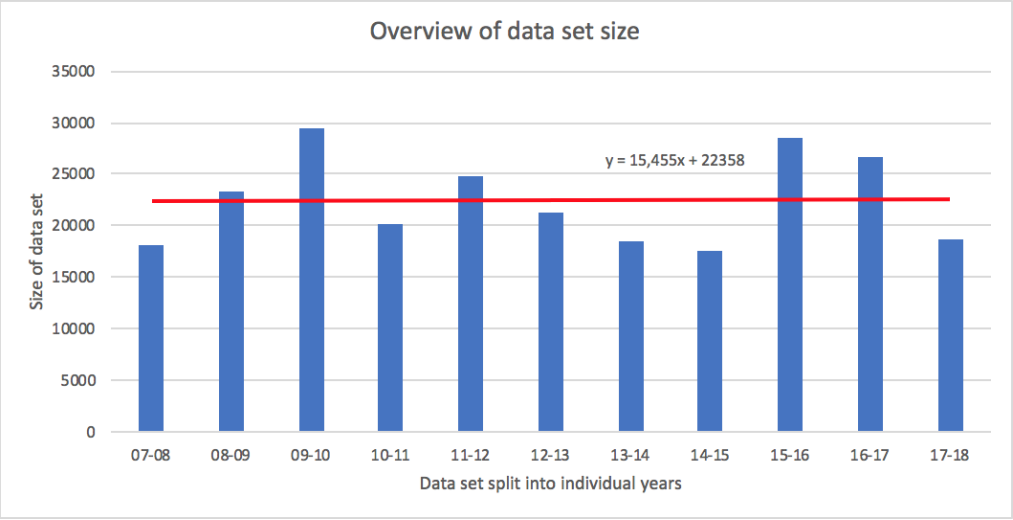
Temporal classification

Having reviewed the performance of the support vector machine and the neural network on the full data set, I wanted to investigate whether there was a temporal aspect to the performance. In my initial assumptions regarding the opinion expressed in each document in the dataset and its representativeness in regards to their target value, I chose to assume that the speeches in the dataset are all ideal representations of their respective parties. In this assumption there is a fault in that the speeches stem from an eleven year long period of time. During this time period the discourse within and between parties may have changed as well as the opinions held by the parties and the topics discussed in general. I therefore ran both algorithms again on each individual consecutive year to investigate the performance of the machine learning in relation to the individual years.

In doing so I expected the performance of the neural network to drop as the amount of data and information available for training severely decreased as the data set is split into eleven parts. In figure 3 below I have presented the size of the data set split according to the individual years. As can be seen there is some variation in the size of the individual data sets, however, the general trendline is relatively stable although slightly increasing. Each political year begins in October and ends around June.

Other than a general performance drop for the neural network I would not expect the performance to diverge excessively between the individual years given the original assumption regarding the representativeness of the full data set. Having tested an early implementation of the support vector machine on a smaller partition of the data set, I would not expect the smaller data set size to affect the performance. However, both the support vector machine and the neural network might benefit from the partition as the speeches might fit the assumption of the homogeneity within the different classes better.

Fig 3 - Data set distribution according to year



Multiclass classification

Below I will first discuss the results of the multiclass classification over time. Figure 4 below shows the accuracy of both the support vector machine and the neural network across the temporal partitions of the full data set. The immediate impression is that the support vector machine has not been negatively affected by the smaller dataset size as the average precision over time has actually increased to around 67%, while the neural network as expected overall have decreased in accuracy.

Fig 4 - Accuracy each year

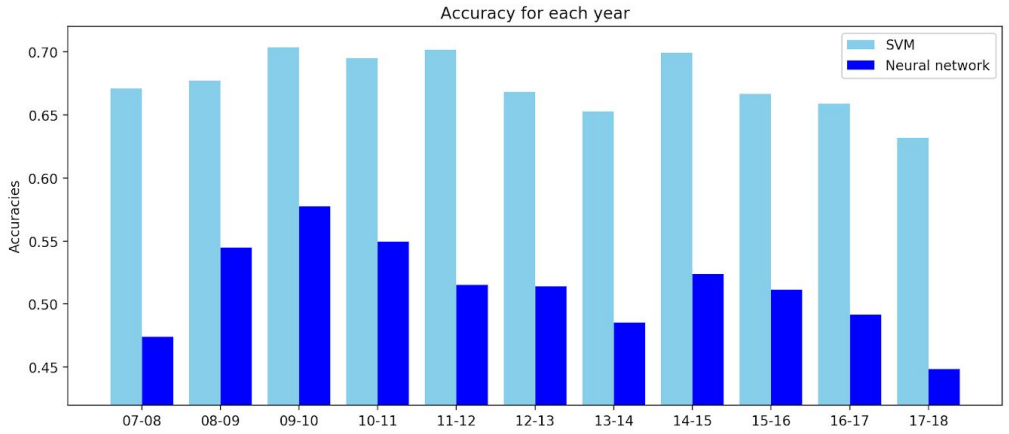


Figure 4 above shows the accuracy of both the support vector machine and the neural network across the temporal partitions of the full data set. The immediate impression is that the support vector machine has not been negatively affected by the smaller dataset size as the average precision over time has actually increased to around 67%, while the neural network as expected overall have decreased in accuracy.

In order to further investigate whether the data set size had affected the accuracies of the learning algorithms, the performances' correlation to the data set size were calculated by finding the Pearson correlation coefficient using Scipy (Jones et al, 2001). Unfortunately the p-values denoting the statistical significance of the results suggest that the current available information is not conclusive.

Table 10 - Correlation between performance and data set size

	Pearson's R	P-value
Support Vector Machine	0.24749	p=0.4446
Neural Network	0.49286	p=0.1235

Looking at table 10 it is worth mentioning that while neither result is statistically significant, the neural network both has the much lower p-value and higher correlation coefficient, which could suggest that the performance of the neural network is more sensitive to the size of the data set, than that of the support vector machine. Regardless of the correlation to the data set size the performance of the neural network is more volatile than that of the support vector machine across the data partitions seen as the range between the range between the highest and lowest accuracy is almost twice as big for the neural network as for the support vector machine.

Timeline

Having partitioned and tested the data set according to consecutive political years, I wanted to compare the performance across the years with a general timeline of the parliament since 2007. I therefore created a crude timeline of changes in government using the overview provided by Statsministeriet (n.d.). I chose to compare the results to government changes as I believe that elections and following change in positions of power, may have changed the discourse of the different parties as well as how strongly they have worded their opinions. In table 11 below the governments highlighted in grey have been formed following an election. The governments which have not been highlighted were formed based on internal changes rather than an election.

Table 11 - Timeline of government changes

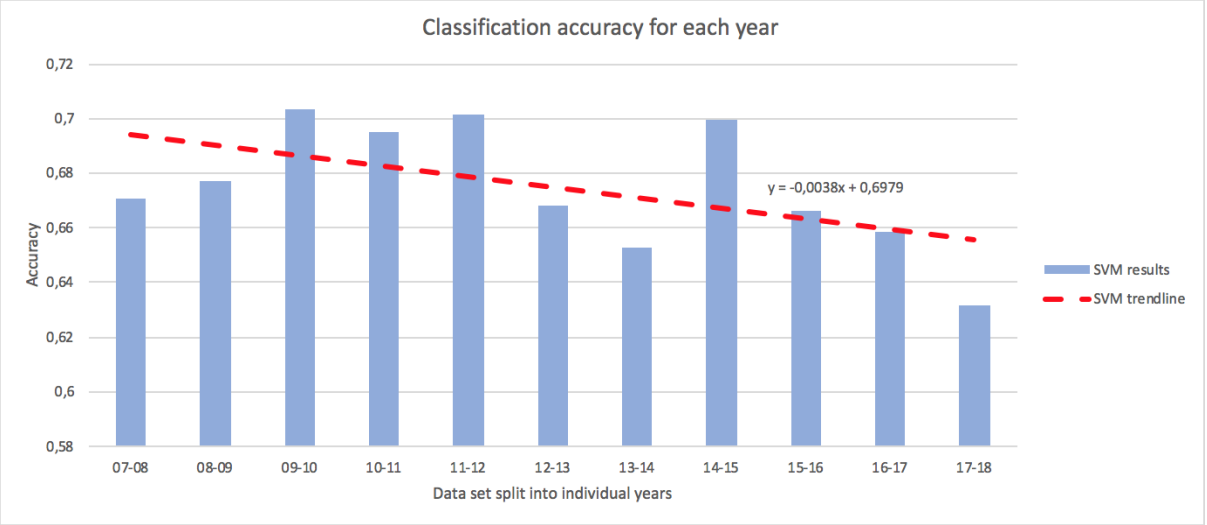
Government	Timeline
Anders Fogh Rasmussen (V-KF)	November ‘07 - April ‘09
Lars Løkke Rasmussen (V-KF)	April ‘09 - October ‘11
Helle Thorning-Schmidt (S-SF-RV)	October ‘11 - February ‘14
Helle Thorning-Schmidt (S-RV)	February ‘14 - June ‘15
Lars Løkke Rasmussen (V)	June ‘15 - November ‘16
Lars Løkke Rasmussen (V-KF-LA)	November ‘16 - Now

Seen as both the machine learning methods are relatively stable in the sense that the variation in performance in the most volatile method varies 12 percentage points from the best and worst performing year, I am looking at the individual performance and variation of performance of both methods separately.

For the support vector machine there are peaks coinciding with the aftermath of changing prime minister in 2009, The start of the S-SF-RV government in 2011-2012, and switching government to the V government in 2014-2015. This could possibly be because the parties need to position themselves more during election and therefore are more expressive in their

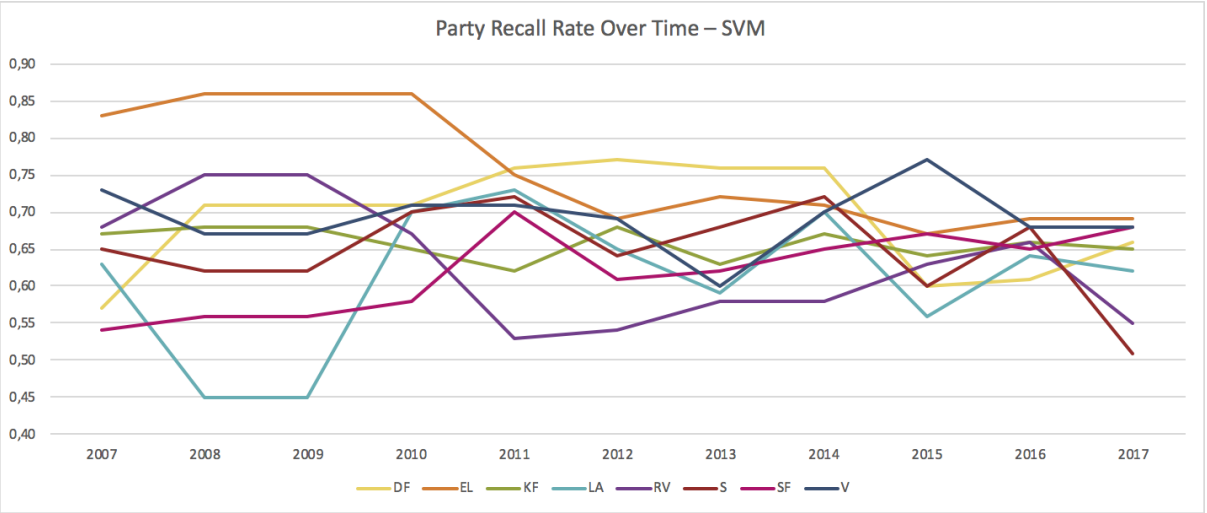
debates. In terms of the first peak following the new prime minister within the same governmental parties it might have been because the new government needed to establish itself and justify not having a new election when Anders Fogh Rasmussen left government to become secretary general of NATO (NATO, 2014). Interestingly, there is no peak when SF left government in 2013-2014 or when LA and KF joined V in a joined government in 2016-2017.

