Exploration of using Twitter data to predict Swedish political opinion polls with neural networks

Alexander Gren, Klara Lundgren

Master's thesis
2023:E36

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

Centrum Scientiarum Mathematicarum

Contents

Abstract

This thesis aims to explore the possibility of using deep learning techniques to mine opinions on Twitter, with the objective to predict the political opinion distribution in Sweden. Different methods of gathering and annotating training data are evaluated to achieve accurate and reliable predictions. The models are quite successful at predicting test data, achieving F1-scores in the range of 70 % to 85 %. Some party divisions are found more difficult to classify than others. It is hypothesized and validated that the context of the tweets can aid in the classification process. In practice, this is carried out by exploiting the structure of the tweet thread structure. When the models were used to predict the general political discussion on Twitter, the results show that the predictions are subject to large variance. Different executions can yield wildly different results and, thus, are determined not reliable enough to use as input to regression when trying to find the relation between the predicted opinion distributions and the opinion polls. The underlying issue causing the large variance is investigated and results suggest that the training data is too small, or of too low quality, which causes the model to overfit and makes patterns hard to recognize. A lexicon-based classification is carried out as a supplement, but no significant relation can be stated between the predicted opinion and the opinion polls. Furthermore, it is discussed that the issue of insufficient results might lie within the method itself. The Swedish political discussion might not be polarized enough to make good classifications or Twitter political discussion might not be representative of the general opinion at all.

1 Introduction

1.1 Background

Machine learning is a popular technique where an algorithm receives training data to learn from before attempting to solve unseen problems. An emerging field in this domain is Natural Language Processing (NLP), which refers to the task of training computers to understand human languages and speech (Kavlakoglu 2020). This task poses a significant challenge since human language is rather ambiguous due to concepts such as sarcasm, metaphors, and other irregularities. Still, NLP appears in many familiar applications, such as chatbots, email spam detection, speech recognition, and sentiment analysis.

Sentiment analysis is a branch of NLP, where a model is taught to predict the sentiment of a sentence, such as emotions or sarcasm. Since a machine learning model has no prior knowledge of sentiment or human language, complex systems and extensive training is needed to accomplish this task. Sentiment analysis becomes an even more difficult task on social media platforms, where traditional grammar rules are often ignored, language is quickly changing and abbreviations are standard. Opinion mining is a branch of sentiment analysis where NLP-techniques are used to learn the opinion of a topic. As the spread of public opinion has shifted from physical distribution to social media, there is an emerging potential to mine the opinions of a multitude of individuals in real-time.

One of the most important opinion generating topics is politics. The knowledge of the public opinion in a political context is essential for politicians and society as a whole, especially close to an election. Each month, a Swedish political opinion poll is created by an organisation called Kantar SIFO, with the aim to get a sense of how the public would vote if the election was at that time. This is a costly operation where 20 000 interviews are being conducted monthly to try to gauge the current political landscape (Kantar 2023). Meanwhile, many political discussions are taking place online on various social media platforms. Perhaps mining the opinion of such platforms could serve as an alternative way of learning the public's opinion?

1.2 Objective

The goal of this research is to explore if it is possible to implement a neural network capable of predicting the political party affiliation of Twitter users in a Swedish context. Furthermore, this thesis seeks to assess the potential of using classified tweets as an alternative to opinion polls similar to what has been successfully implemented in the US. Predicting the political sentiment of an extensive amount of real-time tweets could serve as an opportunity to reduce the time and money spent on creating opinion polls.

1.3 Contribution from this work

The contribution of this work is to utilize and customize techniques used to mine political opinions on Twitter in a Swedish context. While several machine learning approaches have been suggested and validated for political opinion mining on Twitter, this work is primarily inspired by previous work where deep learning models have been employed for a similar task. The proposed approach was customized to fit the Swedish political landscape. Additionally, this work combines techniques for model initialization and pre-training to improve accuracy and for acquiring labeled training data from Twitter. As an extension, Twitter threads were utilized, with the aim of classifying more tweets by taking the context into consideration.

1.4 Previous research

This work was inspired by a technique called Iterative Opinion Mining using neural networks (Belcastro et al. 2020). The technique relies on an iterative classification approach, where the classification rules are updated at each iteration. This enables the model to start with a small training set of pre-classified tweets and grow it iteratively when more tweets are classified. The first tweets are annotated with party sentiment based on keywords commonly used in favour of the different parties, e.g., #VoteForTrump is annotated as a Trump-tweet in the US. After the classification process, the political view of each user is estimated based on their classified tweets. The resulting opinion distribution is compared to independent opinion polls to determine the accuracy of the model.

From a comparison of different machine learning methods used to predict sentiments on tweets concerning the Indonesian Presidential Election, it is concluded that deep learning models outperforms other machine learning techniques (Hidayatullah, Cahyaningtyas, and Hakim 2020). The deep learning models implemented are shown to outperform Support Vector Machines, Multinomial Naive Bayes, and Logistic Regression in terms of accuracy in predicting sentiment. The five deep learning techniques exhibit similar performance in terms of accuracy.

Severyn and Moschitt highlight the difficulty of initializing a deep learning model in their work on Twitter sentiment analysis (Severyn and Moschitti 2015). Since neural networks rely on optimization, the problem of local optima arises. To address this, one can try to initialize the model by pre-training word embeddings. Random initialization is compared to techniques such as pre-training and word2vec, described later. The comparison shows that models with pre-trained word embeddings are significantly more accurate in predicting sentiment.