Table ZZ - Support vector performance across years



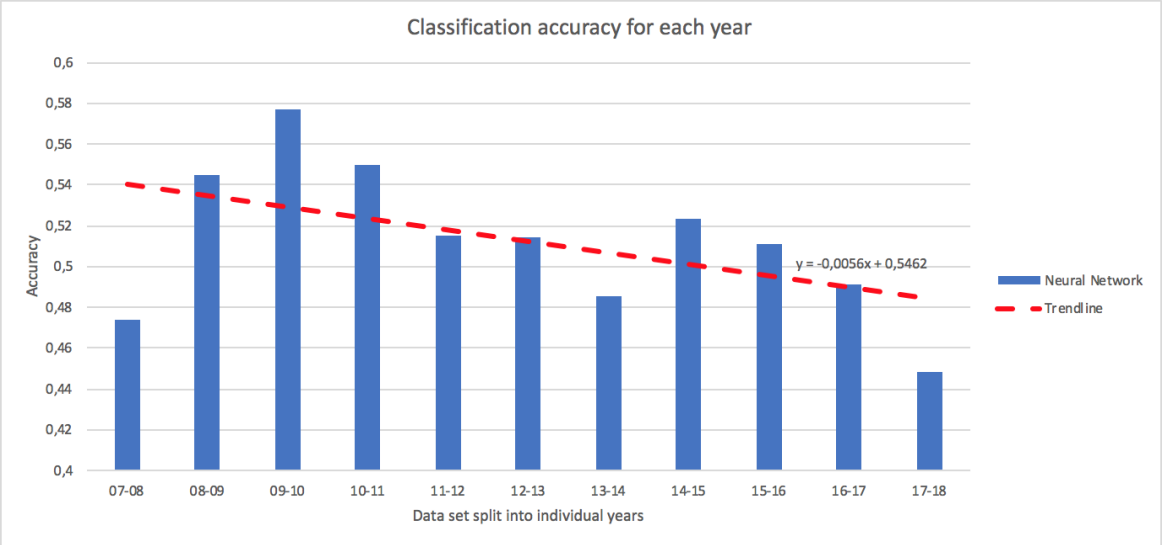
Considering the general trend of the performance, it is interesting that there seem to be a slight decrease over time of the performance of the machine. This joined with the year 2017-2018 being the worst performing year for the support vector machine may suggest that the parties have become more difficult to distinguish. In the appendix I have included an overview of the confusion matrices for each year, however, I have collected the recall value for each party over time in figure 5 below.

Fig 5 - Recall value over time



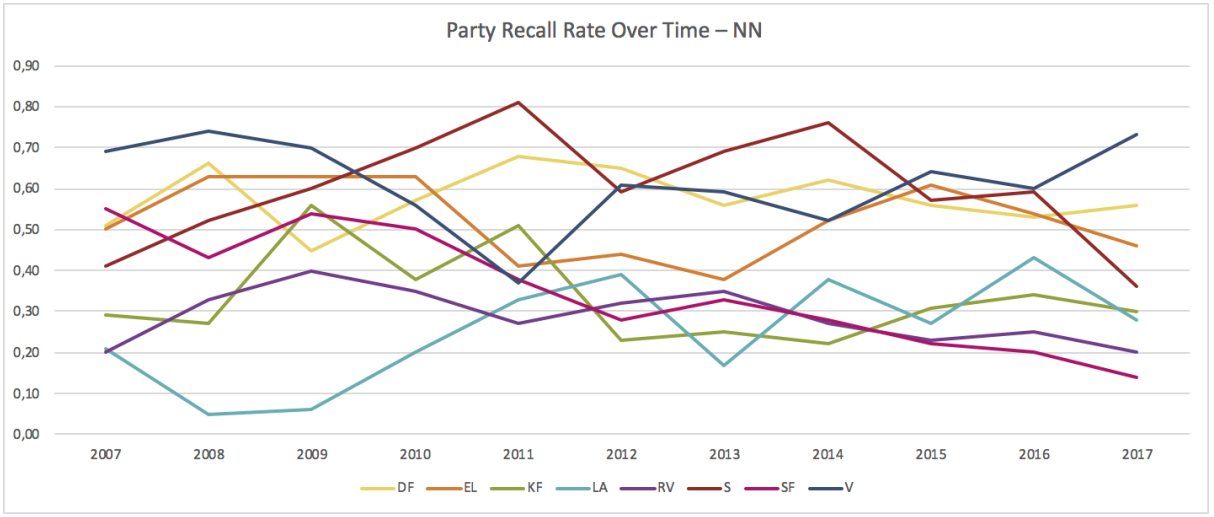
The confusion matrices collectively show the same tendencies as the confusion matrix of the performance when trained on the full data set. Some general takeaways from the model is that the parties DF and EL, which the SVM on the full data set was relatively better at recognizing, have become less defined over the years, with a general decrease in recall as can be seen in figure 5.

Fig 6 - Neural network performance across year



Like the support vector machine the results of the neural network over time shows a general decreasing trend from older years compared to current speeches. The neural network also has its lowest performance on accuracy in 2017-18. Although both learning methods are relatively stable, the decreasing trendline is also twice as steep for the neural network than for the support vector machine. The neural network also has a peak in performance in 2009-10 and 2014-15, but not when there was a change in government in 2011-12. Figure 7 show the recall for each party for each year using the neural network. The figure generally supports the above notion where the parties with the lowest representation in the dataset have the worst recall values while well represented parties likewise are performing comparatively better.

Fig 7 - Recall values over time



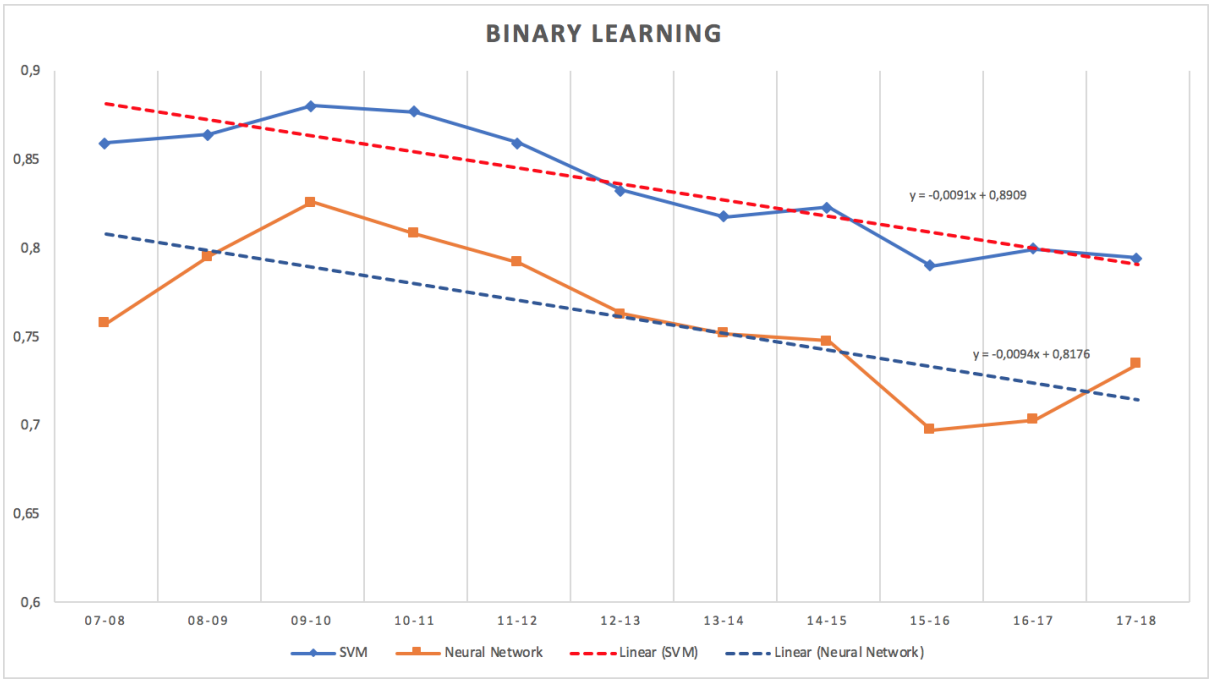
Binary classification

The effect of the dataset size on the neural network become even more apparent in the results for the binary classification over time. In the overall the results the performance of the support vector machine and the neural network are more similar in the binary task than when doing multiclass classification, however, as can be seen in figure 8 below, the difference in performance of the two have increased, when the data set was partitioned. Interestingly both methods average result of all eleven iterations improved compared to the overall score on the full data set, which might suggest some degree of temporality or other features in the data, which makes the classification on the overall data set harder.

In this regard it is worthwhile noting that neither of the performances below seem affected by the timeline of major events presented in table 11, other than 09-10 and for the neural network 17-18.

Both models are surprisingly similar in their shape in respect to rise and falls in performance. Like for the multiclass classification both methods have a declining trendline, which for the binary test is slightly steeper in both cases. Combining the declining trendline with the lack of connection with the above timeline might suggest, that the parties have become harder to distinguish in general over time. This would to some extent coincide with the analysis by Buhl & Rasmussen, who in their visualization of party movement over time (Hvemstemmerhvad B, n.d.) also find that some parties in terms of their voting patterns have become more similar over time.

Fig 8 - binary classification over time



Literature comparison

Having finished the initial classification task I will now compare my results to the results of the literature in the literature review. Because the literature review covered several articles with different scopes and intention of investigation I have chosen three articles to compare my method and results with. The three selected articles are representatives of the overall literature review and represent different aspects of the project.

The majority of the papers in the literature review that investigated political affiliation used support vector machine implementations combined with bag of word vector representations of their data in a binary classification environment. Diermeier et al (2011) and Yu et al (2008) both perform document classification on speeches from the American Senate and are common references in the assessed papers. The two articles overlap both in data set, methodology and contributing authors, but they differ in scope of the research. Yu et al (2008) investigate the possibility of creating an ideology classifier and how generalizable this would be between the House of Representatives and the Senate, where Diermeier et al (2011) expands on Yu et al (2008) to investigate voting behaviour and topics dividing the senators. As the scope of Yu et al (2008) is closer to my research question I will discuss their results in comparison to mine.

Yu et al (2008): Classifying Party Affiliation from Political Speech

Yu et al (2008) analyze congressional speeches from the American Senate and House of Representatives. In this paper they classify speeches by using speeches from the Senate as the training data and the House as testing data and reversed with the intention of both investigating whether it is possible to classify speeches based on political affiliation and whether the learning algorithm can be generalized to perform well on a different data set.

All speeches from the same individual was aggregated into one long document, thus creating a smaller data set but with longer and more informative documents as data points. The documents were afterwards transformed into a bag of words representation, where the most rare and common words were removed. This meant that words, which either appeared less than three or more than fifty times across the data set were removed. For the classification Yu et al (2008) implement both a Naive Bayes classifier as well as a support vector machine but for the sake of comparison I will only be addressing the performance of the support vector machine. Yu et al (2008) test the support vector machine under three different conditions:

- Simple bag of words noting the presence of words (bool)
- Bag of words representation with term frequency weights (tf)
- Bag of words representation with term frequency and inverse doc frequency (tf-idf)

The baseline was set by simple majority representation in the data set. The support vector machine was first tested using speeches from the House of Representatives as training data and the Senate speeches for testing and afterwards the datasets were switched.

Table 12 accuracy comparison

	Train: House, Test: Senate	Train: Senate, Test: House
Baseline	51.5%	55.0%
SVM (bool)	75.1%	73.7%
SVM (tf)	69.8%	55.6%
SVM (tf-idf)	80.1%	69.7%
My baseline (binary)	50.2%	
My SVM (binary)	79.5%	

In the immediate comparison it seems that my support vector machine is performing as well as the implementation of Yu et al (2008), however, it should be kept in mind that my training and testing data is from the same domain and thus the machine is less prone to lose accuracy on the test set. The potential sensitivity of the performance related to choice of domain for training and testing is also apparent in the results above. This can be seen in the range of performance between the two testing environments based on what is used as the training and testing data respectively.

Lastly, Yu et al (2008) investigated the potential time dependency of the performance by testing their support vector machine on consecutive years from 1998 through 2006 and checked the performance. The final graph showed some variance in performance but a general improvement in performance over time. The reason from this may have several origins, seen as the internal variance in topics may have affected the results, but Yu et al (2008) does suggest that partisanship within the Senate may have increased over time.

I also wanted to assess the multiclass classification results but few of the selected papers in the literature review had done multiclass analysis for political affiliation (Dahllöf, 2012; Biessmann et al, 2016; Søyland & Lapponi, 2017). Although Dahllöf (2012) used a data set consisting of speeches from the Swedish parliament, which, like the Danish Parliament, consists of multiple parties, the paper had a general focus on author recognition rather than document classification and sought to combine the political aspect in a binary left-right metric with the gender and age of the speaker in question. Both Biessmann et al (2016) and Søyland & Lapponi (2017) work in a multiclass setting, but of the two, Søyland et al (2017) proved more relevant seen as it is based on similar data and implement a support vector machine. Comparatively Biessmann et al (2016) have a smaller number of classes and

implement a different learning algorithm - namely multinomial logistic regression. Furthermore, Biessmann et al (2016) use a different type of data set, where they combine shorter texts to create longer documents for the classification task.

Søyland & Lapponi (2017): Party Polarization and Parliamentary Speech

Søyland & Lapponi (2017) investigate party polarization using support vector machines. They seek to investigate how the machine performs in a multiclass setting rather than a binary one and furthermore how different preprocessing methods affect performance. Their data set consists of speeches from the Norwegian Parliament in the time period 1998-2016 from the corpus Talk of Norway. This provided a data set consisting of 250.373 speeches spread out in 7 different parties. The speeches were extensively pre annotated with linguistic information including lemmas and part-of-speech tags. The corpus furthermore contained 99 extra variables with speaker characteristics, party affiliations and date.

Like Yu et al (2008) Søyland & Lapponi (2017) test their support vector machine with different methods of preprocessing.

- Tokenization and lemmatized unigrams
- Part of speech tagging
- N-grams
- Meta data

In the last implementation Søyland & Lapponi (2017) is including meta data such as gender, type of debate, keywords and name of committee leading the debate. Figure 9 below is their representation of their results.

Fig 9 - Representation of results (Søyland & Lapponi 2017)

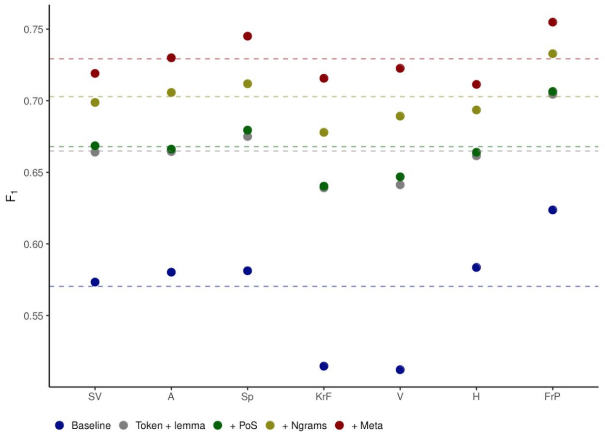


Figure 1. F_1 scores on the full dataset. Points represent the F_1 score for the parties on the x-axis and the dashed lines the macro F_1 scores for all models. The models are differentiated according to the color of the points and lines.

Table 13 - My results

	Accuracy
My baseline	21.8%
My SVM (multiclass)	63.6%

In the case of multiclass classification my support vector machine is not performing as well as even the lowest level of preprocessing of Søyland & Lapponi (2017). However, it would seem that effect of lemmatization improve performance a lot. Interestingly part-of-speech tagging seem to have little effect on performance compared to the addition of n-grams and meta data.

Søyland & Lapponi (2017) lastly comment that their result show little sign of polarization in its classification, as the support vector machine showed no preference for classifying false negatives as politically similar parties rather than parties of a different political orientation. Søyland & Lapponi (2017) suggest that this might mean that discussing polarization in multiclass environments with a lot of agreement as in the Norwegian Parliament does not make sense.

Considering model 4 and 5 above with the confusion matrices for both the support vector machine and neural network, there is little difference in the number of misclassifications, but still a slight tendency to still classify within the same wing of political affiliation. This is especially the case of the neural network implementation as seen in model 5. Furthermore, combining this with the positioning visualization presented in model 6, it suggest that there is a polarisation in the Danish parliament speeches.

Lastly to compare the neural network results with current research I am comparing the results to the paper by Iyyer et al (2014). None of the research covered, performed political multiclass classification using neural network, so there will only be a comparison with the results in the binary classification tasks.

Iyyer et al (2014): Political Ideology Detection using Recursive Neural Networks

Iyyer et al (2014) conducts a binary investigation of the political affiliation of speeches and ideological books using a recursive neural network. A recursive neural network compared to a recurrent neural network, does not have the same sequential aspect of time and is as such a generalized recurrent network.

The data set is split in two. One part consists of transcripts from US congressional floor debate in 2005. The transcripts have since been filtered to leave only explicitly biased

sentences, which provided 7.816 sentences for training. The second part consists of ideological books which likewise have been filtered, which provided 55.932 opinionated sentences.

The research has set two main baselines: Random guess at 50%, and a logistic regression model which is either tested with a bag of word representation of the data or on pre trained word embeddings. The logistic regression has been included as a baseline seen as Iyyer et al (2014) wants to compare the performance of the recursive neural network, where sentence structure is retained with the performance of a classifier with the bag of words representation.

The classification have been implemented with three versions of the neural network:

- Neural network, where all parameters are randomly initiated
- Neural network with pre trained word embeddings
- Neural network with pre trained word embeddings and annotated phrase labels

Table 14 - Comparison of results

	Debate transcripts	Ideological books
Baseline (random)	50.0%	50.0%
Baseline (log_reg_BoW)	64.7%	62.1%
Baseline (log_reg_embed)	66.6%	63.7%
RNN (random)	69.4%	66.2%
RNN (embeddings)	70.2%	67.1%
RNN (embeddings+phrase)	-	69.3%
My baseline	50.2%	
My RNN (binary)	72.1%	

My implementation of the recurrent neural network is at the same level as that of Iyyer et al (2014), however, my data set is far larger than that of Iyyer et al (2014). Rather than doing document classification, Iyyer et al (2014) performed sentence level analysis. If the Danish dataset was split into individual sentences it would consist of 2.659.735 sentences, which is far larger than the largest data set based on books, which consists of 55.932 sentences. Having noted how neural networks tend to perform better with more data, I would have expected the recurrent neural network based on the Danish data set to perform comparatively a lot better than what is the case.

Sentence sentiment classification is different from regular document sentiment classification, however, seen as the average document length in the Danish data set is 210 words, which is comparatively shorter than the documents described in most of the other literature in the literature review, I still believe the results presented by Iyyer et al (2014) are comparable to some extent.

Sources of error

For this investigation there are some sources of error, which need to be kept in mind when reviewing the results. I have split the main sources of error in to three types: Basic assumptions, preprocessing and methods and how representative the data is.

Assumptions

In the project I have made some basic assumptions regarding the nature of the data set and the task to justify doing sentiment analysis.

1. One document represents one opinion as is from one opinion holder
2. The data points within a class are homogenous
3. The classes are separable

Regarding the first point each document represents one speech from one speaker. Thus the data is true to half of the first assumption. Regarding whether each document represents a single opinion is not always the case. The average document length is 210 words, and most speeches are part of a continuous discussion, where a single topic is being discussed, however, some of especially the longer documents represents a longer speech held by one politician, who tries to respond to several previous statements. As a result longer documents may be problematic for the learning method as it may be discussing several topics simultaneously.

In case of the homogeneity of the documents and thus data points within a single class, I have chosen to assume that each document is an ideal representation of the assigned target value. This means that any given document from e.g. DF would be a representative opinion from DF as a whole. However, as was just seen in the data set partition above the performance would change depending on what year the machine learning method was analyzing, which suggests that there may be a temporal difference in the voiced opinions. Furthermore, each party consists of several different speakers, who although they agree on a general direction and opinion might have slight differences in opinion and how they express said opinion. Furthermore some parties consist of a relatively high number of members and are changing members during the eleven year period more often than the smaller parties, making the data less clear.

In the reviewed literature investigating political tendency the papers are to a large degree investigating binary distributions of data, where the parties are assumed separable into two groups. In this project there are first and foremost eight parties represented. Although the Danish political landscape does have a left and right wing (Larsen, 2015), the wings consist of various individual parties making the internal variation of opinions within the block bigger than what is arguably the case in a pure two party system. Furthermore in terms of the separability of the individual parties their individual voting history show, that despite the positioning visualized in model 6 based on opposing votes and koalition, the data shows that more than 70% of all votes in Parliament passes based on broad coalitions across the two wings (hvemstemmerhvad D, n.d.), and parties cross the different political scales vote in agreement with each other at least 40% of the time (hvemstemmerhvad E, n.d.).

Preprocessing and methods

The literature review and the direct comparison of the results with the three selected papers highlighted the effect the choice of preprocessing of the dataset may have on the overall performance of the classifier. Much of the reviewed literature for this project either discuss or investigate different levels of preprocessing or annotation of the data prior to the classification. Furthermore, Søyland & Lapponi (2017) clearly visualize how the support vector machine benefited from having the data being preprocessed and annotated.

In this project the data was only preprocessed to a limited extent and documents were only annotated with the related target value. This was in part to keep the coding simple and able to manage the classification within a reasonable timeframe and in part due to the previously discussed issues with the state of the currently available linguistic tools of analysis for Danish.

Should this project be extended or improved in the future, the results and available research strongly suggest that the data would improve from more preprocessing such as lemmatization and possible annotation with background variables about the speaker considering how meta data improved the results of Søyland & Lapponi (2017). In relation to this the implemented support vector machine was purely implemented with a combination of unigrams and bigrams, and the performance might benefit from expanding the size of the n-gram.

Representative Data

The level of representativeness of the data is a matter of how generalizable the model is to be inferred onto other domains and types of data. This is especially interesting for sentiment analysis, which is sensitive to domain change and often runs the risk of overfitting to its

training data as opinions and sentiment and how it is expressed is highly subjective and may change across domains (Liu, 2017).

The data used for this project consists speeches from the Danish Parliament. Although it is transcribed speeches the language usage in Parliament is very formal as the members for instance are not allowed to address each other directly, and must refer to each other in third person only (Folketinget, n.d.). Furthermore, as all the speakers are elected politicians, their general language use and style of voicing their opinion might not be representative of normal language use outside the Parliament.

Testing on different data

To this point the models have only been tested by splitting the data set into a training and test set in a 90/10 percent split. This means that the models have been tested under the best circumstances in terms of performance, in the sense that they are tested on data instances that are very similar to the training data both in terms of language use and format. In terms of assessing the models' general applicability to other types of texts, they would have to be tested on another data set collected from a different domain than the speeches in the Danish parliament.

I decided to collect tweets and updates from Facebook to create a new test set. I specifically wanted to collect tweets and updates from the politicians in the Danish Parliament. This was to use the same assumption as with the general data set in terms of political affiliation and thereby save time from annotating and assessing each individual data point. Furthermore such tweets and updates would presumably also mainly contain actual opinionated texts rather than general updates seen as such official profiles would be used by politicians to update their potential voters on their opinions. The new data set would still differ from the general as the language usage on both Twitter and Facebook is quite different from the formal language required in the Parliamentary speeches. Lastly, the official profiles are used to inform and engage with the general public rather than debate with other politicians and the language usage would presumably also thereby be different and less formal.

For this task I had initially planned to use an API to collect tweets and updates from the current Danish Parliament politician's official Twitter profiles and Facebook pages. Due to recent restrictions to both the Twitter and the Facebook API following the scandal surrounding the Cambridge Analytica data breach (Perez, 2018) this has not been possible. I have instead found two smaller datasets collected from Twitter and Facebook, which I will describe below before going through the testing and performance of the datasets.