Kamış and Goularas attempted to establish which Neural Network configuration excels at sentiment analysis on Twitter data (Goularas and Kamis 2019). It appears that dedicating time and effort towards developing high-quality training

sets offers more benefits than tinkering with various configurations or settings of the neural network.

2 Building of classification model

Twitter was picked as the social media for scraping in this study due to Swedish politicians' large presence on the platform and as it is commonly used for political discussions. Moreover, scraping from Twitter was proven successful in another study with similar objective (Belcastro et al. 2020).

The political views of the Swedish political parties have shifted over time, but are mainly driven by the three largest parties; Socialdemokraterna, Moderaterna, and Sverigedemokraterna. To simplify, the classification of parties was split into three divisions, which will from now on be referred to as *red parties*, *blue parties*, and the *yellow party*. The *red parties* refers to Socialdemokraterna, Miljöpartiet, Vänsterpartiet, and Centerpartiet. The *blue parties* refers to Moderaterna, Kristdemokraterna, and Liberalerna. Since Sverigedemokraterna is by itself the second largest party, this party alone will be referred to as the *yellow party*.

An overview of the four phases of the methodology proposed is shown in figure 1.

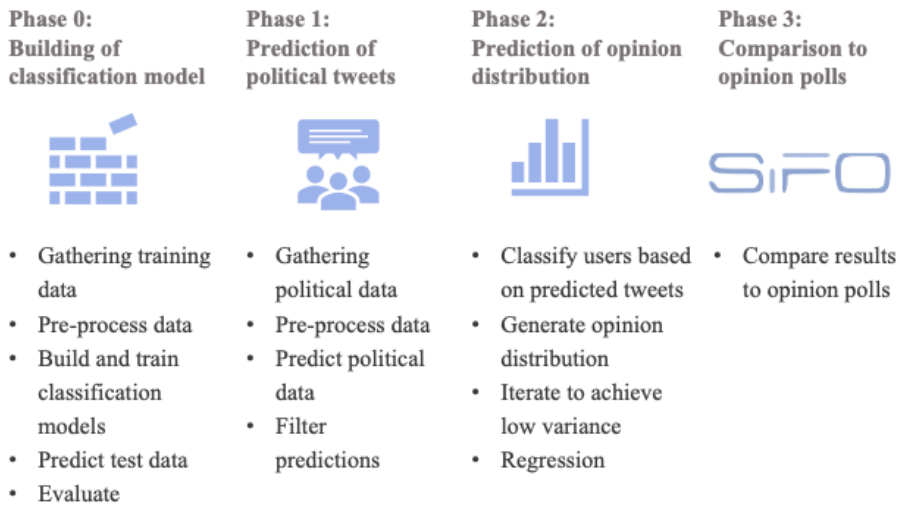

Figure 1: An overview of the methodology proposed.

2.1 Swedish political discussions on Twitter

A screening of Twitter showed that much of the political discussions were carried out under the hashtag "#svpol", an abbreviation for "Swedish politics". This assertion was further proven by the fact that the hashtag is one of the most frequently used overall on Swedish-speaking Twitter (Nilsson 2019). Furthermore, the word cloud created from tweets containing the hashtag #svpol presented in figure 2 shows that words commonly used under this hashtag are connected to politics.

Figure 2: Word cloud from #svpol 2022.

Some initial data exploration shows that the discussions are skewed by a loud minority. About 10 % of the users generated 53 % of the total tweets in a sample of 150 000 tweets, with a few individuals dominating the discussion, e.g. the most active user posted 543 of the tweets collected. This aspect has to be taken into consideration in the modeling to ensure the loud minority does not affect the generated opinion distribution.

To get a sense of the data, approximately 2000 tweets were manually read. The investigation showed that many of the tweets were unrelated to Swedish politics, e.g.

> *"VM guld i bordtennis! Grattis Mattias Falck/Kristian Karlsson !#svpol #vmguld #bordtennis"*

Moreover, many tweets carried their information in the form of a link or an image. With images and links being removed in pre-processing, this information is lost when the tweets are fed into the model. Other tweets were dependent on the context they were presented in, e.g. as a reply to a previous tweet. This

left many tweets with too little information to suggest any party sentiment. It is therefore likely appropriate that irrelevant and uninformative tweets are classified as neutral when constructing a model.

2.2 Data gathering

To gather tweets, keywords were used to scrape Twitter with *snscrape* (JustAnotherArchivist 2018). The keywords were used to retrieve relevant tweets, both for training data and for prediction. The keywords used to gather training data are presented in the following section. To gather data for prediction the hashtag "#svpol" was used and all mentions of parties or party leaders. Simultaneously as tweets were gathered, other specifications such as users, time, tweet ID, and reply ID were gathered.

2.3 Supervised vs unsupervised learning

In machine learning, there are two broad categories of learning, supervised learning and unsupervised learning (Delua 2021). In supervised learning, the model aims to predict new data, while the aim of unsupervised learning is to make sense of the data. The main difference between the two approaches is that supervised learning requires labeled data. The algorithm learns to recognize patterns based on the labeled data. The objective of this work is to classify tweets based on their political affiliation. This requires the model to predict and classify new data, and hence the supervised approach fits better. An overview of how supervised learning works is presented in figure 3.