Twitter dataset

The dataset consists of 1.569 tweets from Danish politicians on twitter and was originally collected two years ago using the Twitter API for another assignment in 2016. The data set is neither large or diverse enough to justify dividing the tweets according to party, but they are however, almost equal in distribution according the the earlier defined division between a left and right wing, and can therefore be used as a test set for the binary version of both machine learning methods.

Table 15 - Distribution of instances

	Left wing tweets	Right wing tweets
Documents	827	736
Percentage	52.9%	47.1%

Table 15 shows the distribution of tweets in the Twitter data set. This distribution is slightly skewed in favor of the left wing parties. There is a higher number of different contributors to the right wing tweets (see appendix), which could cause more internal variation in that class compared to the left wing tweets. Lastly the format of the data points are quite different as can be seen in the two examples below:

Tweet from twitter dataset:
RT @j_esmann: BLOKERET! Dansk Folkepartis folk giver ikke plads til kritik på sociale medier. #dobbeltmoral https://t.co/bbsiwMPpnJ #dkpol

Speech from original dataset:
Hvad er grunden til, at ministeren i den selv samme uge, hvor man bryster sig af genopfundne ambitioner for klimaet, ønsker at fjerne de øremærkede pso-midler på i alt 2,5 mia. kr., som ellers var prioriteret den grønne omstilling, og i stedet regne dem ind i det generelle råderum?

As can be seen from the two representatives of either dataset, the tweets have a large degree of twitter specific features, which are not present in the training set based on parliamentary speeches, which could harm the performance of both models.

Facebook dataset

The data set was collected manually from Facebook in September 2018 by going to the official pages of the Danish parties and politicians and save their most recent updates to a local file. There was only collected a small data set from Facebook due to the time

consuming nature of the collection process. For each of the eight previously mentioned parties in the Danish Parliament ten randomly selected members of parliament were chosen. The ten most recent updates from the official pages of those ten politicians were then collected. Lastly the twenty most recent updates from the official page of the party as a whole was also collected.

Two of the eight parties had less than ten members in parliament. In those cases the lacking amounts of updates were collected from the official page e.g. SF has only seven members in parliament, so 50 update were collected from the official SF page to ensure that the SF part of the data set would be as big ass the other parties'. Any smileys in the updates were removed to avoid possible encoding errors.

This process provided a data set consisting of 960 instances perfectly divided between the eight parties. The updates from Facebook do not have the same extent of terms specifically used on facebook, and would therefore presumably have higher accuracy scores than for the Twitter dataset. The language in the updates a much less formal than that of the parliament speeches, and I would therefore still expect a general decrease in performance from both models.

Coding

The original python code for both models was slightly altered to incorporate the two data sets (see OOD-SVM-py, OOD-RNN.py). After loading the full data set, the two new data sets are also loaded and saved as variables.

Example from OOD-SVM.py

```
#Preparing the data
print('\n\nloading dataset...')
folders = glob.glob('binary_tale')

for f in folders:
    print(f)
    Pol_data = load_files.load_files(f,ignore_files='.DS_Store',
                                     encoding='utf-8', shuffle=True)

#loading ood test data
tweets = glob.glob('politikere')
for t in tweets:
    print(t)
    test_data = load_files.load_files(t,ignore_files='.DS_Store',
                                      encoding='utf-8', shuffle=True)

fb_post = glob.glob('fb_tale_binary')
for p in fb_post:
    print(p)
    fb_test_data = load_files.load_files(p,ignore_files='.DS_Store',
```

```
                                        encoding='utf-8', shuffle=True)

#Preparing training dataset
X_train = Pol_data.data
y_train = Pol_data.target
#preparing test set
X_test = test_data.data
y_test =test_data.target
X_test_2 = fb_test_data.data
y_test_2 = fb_test_data.target
```

The main change to the code in both cases is, that rather than splitting the original dataset, the full set is used for training and saved in the training variables before the preprocessing. Likewise the two new data sets are loaded fully as test set 1 and 2 before processing. By the end of the code the model evaluates both test sets for accuracy and all results are printed independently of each other.

Results

Both the twitter and the facebook dataset was used for the binary testing using simple majority for the baseline in both datasets. Seen as the twitter data set couldn't be split into eight reasonably sized classes, only the facebook data set was used for the multiclass testing with a majority baseline of 12.5%.

For the binary results neither machine learning method managed to beat the baseline for the twitter data set although both did manage to perform on a similar level as the set baseline. In the Facebook dataset both methods did manage to beat the baseline suggesting some degree of general applicability for the methods to at least Facebook updates. The performance is however still lower than for the original test, where the full data set was split 90/10. The methods would therefore have to be expanded to better handle unknown texts. All testing exhibit the same pattern as in the general testing, where the support vector machine slightly outperforms the neural network.

Figure 10 - binary test

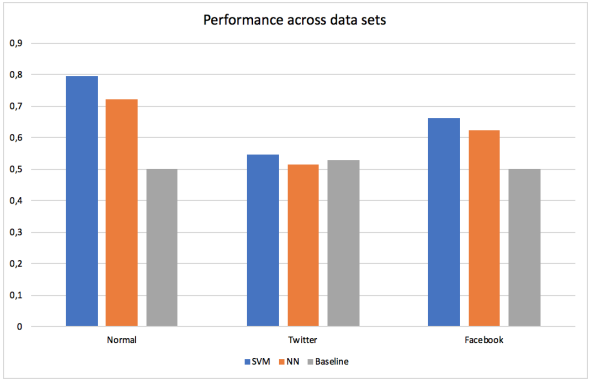
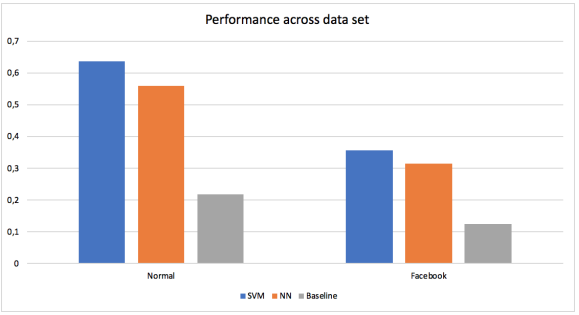
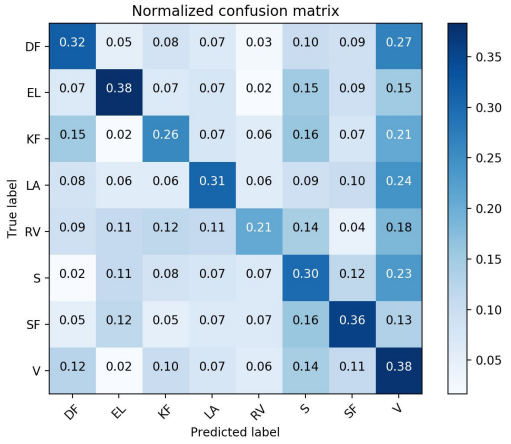


Figure 11 - multiclass test

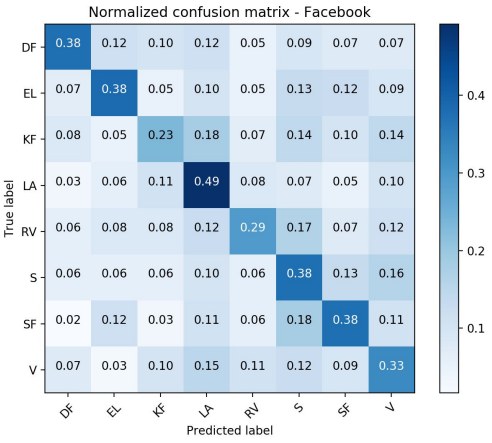


The multiclass test results show the same patterns as the binary test for the Facebook test set, but the overall performance has decreased more. The baseline has also decrease though, which might have affected the performance. Although both methods beat the baseline, their precision overall is quite low, with neither method able to get more than 40% accuracy on the multiclass task.

Model 7 - Neural network



Model 8 - Support vector machine



Considering the confusion matrices of both models for the multiclass classification task the neural network's preference for the three large parties in the training data i.e. V, S and DF, is clear in its predicted labels. Interestingly both models are perform comparatively well in predicting SF data points considering the results in the initial testing. The reason for that might be that almost half of the SF data points are from their official facebook page, rather than several individuals, making their posts much more homogenous. The support vector machine performs surprisingly well, when classifying LA. Furthermore, the close relation between RV, S and SF shows in the confusion matrix. Both methods have problems classifying KF, whom for both models most often is confused with other right wing parties as well as S.

Keeping in mind that both machine learning methods have a relatively simplistic setup, both methods still managed to beat the baseline for the Facebook dataset. This suggest some degree of generalizability of the models given more specifications or preprocessing of the data might increase.

Broader Perspectives

This section discusses the potential use case of automated detection of political bias in text, should the model be developed and generalized further.

Microtargeting

Our world is becoming increasingly digital. According to the 2018 Global Digital Suite report there are now around 4 billion internet users and 3 billion social media users worldwide amounting to 53% and 42% of the total world population respectively (wearesocial, 2018). This creates a new digital environment, where people across the globe interact with each other through social media and other online platforms. The World Economic Forum estimates in a report addressing digital and social media, that people on average spend three hours a day using social media and the related platforms (World Economic Forum, 2016). According to Domo (2018) the data usage per person on earth today amount to up to 1.7 MB data per minute.

When people become more present online it is to be expected that companies and organizations would have an interest in also digitizing and thereby reach out to their potential customers, partners and / or stakeholders. An example of this is the rise of online marketing and within this the usage of online political marketing by for instance NGOs, political campaigns and politicians. An increasingly common feature within online marketing is the use of microtargeting, which is the act of creating specifically target messages for a narrow category of users based on data analysis, which may often include the individual's demographic information and online habits (Gorton, 2016).

Microtargeting can in political marketing be used to identify voters of a certain political affiliation and target the marketing to sway their votes by for instance select or emphasize political stances most likely to match the targeted voter. As the technologies have grown more sophisticated, microtargeting have become common, and the focus have switched from wanting a broad reach to everyone, to getting targeted messages to the right people (Borgesius et al, 2018).

As the microtargeting algorithms often utilize background data such as information regarding the individual's age and gender as well as their behaviour on a given platform, an automated detection of political affiliation could become a powerful tool within microtargeting. A fully developed general model would be able to analyze and use the individual users' written interactions and statements on a given platform rather than their background variables and otherwise online behaviour such as likes and clicking history to classify and thereby target potential voters.

Bias detector

A digitized environment with an increasing amount of internet penetration and social media presence foster the online communities such as interest based online groups and forums.

While the online communities and digital media provide ample opportunities for causes and networks to spread information and organise, the more direct path between content creators and consumers also allows for a large amount unsubstantiated and disintermediated information to circulate (Vicario et al, 2016). Furthermore, recent research show that people are consuming more news online and on online platforms - especially among the younger generations (Wandsøe-Isaksen et al, 2018; Mitchell et al, 2016; Nielsen, 2016).

The communications firm Edelman has for the past 18 years annually surveyed and assessed the trust in people and institutions on a global scale. In their most recent report, Edelman reported a general rise in the distrust to social media companies and their capability to control fake news and polarised content in part due to divisive ads, twitter bots and fake news on their platforms (Edelman, 2018). More than half of the survey participants believe, it is hard to distinguish between good and bad journalism, and Edelman (2018) found an increased scepticism towards news organisations and their ability to remain neutral and balance good journalism with commercial gain. Despite this, Edelman (2018) discovered that even though nearly half of the participants state, that they do not trust the online platforms, they rely on them for their news, with more than 60% saying they receive all news from online platforms.

Both local government and online platforms try to combat the distrust in the media and online information. US government is for instance considering legislating against partisan ads (Newman, 2018), the french parliament legislate against fake news (Morin et al, 2018) and both Facebook and Google integrating fact checking into their algorithms (Newman, 2018; Gartenberg, 2018). But, as Newman (2018) states in a Reuters report, these efforts seem to have a limited effect on the amount and reach of partisan content. The individual users, therefore, still need to critically assess, whether the available online information from websites, blogs, microblogs and other online user interaction is valid and trustworthy.

A recent example of extreme content and reactive power of the major platforms, is the case of the founder of 'Infowars', Alex Jones, who was recently removed from Apple's Itunes Store, and shortly after from Facebook, Youtube and Spotify for encouraging hate speech and bullying (Hern, 2018). This among other things resulted in a discussion of the role of online platforms and their responsibility to monitor and screen their content, and whether actions such as banning Jones makes them partisan themselves (Wong & Solon, 2018). Ingrained in this discussion lies the question of whether or not online platforms have a responsibility as a publisher of the online content, or if they are neutral platforms, responsible of distributing the content rather than monitoring it (Bauckhage, 2014). When the platforms integrate fact checking mechanics in their algorithms, which in turn enable the users on the platform to decide for themselves whether content is reliable, rather than screening content, they retain their status as being a neutral platform. Comparatively by banning Jones the companies, who run the online platforms, become more actively involved in what kind of content is being distributed and thereby take on more a role of a publisher.

An alternative way of handling partisan content, which would enable online platforms to retain their neutral platform status for content, could be to employ mechanics similar to the fact checking machines of Facebook and Google, but for detecting for instance hate speech or political bias. Rather than becoming an active monitor and screener of content on the platform this mechanic should instead inform the users of the possible nature or bias of the content, they are experiencing. This would improve the individual's possibility of critically assessing the available online content.

The code itself rather than being integrated into existing algorithm could also be reworked into a detector itself, which could be fed with texts to be analyzed. As an example of this I reworked the code for the support vector machine so it could be interacted with in the command line or terminal.

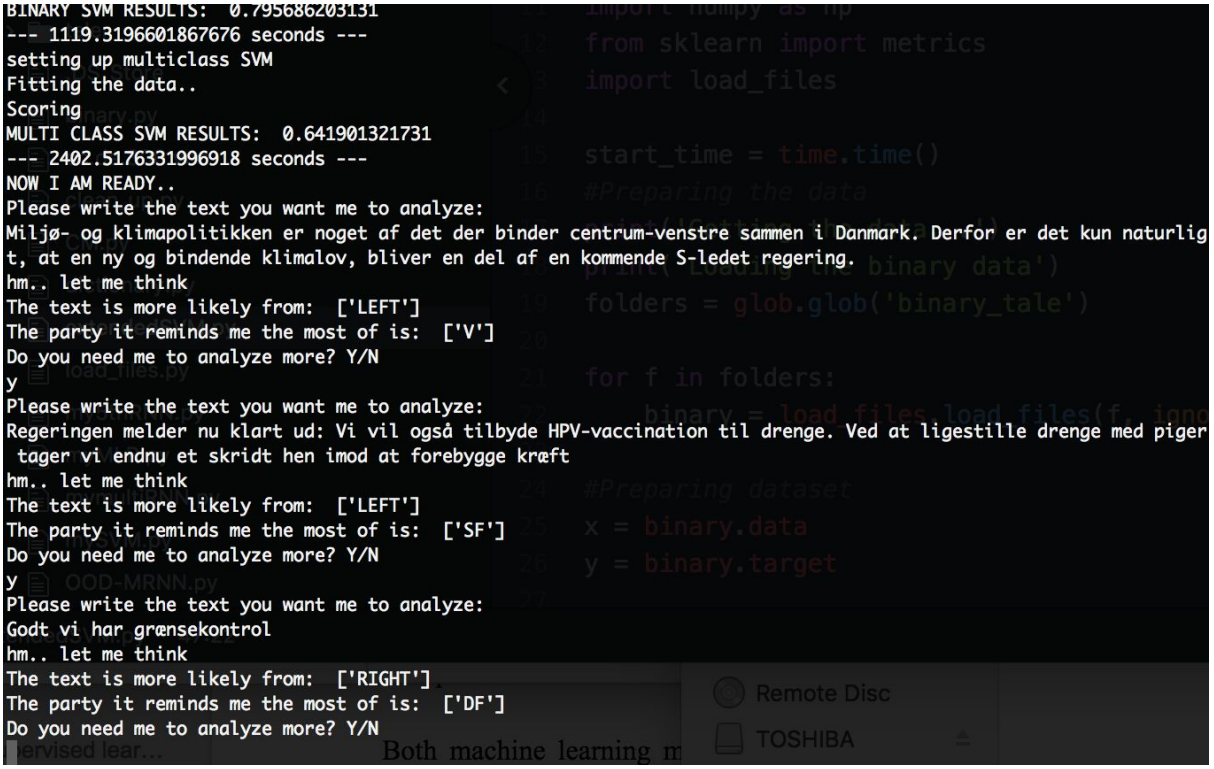
The code itself works as the regular mySVM.py, however after fitting the classifier to the data sets, I added some lines of code, which allowed me to type in strings of texts for the machine to analyze and predict the political affiliation of using the python input function.

Example from extendedSVM.py

```
done = 0
while done==0:
    text = input('Please write the text you want me to analyze:\n')
    print('hm.. let me think')
    txt = [text]
    #Binary prediction
    binary = binary_clf.predict(txt)
    binary_guess = binary_label.inverse_transform(binary)
    print('The text is more likely from: ', binary_guess)
    #multi class prediction
    multi = multi_clf.predict(txt)
    multi_guess = multi_label.inverse_transform(multi)
    print('The party it reminds me the most of is: ', multi_guess)
    loop = input('Do you need me to analyze more? Y/N\n')
    if loop=='N' or loop=='n':
        done = 1
        print('Then I will rest now...')
```

To make the machine return the prediction as a string rather than an integer, the code also had to include Scikit Learn's labelencoder.

Output example from the terminal



This is a small piece of code mainly intended to show, how it could make the code interactable for random piece of texts, should it be optimized to recognize unknown texts better. In the code above the code assess the binary dimension independently of the multiclass classification, which for instance is why it is able to classify the same speech as being both left wing and from V.

Conclusion

The purpose of this thesis was to investigate opinionated text by implementing a machine learning method to classify the texts according to their political affiliation. In this project I have implemented two machine learning methods to classify political speeches in Danish. Despite their simple level of preprocessing and setup both models managed to beat the set baseline and performed relatively well compared to current research suggestion that this may be a point of research to continue investigating.

Both machine learning models were tested on a test set consisting of unknown texts from another domain to investigate their potential generalizability to move beyond analyzing parliamentary speeches. Although the general performance decreased both models still managed to beat the set baseline for the unknown texts. Lastly I discussed the potential sources of error as well as the potentially broader perspectives and possibilities of using this kind of machine learning.

The project implemented a support vector machine as well as neural network. Both implementations were kept relatively simple and basic in their setup but despite that the neural network proved much more complex than the support vector machine. It therefore consumed a lot of time and computer processing power compared to the more simple support vector machine given the same technical environment, but without a justifiable better performance. Even if current computing power and development within the field allows for deeper learning methods, it is worth considering whether deeper learning methods such as the neural network are at a level of efficiency and performance compared to simpler methods for it to be a justifiable option.

In a future project I would like to take a more explorative approach to the data set prior to the learning to better understand the data at hand. All relevant research emphasized the importance of preprocessing and in that regard a future implementation could benefit from a more complex annotation scheme and amount of preprocessing.

In my project I was implementing the machine learning on a Danish data set, and it was severely slowing to the process that a lot of the basic resources for textual representation and analysis was either fragmentarily available with a lot of extra coding necessary to fit the tool to the task at hand or simply not developed for Danish at a satisfactory level to be used in a baseline or as part of a preprocessing process. To me it highlighted what was already noted in the literature that if Danish language technology wish to be able to follow current research and development of for instance English language technology, Danish research in computational linguistics require a lot more data and tools for build upon.

Literature

Books

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

Croft, W. B., Metzler, D., & Strohman, T. (2009). *Information retrieval in practice*. Pearson Education.

Jurafsky, D., & Martin, J. H. (2012). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Braille Jymico.

Liu, B. (2017). *Sentiment analysis mining opinions, sentiments, and emotions*. Cambridge University Press.

Manning, C. D., Schütze, H. (2003). *Foundations of statistical natural language processing*. MIT Press.

Provost, F., & Fawcett, T. (2013). *Data science for business: What you need to know about data mining and data-analytic thinking*. O'Reilly.

Coding material

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G, Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jozefowicz, R., Jia, Y., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Schuster, M., Monga, R., Moore, S., Murray, D., Olah, C., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Retrieved from https://tensorflow.org

Boulton, R. (2006). PyStemmer. Retrieved from http://snowballstem.org

Chollet, Francois and others. (2015). Keras. Retrieved from https://keras.io

Costa-Luis, C., Larroque, S., Mary, H., Yorav-Raphael, N., Korobov, M., Chen, G. (2018, May 22). tqdm/tqdm: tqdm v4.23.4 stable (Version v4.23.4). Zenodo. http://doi.org/10.5281/zenodo.1251290

Honnibal, M., Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. Unpublished. Version 2.0.12 Retrieved from https://spacy.io/

Hunter, J. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9, 90-95. DOI:10.1109/MCSE.2007.55

Jones, E., Oliphant, T., Peterson, P. et al (2001). SciPy: Open Source Scientific Tools for Python. Retrieved from http://www.scipy.org/

McCallum, Andrew Kachites. (2002) MALLET: A Machine Learning for Language Toolkit. Retrieved from http://mallet.cs.umass.edu

Oliphant, T. (2006). A guide to NumPy, USA. Trelgol Publishing.

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

Python Software Foundation. Python Language Reference, version 3.6.0. Retrieved from http://www.python.org

TFLearn Contributors (2016). TFLearn: Deep learning library featuring a higher-level API for TensorFlow, version 0.3. Retrieved from http://tflearn.org/

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. Retrieved from https://arxiv.org/pdf/1605.02688.pdf

Online media and other sources

Ammitzbøl, L. (2013, September 27). Dansk er truet på computeren. Retrieved from https://www.magisterbladet.dk/magisterbladet/2013/122013/122013_p13

Bauckhage, T. (2014, December 04). Platform Or Publisher? Whatever You Call It, It's The Future Of Media. Retrieved from https://techcrunch.com/2014/12/04/platform-or-publisher-whatever-you-call-it-its-the-future-of-media/

Carlton, R. (2014) 5 Ways Big Data is Changing ERP Software. Retrieved from https://www.erpfocus.com/five-ways-big-data-is-changing-erp-software-2733.html

Chaykowski, K. (2016, July 05). Meet The AI Team Powering Facebook's Language Tech Efforts. Retrieved from https://www.forbes.com/sites/kathleenchaykowski/2016/07/05/meet-the-ai-team-powering-facebooks-language-tech-efforts/#5df700901174

Data Never Sleeps 6 | Domo. (n.d.). Retrieved from https://www.domo.com/learn/data-never-sleeps-6

DESI (2018). The Digital Economy and Society Index (DESI). (n.d.). Retrieved from https://ec.europa.eu/digital-single-market/en/desi

Edelman (2018). *2018 Edelman Trust Barometer - Executive Summary*. Retrieved from http://cms.edelman.com/sites/default/files/2018-02/2018_Edelman_TrustBarometer_Executive_Summary_Jan.pdf

Flaiz, W. (2018, February 21). Using Chatbots As A Customer Service Virtual Agent. Retrieved from https://www.forbes.com/sites/forbescommunicationscouncil/2018/02/21/using-chatbots-as-a-customer-service-virtual-agent/#7aa4ca1b2a90

Folketinget (n.d.) Hvilke regler er der for sprog og opførsel i Folketingssalen? Retrieved from https://www.ft.dk/da/ofte-stillede-spoergsmaal/mode_hvilke-regler-er-der-for-sprog-og-opfoersel-i-folketingssalen