The main limitation of supervised learning is acquiring a sufficient amount of correctly labeled training data, as the results are highly dependent on the provided training data. The difficulty of obtaining or labeling training data is a recurring theme in previous work. This process serves as the most crucial and complex part of this work. This is mainly due to the complexity of the data and the sensitivity to bias. The complexity of the data relates to how people express their political viewpoints in tweets. Few users express their opinions in a straightforward manner, rather the information is found between the lines.

Previous research suggests that deep learning models are more accurate than other supervised techniques in predicting the sentiment of tweets. It was also demonstrated that the performance between different deep learning models in terms of accuracy is similar. Consequently, a neural network is implemented in this work.

2.4 Training data

Three approaches to annotate training data were considered. Firstly, the data can be manually annotated, which in practice means reading tweets and labeling them. The benefits of this method are better control of the training data

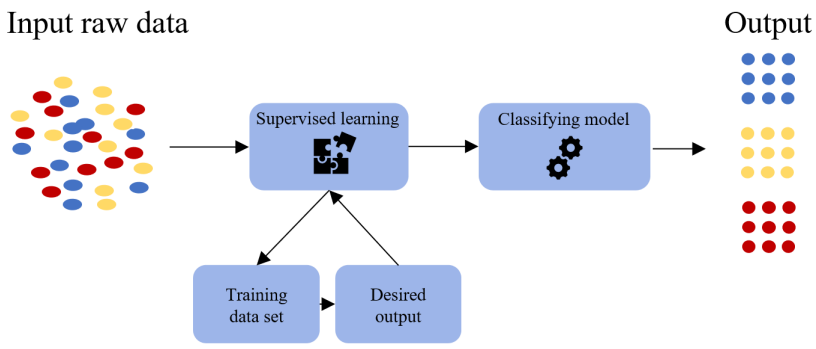

Figure 3: An overview of supervised learning.

set and more consistency in the labeled tweets. The downside is that letting humans manually annotate implies influence by the biases of said humans. It is also a time-consuming process. Secondly, the tweets can be annotated based on keywords that strongly suggest an affiliation to a specific party division. In practice, this would mean identifying keywords often used in favour of a specific party. The benefits of this method are less influence of human bias in selecting tweets, and less time needed to annotate tweets. Although, identifying representative keywords is also difficult and subject to human bias. The keywords might also generate mislabeled tweets and affect the outcome of the classification. Thirdly, one can scrape based on mentions of parties or party leaders and then use a sentiment analyzer to filter away negative or neutral tweets so only positive tweets remain.

The methods chosen were to annotate the training data based on keywords as well as to use party mentions in combination with lexicon-based sentiment analysis. These were picked due to time efficiency. Using two techniques permits more breadth in the training data. The size of the training data set is limited by the quality of the keywords. A large training data set prevents overfitting, but more noise and low quality of the training data effects the classification negatively.

2.4.1 Annotating with keywords

The method to annotate training data with keywords was implemented by manually examining 2000 tweets, identifying keywords or hashtags commonly in favour of each party division. In total, 4500 tweets were scraped, evenly distributed over the three categories of parties, i.e., 1500 tweets for each party group. The tweets were scraped based on the first set of keywords presented in figure 4. The number of keywords needed was decided by how frequently used these keywords are, e.g., for the blue parties, more words were needed to reach

Priority of keywords	Blue parties	Yellow party	Red parties
Primary keywords	(s)verige	sd2022	blåbrun
Secondary keywords	rödgrön röra	migpol	
Tetriary keywords	sossar AND haveri		
	socialdemokraterna AND haveri		
	sossarna AND haveri		

Figure 4: First set of keywords utilized for annotation

Priority of keywords	Blue parties	Yellow party	Red parties
Primary keywords	#MagdalenaAndersson	sd2022	blåbrun
Secondary keywords	skatter	migpol	högerextrem
Tetriary keywords	skatt		

Figure 5: Second set of keywords utilized for annotation

the goal of 1500 tweets for a specific time frame.

The keywords chosen to label training data play a crucial role in how the model classifies tweets. To explore the impact of the set of keywords, another data set is created based on another set of keywords. The second set of keywords is shown in figure 5.

2.4.2 Annotating with VADER

The second method to annotate training data was implemented with the help of a lexicon-based sentiment annotation. Tweets were scraped based on mentions of party or party leader and a sentiment analyzing tool was used to filter away all the tweets that are not in support of the mentioned party. This approach was implemented using a Valence Aware Dictionary and Sentiment Reasoner (VADER) (Hutto and Gilbert 2014). VADER is used to predict sentiment on text and is tuned to be used on social media. In practice, this means that it is sensitive to the polarity of sentiments found on social media. In the Swedish version, the lexicon is created by 10 independent human raters, which resulted in 7500 features rated. Both emoticons and slang abbreviations are considered. The rule-based tool captures the order and relationship between words and therefore gathers more information than a bag-of-word model. Each word in the processed text receives a sentiment rating between -1 and 1. Below -0.05 results in a negative sentiment, and above 0.05 results in a positive sentiment. A score in between results in neutral sentiment. A compound score for the sentence is calculated by adjusting the score for each word based on rules, and then summing all scores and normalizing it between -1 and 1.