Fröhlich, J. (2018). Neural Net Overview. Retrieved from https://www.nnwj.de/neural-net-overview.html

Gartenberg, C. (2018, March 20). Google News Initiative announced to fight fake news and support journalism. Retrieved from https://www.theverge.com/2018/3/20/17142788/google-news-initiative-fake-news-journalist-subscriptions

Gartner (2013). *Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units by 2020*. Retrieved from https://www.gartner.com/newsroom/id/2636073

Hern, A. (2018, August 06). Facebook, Apple, YouTube and Spotify ban Infowars' Alex Jones. Retrieved from https://www.theguardian.com/technology/2018/aug/06/apple-removes-podcasts-infowars-alex-jones

Hvemstemmerhvad A (n.d). Hvem stemmer hvad i Folketinget? - om dansk politik og politikere med fokus på Folketinget. Retrieved from https://hvemstemmerhvad.dk

Hvemstemmerhvad B (n.d.) Positionering af politiske partier. Retrieved from http://www.hvemstemmerhvad.dk/analyse/positioner/

Hvemstemmerhvad C (n.d.) Ny dimension i dansk politik. Retrieved from http://www.hvemstemmerhvad.dk/analyse/positioner/samling2011-12.php

Hvemstemmerhvad D (n.d.) Politiske alliancer - se hvor ofte de politiske partier indgår i koalitioner med hinanden. Retrieved from http://www.hvemstemmerhvad.dk/analyse/alliancer.php?periode=all&alliance=&party_id=

Hvemstemmerhvad E (n.d.) Så ofte stemmer partierne sammen. Retrieved from http://www.hvemstemmerhvad.dk/analyse/stemmemoenstre/matrix.php

Larsen, L. (2015, March 31). Folketinget er delt i rød og blå blok. Retrieved from https://www.dr.dk/ligetil/folketinget-er-delt-i-roed-og-blaa-blok

Lloyd, C. (2018, April 03). Why Smart Fridges Are the Future. Retrieved from https://www.howtogeek.com/347408/why-smart-fridges-are-the-future/

Microsoft (2018). Microsoft Research Blog - Speech and Dialog. Retrieved from https://www.microsoft.com/en-us/research/blog/category/intelligence/speech-and-dialog/

Mitchell, A., Gottfried, J., Barthel, M., & Shearer, E. (2016, July 07). 1. Pathways to news. Retrieved from http://www.journalism.org/2016/07/07/pathways-to-news/

Morin, R., Lippman, D., King, E., Stan, M., Tismaneanu, V., Samuelsohn, D., . . . Anderson, E. (2018, July 04). French Parliament passes law against 'fake news'. Retrieved from https://www.politico.eu/article/french-parliament-passes-law-against-fake-news/

NATO. (2014, October 1). NATO Secretary General Anders Fogh Rasmussen. Retrieved from https://www.nato.int/cps/en/natohq/who_is_who_56703.htm

Newman, N. (2018). *Digital News Project - Journalism, Media, and Technology Trends and Predictions 2018*. The Reuters Institute for the Study of Journalism. Retrieved from http://reutersinstitute.politics.ox.ac.uk/sites/default/files/2018-01/RISJ%20Trends%20and%20Predictions%202018%20NN.pdf

Nielsen, F.Å. (2018, February 1). Danish Resources. Retrieved from http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/6956/pdf/imm6956.pdf

Nielsen, R.K. (2017, May 30). Where do people get their news? – Oxford University – Medium. Retrieved from https://medium.com/oxford-university/where-do-people-get-their-news-8e850a0dea03

Perez, S. (2018, July 2). Facebook rolls out more API restrictions and shutdowns. Retrieved from https://techcrunch.com/2018/07/02/facebook-rolls-out-more-api-restrictions-and-shutdowns/?guccounter=1

Rehm, G., & Uszkoreit, H. (2012). *The Danish Language in the Digital Age*. Springer Berlin Heidelberg.

Statsministeriet (n.d.) Regeringer fra 1953 til i dag. Retrieved from http://www.stm.dk/_p_7812.html

Varder, A. F. (n.d.). Om digitalisering af de historiske dokumenter. Retrieved from http://www.folketingstidende.dk/Laes_mere/Om_digitalisering_af_de_historiske_dokumenter.aspx

Wandsøe-Isaksen, R., Vasiljeva, K., Pedersen, F.D. (2018). *Analyse - Ingen truende medieudvikling - men bekymringspunkter*. Kraka.

Wearesocial (2018, January 30). *Digital in 2018: World's Internet Users pass the 4 billion mark.* Retrived from https://wearesocial.com/blog/2018/01/global-digital-report-2018

Wong, J. C., & Solon, O. (2018, August 10). Does the banning of Alex Jones signal a new era of big tech responsibility? Retrieved from https://www.theguardian.com/technology/2018/aug/10/alex-jones-banning-apple-facebook-youtube-twitter-free-speech

World Economic Forum (2016). *Digital Media and Society - Implications of a hyperconnected Era*. Retrieved from http://www3.weforum.org/docs/WEFUSA_DigitalMediaAndSociety_Report2016.pdf

Research and conference papers

Ahmed, A., Xing, E. (2010). Staying Informed: Supervised and Semi-Supervised Multi-view Topical Analysis of Ideological Perspective. *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11*

Biessmann, F., Lehmann, P., Kirsch, D., and Schelter, S. (2016): Predicting Political Party Affiliation from Text. Paper was presented to the International Conference on the Advances in Computational Analysis of Political Text (PolText). Croatia, July 2016. Retrieved from https://ssc.io/pdf/poltext.pdf

Bojanowski, P., Grave, E., Joulin, A. & Mikolov, T. (2017, June 19). Enriching Word Vectors with Subword Information. Retrieved from https://arxiv.org/abs/1607.04606

Borgesius, F. J., Möller, J., Kruikemeier, S., Fathaigh, R. Ó, Irion, K., Dobber, T., . . . Vreese, C. D. (2018, 02). Online Political Microtargeting: Promises and Threats for Democracy. *Utrecht Law Review, 14*(1), 82. doi:10.18352/ulr.420

Borin, L., & Edlund, J. (2018, July 03). Language Technology and 3rd Wave HCI: Towards Phatic Communication and Situated Interaction. Retrieved from https://link.springer.com/chapter/10.1007/978-3-319-73356-2_14

Chung, J., Gulcehre, C., KyungHyun, C., Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. Retrieved from https://arxiv.org/pdf/1412.3555v1.pdf

Dahllof, M. (2012, 04). Automatic prediction of gender, political affiliation, and age in Swedish politicians from the wording of their speeches--A comparative study of classifiability. *Literary and Linguistic Computing, 27*(2), 139-153. doi:10.1093/llc/fqs010

Diermeier, D., Godbout, J., Yu, B., & Kaufmann, S. (2011, 05). Language and Ideology in Congress. *British Journal of Political Science, 42*(01), 31-55. doi:10.1017/s0007123411000160

Dong, L., Wei, F., Tan, C., Tang, D., Zhou, M., & Xu, K. (2014). Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. doi:10.3115/v1/p14-2009

Elman, J. L. (1990, 03). Finding Structure in Time. *Cognitive Science, 14*(2), 179-211. doi:10.1207/s15516709cog1402_1

Enevoldsen, K., Hansen, L. (2017). Analysing Political Biases in Danish Newspapers Using Sentiment Analysis. *Journal of Language Works - Sprogvidenskabeligt Studentertidsskrift 2 (2), 87-98*

Gorton, W. (2016) Manipulating Citizens: How Political Campaigns' Use of Behavioral Social Science Harms Democracy'. *New Political Science,* 38 (1), 61-80 https://doi.org/10.1080/07393148.2015.1125119

Hochreiter, S., & Schmidhuber, J. (1997, 11). Long Short-Term Memory. *Neural Computation, 9*(8), 1735-1780. doi:10.1162/neco.1997.9.8.1735

Irsoy, O., & Cardie, C. (2014). Opinion Mining with Deep Recurrent Neural Networks. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. doi:10.3115/v1/d14-1080

Iyyer, M., Enns, P., Boyd-Graber, J., & Resnik, P. (2014). Political Ideology Detection Using Recursive Neural Networks. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. doi:10.3115/v1/p14-1105

Johnson, R., & Zhang, T. (2015). Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. doi:10.3115/v1/n15-1011

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of Tricks for Efficient Text Classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. doi:10.18653/v1/e17-2068

Khatib, K., Schütze, H., Kantner, C. (2012). Automatic Detection of Point of View Differences in Wikipedia. *Proceedings of COLING 2012: Technical Papers, pages 33–50*

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. doi:10.3115/v1/d14-1181

Le, Q., Mikolov, T. (2014). Distributed Representations of Sentences and Documents. *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, 1188-1196*

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278-2324. doi:10.1109/5.726791

Liu, P., Joty, S., & Meng, H. (2015). Fine-grained Opinion Mining with Recurrent Neural Networks and Word Embeddings. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. doi:10.18653/v1/d15-1168

Loper, E., Bird, S. (2002). NLTK: The natural language toolkit. *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, pp 62-69,*

Ma, M., Huang, L., Zhou, B., & Xiang, B. (2015). Dependency-based Convolutional Neural Networks for Sentence Embedding. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. doi:10.3115/v1/p15-2029

Maynard, D., & Funk, A. (2012). Automatic Detection of Political Opinions in Tweets. *Lecture Notes in Computer Science The Semantic Web: ESWC 2011 Workshops,* 88-99. doi:10.1007/978-3-642-25953-1_8

Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013a, September 07). Efficient Estimation of Word Representations in Vector Space. Retrieved from https://arxiv.org/abs/1301.3781

Mikolov, T., Chen, K., Corrado, G., Dean, J., Sutskever, I. (2013b, October 16). Distributed Representations of Words and Phrases and their Compositionality. Retrieved from https://arxiv.org/pdf/1310.4546.pdf

Mohammad, S., Kiritchenko, S., Sobhani, P., Zhu, X., & Cherry, C. (2016). SemEval-2016 Task 6: Detecting Stance in Tweets. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. doi:10.18653/v1/s16-1003

Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F., & Stoyanov, V. (2016). SemEval-2016 Task 4: Sentiment Analysis in Twitter. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. doi:10.18653/v1/s16-1001

Nguyen, T. H., & Shirai, K. (2015). PhraseRNN: Phrase Recursive Neural Network for Aspect-based Sentiment Analysis. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. doi:10.18653/v1/d15-1298

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - EMNLP '02*. doi:10.3115/1118693.1118704

Pla, F., Hurtado, L. (2014). Political Tendency Identification in Twitter using Sentiment Analysis Techniques. *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers.* 183-192

Poria, S., Cambria, E., & Gelbukh, A. (2015). Deep Convolutional Neural Network Textual Features and Multiple Kernel Learning for Utterance-level Multimodal Sentiment Analysis. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. doi:10.18653/v1/d15-1303

Recasens, M., Danescu-Niculescu-Mizil, C., Jurafsky, D. (2013). Linguistic Models for Analyzing and Detecting Biased Language. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics,* 1650–1659

Ringsquandl, M., Petkovic, D. (2013). Analyzing Political Sentiment on Twitter. *Analyzing Microtext: Papers from the 2013 AAAI Spring Symposium*

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*(6), 386-408. doi:10.1037/h0042519

Rosenthal, S., Nakov, P., Kiritchenko, S., Mohammad, S., Ritter, A., & Stoyanov, V. (2015). SemEval-2015 Task 10: Sentiment Analysis in Twitter. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. doi:10.18653/v1/s15-2078

Santos, C., Gatti, M. (2014). Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers.* 69-78

Severyn, A., & Moschitti, A. (2015). UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. doi:10.18653/v1/s15-2079

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: A simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research, 15, 1929-1958*

Søyland, M., Lapponi, E. (2017). Party Polarization and Parliamentary Speech. *ECPR 2017 General Conference.*

Tang, D., Qin, B., & Liu, T. (2015). Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. doi:10.18653/v1/d15-1167

Tang, D., Wei, F., Qin, B., Liu, T., & Zhou, M. (2014). Coooolll: A Deep Learning System for Twitter Sentiment Classification. *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. doi:10.3115/v1/s14-2033

Thomas, M., Pang, B., & Lee, L. (2006). Get out the vote. *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing - EMNLP '06*. doi:10.3115/1610075.1610122

Vicario, M. D., Bessi, A., Zollo, F., Petroni, F., Scala, A., Caldarelli, G., . . . Quattrociocchi, W. (2016, 01). The spreading of misinformation online. *Proceedings of the National Academy of Sciences, 113*(3), 554-559. doi:10.1073/pnas.1517441113

Wang, X., Weijie, J., Zhiyong, L. (2016). Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers.* 2428-2437

Wei, W., Zhang, X., Liu, X., Chen, W., & Wang, T. (2016). Pkudblab at SemEval-2016 Task 6 : A Specific Convolutional Neural Network System for Effective Stance Detection. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. doi:10.18653/v1/s16-1062

Wilson, T., Wiebe, J., & Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05*. doi:10.3115/1220575.1220619

Yano, T., Resnik, P., Smith, N. (2010). Shedding (a Thousand Points of) Light on Biased Language. *CSLDAMT '10 Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. 152-158

Yu, B., Kaufmann, S., & Diermeier, D. (2008, 07). Classifying Party Affiliation from Political Speech. *Journal of Information Technology & Politics, 5*(1), 33-48. doi:10.1080/19331680802149608

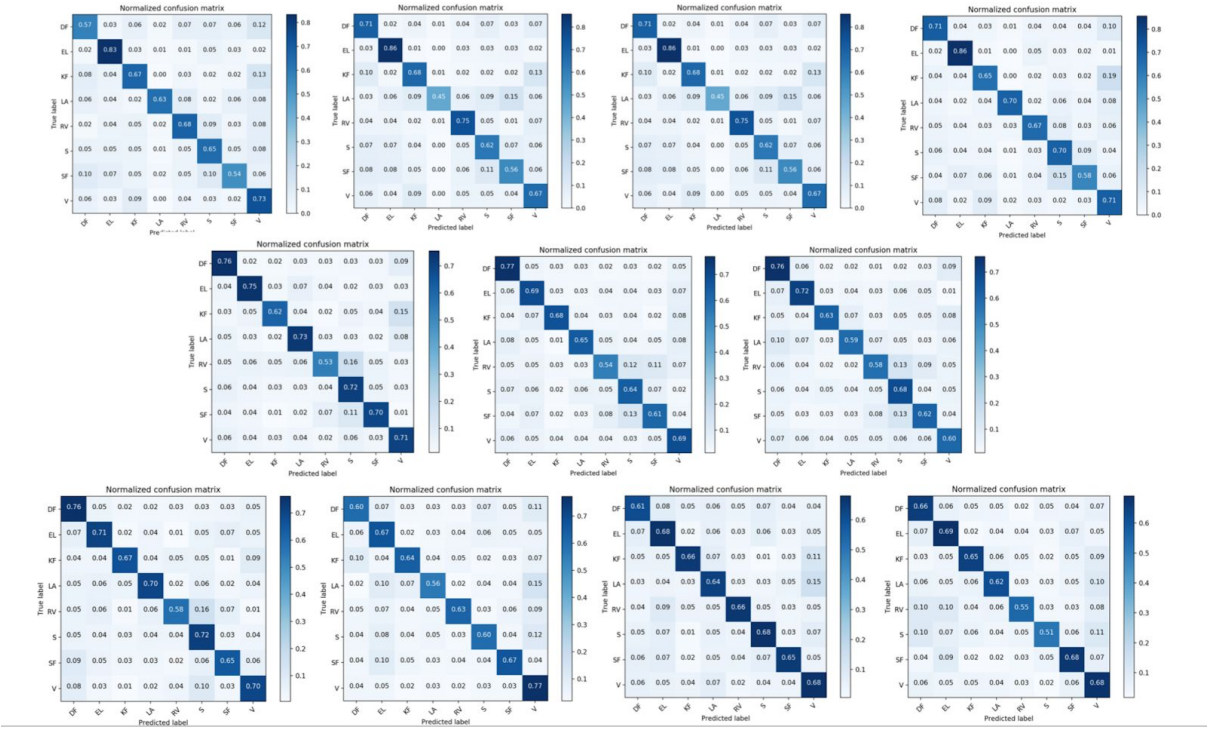
Appendix

Appendix A - Data set distribution

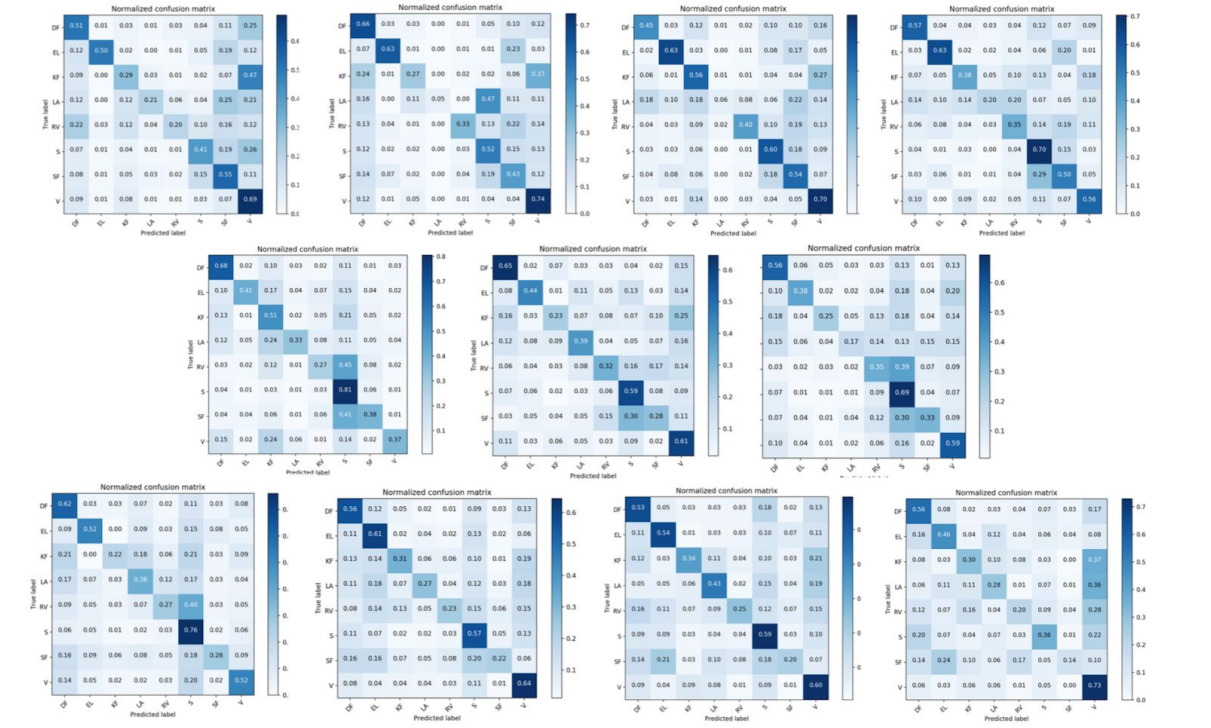
Folder	No. Of meetings	ALT	DF	EL	KD	KF	LA	RV	S	SF	V	LEFT	RIGHT	Total
2007-08 (1. samling)	10	0	172	223	0	151	58	140	247	133	347	743	728	1.471
2007-08 (2. samling)	87	0	2.518	1.978	0	2.184	477	1.352	3.577	1.976	4.001	8.883	9.180	18.063
2008-09	106	0	3.241	2.407	0	3.013	281	1.589	4.745	2.761	5.290	11.502	11.825	23.327
2009-10	109	0	3.395	2.276	1	3.601	536	1.758	7.055	3.493	7.377	14.582	14.910	29.492
2010-11	110	0	2.402	1.696	135	2.310	458	1.210	4.466	2.556	4.986	9.928	10.291	20.219
2011-12	102	0	4.105	2.279	0	1.330	1.869	1.871	5.153	2.701	5.421	12.004	12.725	24.729
2012-13	115	0	3.168	2.921	0	1.131	1.639	1.703	4.411	2.166	4.189	11.201	10.127	21.328
2013-14	109	0	2.544	2.392	0	1.149	1.250	1.762	3.968	1.639	3.776	9.761	8.719	18.480
2014-15 (1. samling)	98	12	2.362	2.165	0	888	1.227	1.425	4.475	1.520	3.465	9.597	7.942	17.539
2014-15 (2. samling)	6	50	34	140	0	31	50	45	39	51	130	325	245	570
2015-16	112	1.957	3.778	4.714	0	1.254	1.890	1.420	4.605	1.992	6.823	14.688	13.745	28.433
2016-17	112	1.815	3.840	3.452	0	1.601	2.821	1.347	4.700	1.799	5.251	13.113	13.513	26.626
2017-18	107	1.424	3.111	2.593	0	1.350	2.025	925	2.752	1.254	3.287	8.948	9.773	18.721
Total	**1.183**	**5.258**	**34.670**	**29.236**	**136**	**19.993**	**14.581**	**16.547**	**50.193**	**24.041**	**54.343**	**125.275**	**123.723**	**248.998**
		2,11%	13,92%	11,74%	0,05%	8,03%	5,86%	6,65%	20,16%	9,66%	21,82%	120.017	123.587	243.604
			14,23%	12,00%		8,21%	5,99%	6,79%	20,60%	9,87%	22,31%			

Appendix B - Confusion Matrices

Support vector machine over time. 07-17 moving in a reading direction.



Neural network over time



Appendix C - Politicians in twitter dataset

Politician	Party	No of tweets
Ida Auken	B	48
Martin Lidegaard	B	18
Morten Østergaard	B	46
Sofie Carsten Nielsen	B	108
Magrete Vestager	B	2
Zenia Stampe	B	3
Per Clausen	EL	25
Pernille Skipper	EL	43
Stine Brix	EL	2
Astrig Kragh	S	1
Benny Engelbrecht	S	143
Christina Antorini	S	50
Dan Jørgensen	S	4
Frank Jensen	S	6
Henrik Sass Larsen	S	3
Magnus Heunicke	S	21
Mattias Tesfaye	S	45
Mogens Jensen	S	25
Pernille Rosenkrantz-Teil	S	12
Jonas Dahl	SF	43
Pia Olsen Dyhr	SF	71
Rasmus Nordqvist	Å	41
Uffe Elbæk	Å	67
	sum	**827**

Politician	Party	No of tweets
Benedikte Kiær	C	15
Brian Holm	C	1
Brian Mikkelsen	C	22
Mai Mercado	C	5
Mette Abildgaard	C	5
Naser Khader	C	6
Nikolaj Bogh	C	1
Rasmus Jarlov	C	7
Søren Pape Poulsen	C	13
Hans Kristian Skibby	DF	5
Karina Adsbøl	DF	5
Liselotte Blixt	DF	17
Morten Messerschmidt	DF	1
Peter Kofod	DF	4
Peter Skaarup	DF	18
Anders Samuelsen	LA	24
Christina Egelund	LA	13
Joakim B Olsen	LA	18
Merete Riisager	LA	45
Ole Birk Olesen	LA	4
Simon Emil Ammitzbøl	LA	78
Lilian Parker Kaule	Løsgænger	5
Anne Ehrenreich	V	20
Caroline Stage	V	10
Ellen Trane Nørby	V	6
Jakob Engel-Schmidt	V	8
Jan E Jørgensen	V	24
Jens Stenbæk	V	9
Jørn Pedersen	V	10
Karsten Lauritzen	V	3
Kristian Jensen	V	50
Kurt Damsted	V	32
Lars Chr Lilleholt	V	9
Lars Løkke Rasmussen	V	14
Michael Aastrup Jensen	V	33
Morten Dahlin	V	6
Pia Allerslev	V	4
Sophie Løhde	V	12
Søren Pind	V	167
Torsten Schack	V	4
Troels Lund Poulsen	V	3
	sum	**736**