2.5 Pre-processing

Pre-processing raw data involves cleaning, normalizing, and transforming it before it is fed to the neural network. Only valuable information should be passed to the neural network. The aim of pre-processing is to remove pieces of the text that do not carry any information about the sentiment, or in other words to reduce noise in the text. The purpose of reducing the noise is to improve the performance and efficiency of the classifier. This work was inspired by the pre-processing approach proposed by Belacastro (Belcastro et al. 2020) in his work on political polarization on Twitter.

As a first step in the pre-processing, all links and pictures were removed. All letters were converted to lowercase and all punctuation was removed, which also includes symbols and special characters. Emoticons have the potential to add value to the sentiment but would require a deeper analysis before they are used in the classification. They have unfortunately not been exploited in this thesis but have the potential to aid in the classification of future work.

In the process of screening 2000 tweets, it was observed that many tweets contain mentions, as they are often in reply to other tweets. These mentions create a thread structure, and the possibility of utilizing these in the classification process will be discussed later. The mentions themselves generally do not add any value to the sentiment and are therefore removed. An exception is made in the keyword training set, if a party leader or a party is mentioned, since the mention of these in combination with positive sentiment is deemed valuable information. In the lexicon-based training set, these mentions were removed as party leaders and party names were used in the scraping process. Keywords used to scrape and annotate the training data were removed, as the model otherwise would recognize these as an indication of positive sentiment.

Common stopwords were removed since they do not add information about the sentiment. The purpose of these words is to link words that carry more information. These are words such as *att, den, det, jag, han.* A list of stopwords was retrieved from an open-source Github created by Peter Dalle and consists of 427 common stopwords in the Swedish language (Dalle 2021). An illustration of the output of the pre-processing is shown in figure 6.

In the work by Belacastro, stemming is also performed on the tweets (Belcastro et al. 2020). Stemming is the process of converting words to their root form, e.g., from voting or vote to vot. This allows all conjugations of the word to be interpreted as the same word. Stemming is preferred since regardless of the conjugation, the word bears the same information about the sentiment. Due to the difficulty of acquiring a set of root words and their conjugations in Swedish, and the labour-intensive work of creating such a dictionary, stemming has unfortunately not been included in this work. Stemming could potentially be explored in future work.

Tweet	Pre-processed	Tokenized
@fredrikhardt @Centerpartiet Och för att det går så bra för Sverige jämfört med våra grannländer 😅😅😅	centerpartiet sverige grannländer	[11, 5, 0, 0, 0, 0, 0 …]
@home4metoo @rektor_linnea @socialdemokrat @isakskogstad Helt korrekt. Han är oerhört kunnig och vass.	socialdemokrat korrekt oerhört kunnig vass	[3, 939, 717, 0, 0, …]
@AndersWidh @perteng1 @MatsErikson @socialdemokrat Vi SD-positiva kanske tycker det är en lite märklig samhällsutveckling vi fått som är samtida med en ohämmad invandring. Har fått uppleva lite för mycket våld mot svenska pensionärer, tjejer och män som inte har annat gemensamt än att de är svennehoror.	socialdemokrat sdpositiva tycker märklig samhällsutveckling samtida ohämmad invandring uppleva våld svenska pensionärer tjejer män gemensamt svennehoror	[3, 22, 692, 339, 20, 231, 0, 0, …]
Det här är ett brutalt angrepp på äganderätten och livsmiljön. Allt för att miljöpartister socialdemokrater vänsterpartister och centerpartister i 08 området - som inte berörs av vindkraft - inte vill ha kärnkraft. Rösta bort dem i valet! #svpol #vindkraft https://t.co/OzFv7qzc7n	brutalt angrepp äganderätten livsmiljön miljöpartister socialdemokrater vänsterpartister centerpartister 08 området berörs vindkraft kärnkraft rösta valet vindkraft	[301, 90, 55, 54, 301, 0, 0, 0, …]

Figure 6: Example of Twitter pre-processing.

2.6 Feature extraction

Since machine learning models can only handle numbers as input, all words need to be transformed into numbers. The procedure of generating unique numbers, or tokens, for each word, is called tokenization (Uzila 2022). These tokens need to be constructed in a way that gives meaning to the deep learning model. This feature extraction is an important step when implementing a deep learning model and can be done in several ways.

The simplest method is to assign a distinct number to each word and count its frequency within the text corpus. The text corpus comprises all the words contained in the collection of tweets. This technique, known as *bag of words*, has multiple variations. Since all stopwords are removed, the model will learn the tweet features based on the tokens with high frequency and likely also the most interesting in the context. One major limitation of this method is that it does not account for the relationship between tokens. Consequently, words that bear meaning together in a sentence will be considered independently and this information is neglected. An illustration of this issue is shown in figure 7. In the sentence *"rösta inte på sossarna"* the party *"sossarna"* is the most frequent word in the text corpus and the word *"rösta"* is the fifth most frequent word. The word *"rösta"* would by itself indicate that the tweet is in favor of the party mentioned. Although if the relationship *"rösta inte"* would be considered, the model would learn that this combination of words would indicate a tweet in opposition to the party.

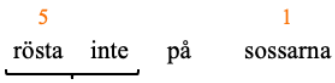

Figure 7: Example of bag of words

In this work, the tweets were tokenized with the help of the Keras tokenizer (Chollet et al. 2015). The tokenizer was fitted on all tweets and a dictionary of all the words and how frequently they appear. The tweet was converted to a number vector where only the 1000 most frequent words are considered. The numbers represent how common the words are in the text corpus. The vector was then zero-padded to ensure that the size of the input to the neural network is the same.