Appendix D - Excel calculations

Temporal multiclass testing

Year	SVM	Data Size	Neural Network
07-08	0,67072496	18063	0,4743
08-09	0,67723961	23327	0,5448
09-10	0,70338983	29492	0,5773
10-11	0,69487307	20219	0,5495
11-12	0,70157703	24729	0,5152
12-13	0,66807314	21328	0,5143
13-14	0,6525974	18480	0,4854
14-15	0,6993725	17539	0,5237
15-16	0,66654079	28433	0,5113
16-17	0,65874295	26626	0,4915
17-18	0,63179191	18721	0,4486

SVM	0,67499302
NN	0,51235455
	0,1287
	0,07159792

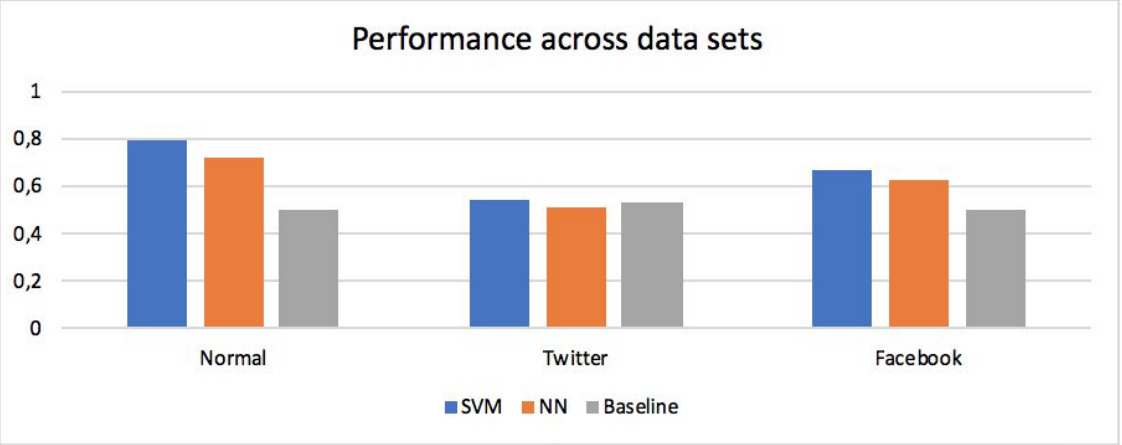
Temporal Binary testing

Year	Left		Right		SVM	Neural Network
07-08	8884	49,18%	9181	50,82%	0,859435529	0,7576
08-09	11503	49,31%	11826	50,69%	0,864181662	0,795113587
09-10	14585	49,45%	14912	50,55%	0,880420054	0,825762712
10-11	9939	49,12%	10294	50,88%	0,87709773	0,808300395
11-12	12007	48,53%	12732	51,47%	0,85904685	0,791835085
12-13	11202	52,52%	10129	47,48%	0,832397004	0,763009845
13-14	9763	52,81%	8723	47,19%	0,817593092	0,752028123
14-15	9599	54,71%	7945	45,29%	0,822727273	0,747578348
15-16	14694	51,66%	13750	48,34%	0,789898281	0,697188049
16-17	13120	49,26%	13515	50,74%	0,799550393	0,703265766
17-18	7525	43,50%	9774	56,50%	0,794219653	0,734682081
Average		50,00%		50,00%	0,836051593	0,761487636

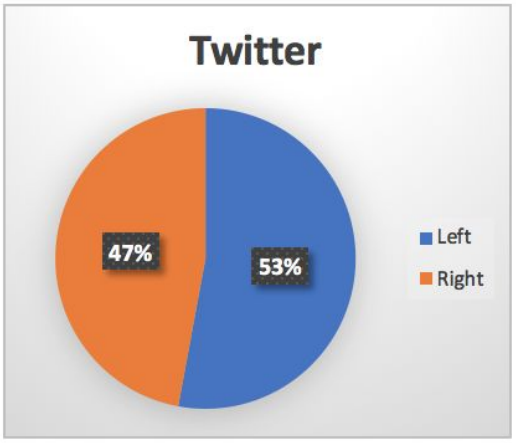
Testing on unseen data

Testing on unseen data

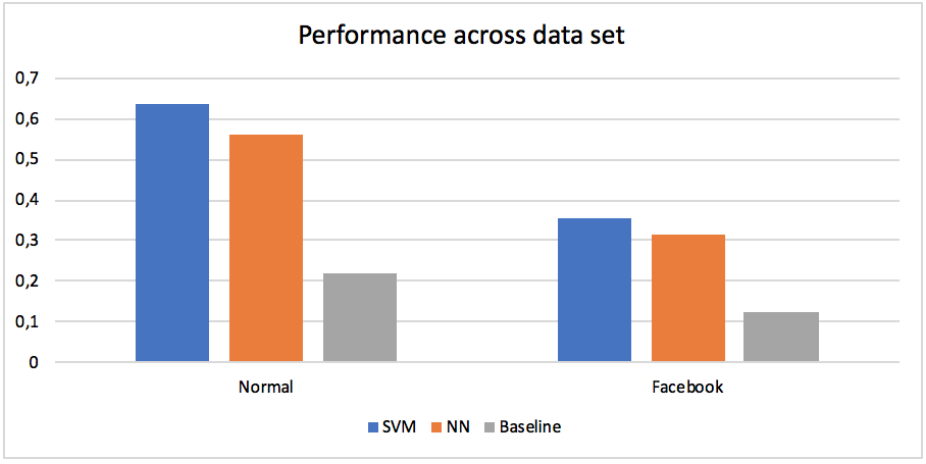
	Normal	Twitter	Facebook
SVM	0,7948	0,54557043	0,6628512
NN	0,7211	0,51497769	0,62226847
Baseline	0,502	0,52911068	0,5



	Left	Right
Twitter	827	736
Facebook	480	480

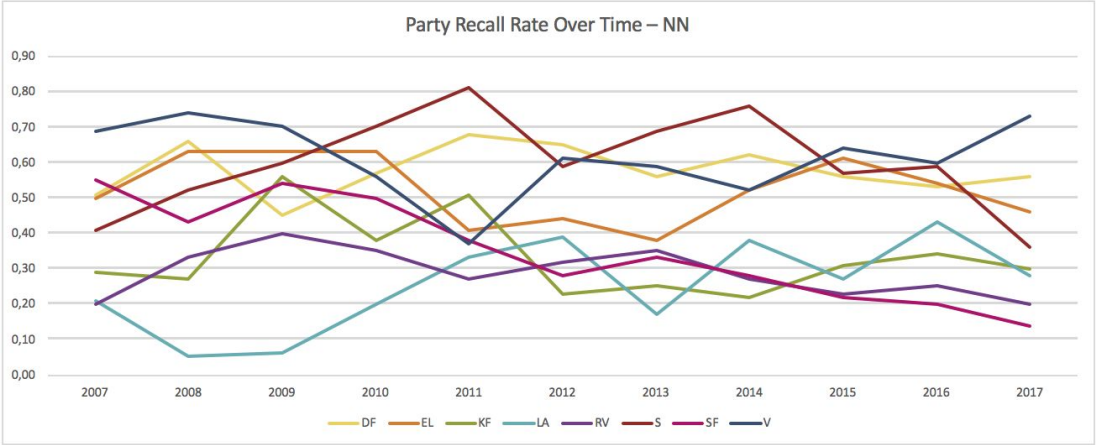


	Normal	Facebook
SVM	0,6364	0,35625
NN	0,56	0,31354167
Baseline	0,2182	0,125



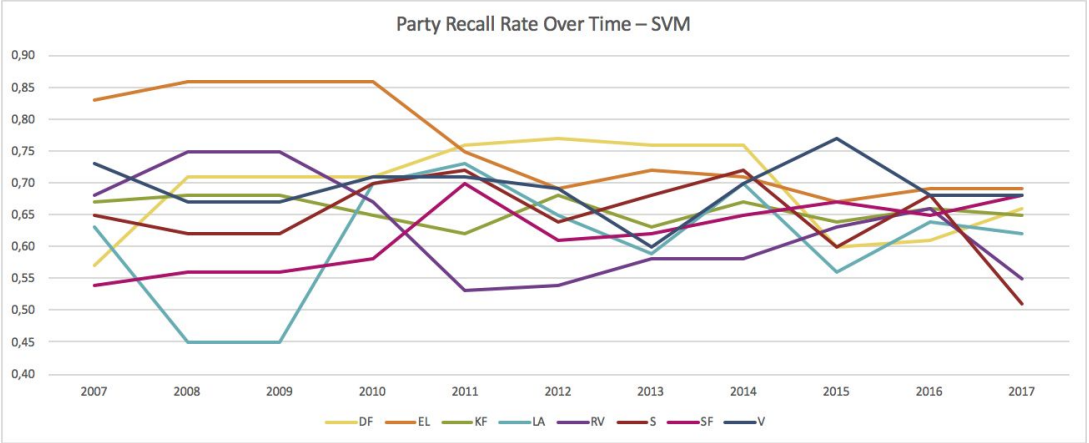
RNN over time (recall value)

NN	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
DF	0,51	0,66	0,45	0,57	0,68	0,65	0,56	0,62	0,56	0,53	0,56
EL	0,50	0,63	0,63	0,63	0,41	0,44	0,38	0,52	0,61	0,54	0,46
KF	0,29	0,27	0,56	0,38	0,51	0,23	0,25	0,22	0,31	0,34	0,30
LA	0,21	0,05	0,06	0,20	0,33	0,39	0,17	0,38	0,27	0,43	0,28
RV	0,20	0,33	0,40	0,35	0,27	0,32	0,35	0,27	0,23	0,25	0,20
S	0,41	0,52	0,60	0,70	0,81	0,59	0,69	0,76	0,57	0,59	0,36
SF	0,55	0,43	0,54	0,50	0,38	0,28	0,33	0,28	0,22	0,20	0,14
V	0,69	0,74	0,70	0,56	0,37	0,61	0,59	0,52	0,64	0,60	0,73



SVM over time (recall value)

SVM	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
DF	0,57	0,71	0,71	0,71	0,76	0,77	0,76	0,76	0,60	0,61	0,66
EL	0,83	0,86	0,86	0,86	0,75	0,69	0,72	0,71	0,67	0,69	0,69
KF	0,67	0,68	0,68	0,65	0,62	0,68	0,63	0,67	0,64	0,66	0,65
LA	0,63	0,45	0,45	0,70	0,73	0,65	0,59	0,70	0,56	0,64	0,62
RV	0,68	0,75	0,75	0,67	0,53	0,54	0,58	0,58	0,63	0,66	0,55
S	0,65	0,62	0,62	0,70	0,72	0,64	0,68	0,72	0,60	0,68	0,51
SF	0,54	0,56	0,56	0,58	0,70	0,61	0,62	0,65	0,67	0,65	0,68
V	0,73	0,67	0,67	0,71	0,71	0,69	0,60	0,70	0,77	0,68	0,68



Appendix E - Coding and requirements

Requirements:
Python
Keras
Tensorflow
Scikit Learn
tqdm

Python scripts uploaded/included:
Neural network:
my5thRNN.py (binary classification)
mymultiRNN.py (multiclass classification)
OOD-RNN.py (binary classification with unseen text)
OOD-MRNN.py (multiclass classification with unseen text)

Support vector machine
mySVM.py (both binary and multiclass classification
OOD-SVM.py (classification with unseen texts)

Sorting files
Clean_up.py
Sort.py
Slet.py

Datasets

Name	Contains
taler	Full dataset (multiclass)
binary_tale	Full dataset (binary)
Temporal	Full dataset divided according to year both binary and multiclass
politikere	Twitter dataset
fb_tale	Facebook posts (multiclass)
fb_tale_binary	Facebook posts (binary)

The embeddings are compressed but included in the folder. They will have to be unpacked.
All code should run without problem.
If the code needs to run of other datasets than the ones already plotted in the code, simply change the variable 'folders'