2.6.1 Keras embedding layer

An alternative to using only the bag of words method when encoding the sentences is to use word embeddings. The embedding works as a dense vector representation of each word (Saxena 2021). During the training, the weights of the embedding layer were trained to an optimal value to fit the training data. The embedding layer learns the structure of the training data and represents each word with an embedding. Similar words in the training data were represented with similar embeddings. Visually, the words would be closer together in the space spanned by the dimension of the features.

The number of features to consider, or the dimension of the embedding, depends on the specific problem. The complexity of the problem increases with the number of parameters to optimize, which increases with the embedding dimension. Increasing complexity in optimization is associated with problems such as getting stuck in local minima and unreliability in the system. A dimension of 100 to 300 words have been proven to perform well in word embeddings (Gu et al. 2021).

Furthermore, one can train the embeddings simultaneously with a machine learning model, or pre-train the embeddings before. An advantage of learning the embedding simultaneously is that the embedding is specifically trained on the text corpus related to the problem statement (Brownlee 2019). Since the construction of sentences in tweets is profoundly different from the sentences found in for example books there is a possibility that training the embeddings simultaneously works better than using a pre-trained embeddings. The downside of this approach is that the training data set needs to be large and the training process may be slow.

2.6.2 Word2vec

Another alternative is to pre-train the embeddings on a large text corpus. This addresses the issue of limited training data. As mentioned, a large training set is required to receive representative word embeddings. Since labeled training data is difficult to acquire in this context, the embeddings can learn from a text corpus. The pre-trained embeddings may then be updated with the training of the neural network or remain unchanged.

As seen in the work of Severyn and Moschitti, pre-training word embeddings can significantly increase the accuracy of Twitter sentiment analysis with neural networks (Severyn and Moschitti 2015). In this work, the word2vec model was utilized to pre-train the embeddings. The word2vec method is a neural network itself that predicts one token given another token. This means that if the word *"rösta"* is present, the model will also consider the word *"inte"* if these two words are commonly used as neighbouring words in tweets. By learning the language structure, the word2vec model can provide valuable insights for the model when classifying tweets. During the training on the text corpus, an embedding vector is generated for each word, which can then be utilized by the deep learning model.

Numerous pre-trained word embeddings for English have been developed using large text corpora, e.g., a free pre-trained version by Google News data set. However, obtaining pre-trained models in Swedish is more difficult, and could be further explored in future work. One option is to pre-train the word2vec model using a text corpus of thousands of Swedish tweets, the drawback is that this approach only considers neighbouring words and is limited to the words present in the training data. Therefore, it is essential to use a very large text corpus to train the word2vec model. These pre-trained embeddings can then be used as a layer in the deep-learning model.

The word2vec model was trained on a dataset of 10 000 tweets scraped with the keyword *svpol* in 2022. An illustration of the word embeddings can be seen in figure 8. The illustration shows how words that are often used in neighbouring contexts are grouped together. One can for example see that Annie Lööf and Märta Stenevi are close to each other and that Ulf, referring to prime minister Ulf Kristersson, is closer to Johan Pehrson. This shows how leaders of parties with similar political view seem to be mentioned in the same context. There is also a group of words that are far from all other words, such as *islam, klankultur, krimpol*, and *hedersförtryck*. This would indicate that these words are often used in the same context, and seldom together with other political words.

2.6.3 Swedish BERT

Both the Keras embedding layer and the pre-trained word2vec embeddings have fixed embeddings for each word, independent of the kind of sentence they ap-

pear in. An alternative is to use context-aware models such as the Bidirectional Encoder Representations from Transformers (BERT) (Börjeson 2020). The advantage of BERT is that it creates dynamically updated word embeddings that are based on the sentence the word appears in.

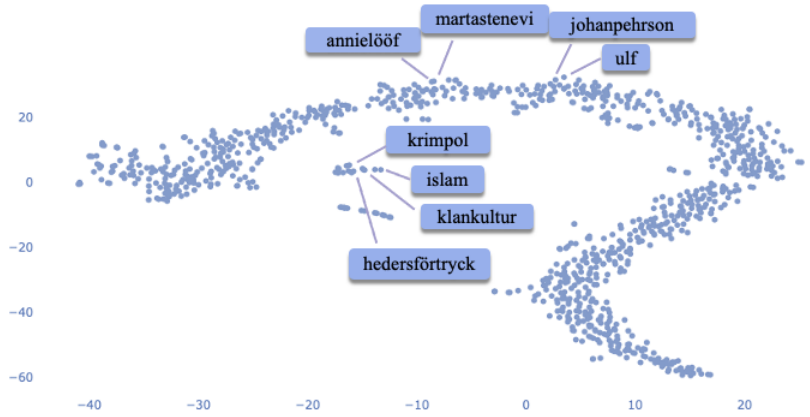

Figure 8: Visualisation of word embeddings from word2vec

The problem with implementing BERT is that the free and existing models are based on and trained in English. However, the Swedish Royal Library released a Swedish version of Google's BERT model in 2020 (Malmsten, Börjeson, and Haffenden 2020). According to the institution the Swedish version outperforms Google's multi-language model. The model developed by the Royal Library is trained on approximately 200M sentences from various sources aiming to provide a representative BERT model for Swedish. When the Swedish BERT model was used, the vocabulary proved to be insufficient for the task at hand.

Using the implementation of the Swedish BERT model to tokenize 1000 political tweets, 61 % of the tweets were empty tokens, and 20 % of the tweets contained only one token. The implication is that the model either did not recognize, or assign any value to the words in these sentences. The words it was able to recognize were often very general, e.g., "Sverige", that do not provide much information regarding political opinions. Since these tokenized tweets are the sole information given to the classifying model, the extensive amount of information that gets lost is unfavorable.

The following tweet can be used to illustrate this issue.

> *"Är man bosatt i Sverige så har man en skyldighet att lära sig svenska. Vägrar man göra det, trots att utbildningen är gratis, så får man också stå för tolkkostnaderna. Det är inte rasism att ställa krav."*

With the Swedish BERT encoder only the word "Sverige" is translated to a token. Words such as "svenska", "rasism" and "krav" that might add value to the classification are left out. It was thus concluded that the Swedish BERT Model vocabulary needs to be expanded before it can be used well in this application.

2.7 Artificial Neural Network

Over the extensive process of evolution, numerous desirable traits have been developed in the human brain. Artificial neural networks try to mimic some of the brain's features to achieve similar abilities.

2.7.1 Neurons

In a biological sense, a neuron is a specialized cell that plays a crucial role in processing information (Jain, Mao, and Mohiuddin 1996). Neurons receive signals or impulses from other neurons. Signals passing through the synapse can modify its effectiveness, allowing synapses to learn from their participation in activities. This ability to adapt based on history acts as a memory.

Modelling the behaviour of these biological neurons with inputs and outputs can look like

$$y = \theta \left(\sum_{j=1}^{n} w_j x_j + u) \right) \tag{1}$$

where x_j represent the inputs to the neuron and is the output. The constant u represents the bias in the modeling and θ the activation function.

The activation function in a neural network enables the neurons to have higher complexity than just a linear function. While linear equations may be straightforward to solve, their capacity for complexity is inherently limited and they are incapable of learning and recognizing intricate data mappings. Neural networks without activation functions operate similarly to linear regression models, exhibiting limited performance and power in most scenarios (Sagar Sharma, Simone Sharma, and Athaiya 2017). The activation function allows for nonlinear functional mappings between input and output.

The choice of activation function depends on the application of the neural network. The sigmoid activation function, visualized in figure 10, was commonly used in neural networks for many years because of its smooth and continuous output, which made it well-suited for gradient-based optimization techniques (Eger, Youssef, and Gurevych 2019). However, as neural networks grew deeper, it became apparent that the sigmoid function suffers from the vanishing gradient problem, which occurs when the derivative of the activation function approaches zero.

The ReLU (Rectified Linear Unit) activation function solves this issue. It has a simple and efficient form and does not suffer from the vanishing gradient problem in the same way as the sigmoid function. Its derivative is either 0 or 1, depending on the input value. The ReLU function is now widely used in practice and has become the default choice for many deep-learning applications.

The Softmax activation function is a type of logistic regression that maps an input value into a probability, describing the probability of a particular outcome (Mahmood 2018).

2.7.2 Teaching the Network

The *loss function* is used to define the performance of the neural network. The model seeks to minimize the *loss function* in order to achieve the best possible accuracy. Cross entropy is a commonly used loss function and is calculated based on the log-likelihood (Qin, Kim, and Gedeon 2019).

$$L = -\frac{1}{n}\sum_{k}(y_k ln(\hat{y}_k) + (1 - y_k) ln(1 - \hat{y}_k)) \tag{2}$$

where y is the correct outcome, $\hat{y}$ is the prediction, and n is the number of items in the training data (Nielson 2020). The training process aims to minimize the loss function by updating the weights associated with the neurons. This is done through backpropagation, which is the process of utilizing the partial derivative of the error of each weight to optimize for loss minimization (Sathyanarayana 2014).

A commonly used method for optimization is called ADAM. It provides an efficient stochastic optimization using first-order gradients and requires little memory. It works by calculating adaptive learning rates for different parameters using estimates of the first and second moments of the gradients (Kingma and Ba 2017).

The training is performed in batches, and the batch size determines the number of samples the neural network processes before re-adjusting the weights of the neurons (Kukreja et al. 2016). To ensure that the weights are tuned correctly, the network should process many batches. To increase the number of available batches, the data set can be run through the network several times. The number of times the entire data set is processed by the network is defined by the *epoch* parameter. Two epochs imply letting the network train on the data set twice.

Early stopping is a regularization method used in neural networks to prevent overfitting (Prechelt 1998). Overfitting occurs when the model is too complex and is trained too well on the training data, leading to poor generalization performance on new data. Early stopping works by monitoring the performance of the neural network during training. It is measured after each epoch of training,

if the performance starts to deteriorate on the testing set, indicating overfitting, the training process is stopped early. A useful parameter in early stopping is called patience and dictates how many epochs without improvement that is allowed before stopping early.

2.7.3 Layers

The neural network is divided into different layers. The entry point for the data is called the *input layer* and the exit point is called the *output layer*. In between these points, there can be *hidden layers* that the users do not handle themselves. The architecture of the input and output layers is determined by the structure of the input data. For instance, when analyzing pictures, the number of pixels would dictate the shape of the input layer. In the context of Twitter posts, the maximum number of words allowed determines the structure of the input layer. The output layer produces the final prediction of the neural network based on the inputs that have been fed into the network and adjusted by previous neurons. The desired output of the neural network determines the shape of the output layer. For a binary problem, one neuron is sufficient. For a classification problem with three possible outcomes, three nodes are required that will produce a probability of the input being from each respective class. It is the hidden layers that perform the bulk of the work. Determining the shape of the hidden layers is a difficult task, as there are no perfectly accurate ways of calculating the best shape analytically (Brownlee 2019). Deciding on the shape of the hidden layers is often done by testing parameters randomly or with a grid search. An illustration of how a neural network can take form is shown in figure 9.

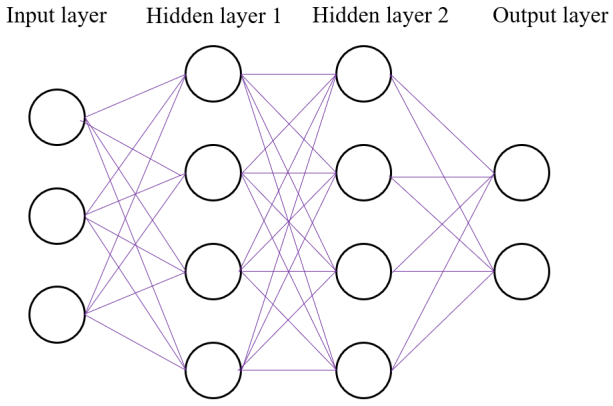

Figure 9: Illustration of neurons and layers and how they are connected in a neural network.

One type of hidden layer is called *flatten*, the flattening layer is used to change the shape of the input tensor before being fed into the subsequent layers (Chollet et al. 2015). By flattening the input tensor, the flattening layer preserves the spatial relationships between the features of the input, while making it easier to process the data in a fully connected layer. Another hidden layer is called *dense*, the dense layer is a fully connected network, meaning it connects each neuron in the layer to every neuron in the previous layer. Every single neuron in the layer produces an output value, which is then passed on to the next layer in the network. The dense layer is defined by the number of neurons in the layer and an activation function. The activation function is applied element-wise to the output of the neuron.

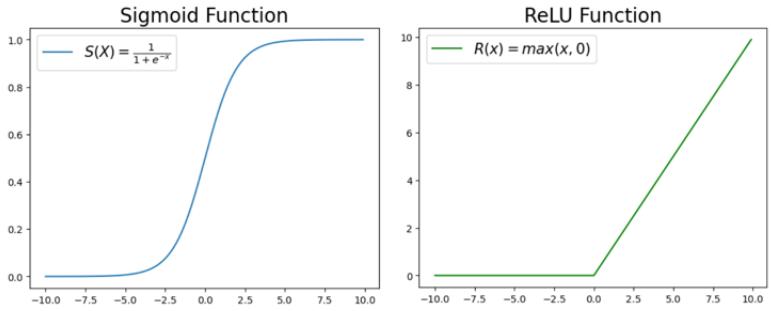

Figure 10: Sigmoid and ReLU activation functions.

2.7.4 Variance and bias trade-off

Variance in a neural network is natural, and there is usually a trade-off between variance and bias (Hinton et al. 2012). The variance tends to be large if the model overfits the training data, and learns the sampling error in the training data set. A neural network tends to overfit the training data when the set of training data is small. One way to overcome this issue is to average over many different models. To receive different neural network models one can change the number of hidden neurons, use different learning algorithms, or change the training data. This procedure is called bagging. In practice, this could mean training the model on different subsets of the training data and drawing an average of the models' results.

Another alternative is to create expert models, that focus only on predicting the classes where it outperforms the other models (ibid.). In this work, that would translate to using different models to predict blue parties, red parties, and the yellow party. Instead of creating new models and averaging over predictions, one can use dropout as a way of averaging over models (ibid.). Dropout means that at each training each neuron is omitted with a given dropout probability. Consequently, sampling is done with different architectures in each epoch.

Dropout is efficient when the neural network is overfitting. This is an extreme form of bagging.

If limited training data is an issue, a simple model will perform better than a complex one since the complex model will overfit the training data (Hinton et al. 2012). Each parameter has posterior probability which means that a large amount of trainable parameters will increase the variance. A more complicated model will fit the training data better, but will not be reliable when applied to a new set of data.

2.7.5 Utilized architecture

Two versions of neural network architectures were utilized in this work and the structure is shown in figures 11 and 12. Both have input layers of size 100, allowing 100 words to be fed into the model as none of the tweets in the data set contained more than 100 words after the pre-processing. The shape of the embedding layer is 100 times 100, with each word being embedded in a vector of size 100. The embedding layer utilizes the pre-trained word2vec embeddings to capture relations between words. The dimensionality of the word embeddings was chosen within the range of common practices described earlier, as the lower endpoint to maintain a simple structure. The embedding information, e.g., if two words often are used together, is described in a 2D matrix.

Subsequently, a flattening layer was performed in order to preserve information from the embedding layer that produces a 2D vector when the dense layers require a single vector as input. The shape of the flattening layer is 10 000, a result of the embedding layer creating a matrix with 10 000 elements.

A dense layer of 10 neurons follows the flattening layer. This number is empirically decided upon simply by observing that an increase in the number of hidden neurons did not result in an improvement in accuracy. This layer has the ReLU activation function previously described. This activation function was picked due to its benefits in the optimization over other alternatives.

Lastly, the output layer is added. The amount of parameters differs in the two versions of the neural network, because of how the classification is structured. The rationale for this division will be discussed in-depth in the following paragraph. Figure 9 shows a model providing a 3-way classification, thus having 3 neurons in its output layer while figure 10 shows a model with a binary classification, therefore only containing one neuron in the output layer.

Furthermore, the ADAM optimization algorithm was used and to prevent overfitting early stopping was implemented with a patience of 3.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_121 (Embedding)    (None, 100, 100)          2272400

 flatten_2 (Flatten)          (None, 10000)             0

 dense_4 (Dense)              (None, 10)                100010

 dense_5 (Dense)              (None, 3)                 33

=================================================================
Total params: 2,372,443
Trainable params: 2,372,443
Non-trainable params: 0
_________________________________________________________________
```

Figure 11: Base model architecture.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_86 (Embedding)     (None, 100, 100)          1477800

 flatten_1 (Flatten)          (None, 10000)             0

 dense_2 (Dense)              (None, 10)                100010

 dense_3 (Dense)              (None, 1)                 11

=================================================================
Total params: 1,577,821
Trainable params: 1,577,821
Non-trainable params: 0
_________________________________________________________________
```

Figure 12: Two-step classification model architecture.

2.8 Designing model structure

Two structures of classification models were implemented with the aim of classifying tweets in favour of the three party divisions.

2.8.1 Base-model

The first classification structure is hereafter referred to as the *Base-model* and is a simple categorical classification with three dummy variables. The data set is split into 80 percent training set and 20 percent test set. The training set is fed to the neural network and then used to predict the test set. The output of the model is a vector of predicted probabilities that the input tweet is in favour of each of the three party divisions. The tweet is classified based on the most likely party. An illustration of the classification procedure is shown in figure 13.

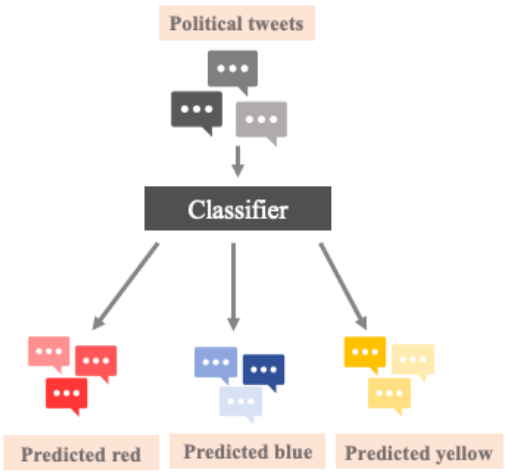

Figure 13: Illustration of the classification procedure with the Base-model.

2.8.2 Two step-model

As an expansion of the Base-model, the *Two step-model* was constructed. The Two step-model relies on the hypothesis that it would be easier to perform the classification in two steps, as the blue parties and the yellow parties are commonly considered closer in political views. The first step of the Two step-model classifies tweets as either in favour of red parties, "blue or yellow" or as uncertain. The second step classifies the "blue or yellow" into blue parties, yellow parties or uncertain. An illustration of how the Two step-model classifies tweets is shown in figure 14.

This model structure requires the data set to be labeled differently for training the different classifiers. Firstly, the data was scrambled and separated into a 20 percent test set and a 80 percent training set. The training data was then sorted after label since the training set for the first classification needed to have the same share of red tweets and blue/yellow tweets to prevent skewed training data. Hence, the training tweets for each division were divided in the relation 2:1:1 for red, blue, and yellow tweets. The tweets labeled as yellow or blue were relabeled as blue/yellow. The data set was scrambled and divided into 80 percent training data and 20 percent validation data.

To train the second classifier, the same training data was sorted and only the blue and the yellow tweets were considered. This subset of data was scrambled and the data was divided into 80 percent training and 20 percent validation. An illustration of this procedure is shown in figure 15.

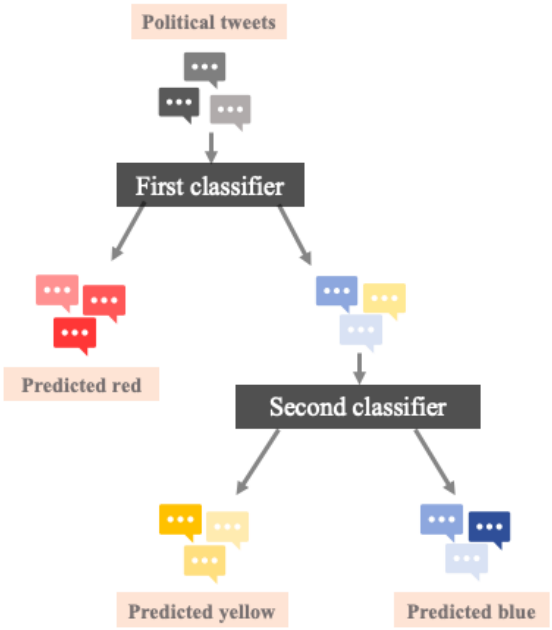

Figure 14: Illustration of the classification procedure with the Two step-model.

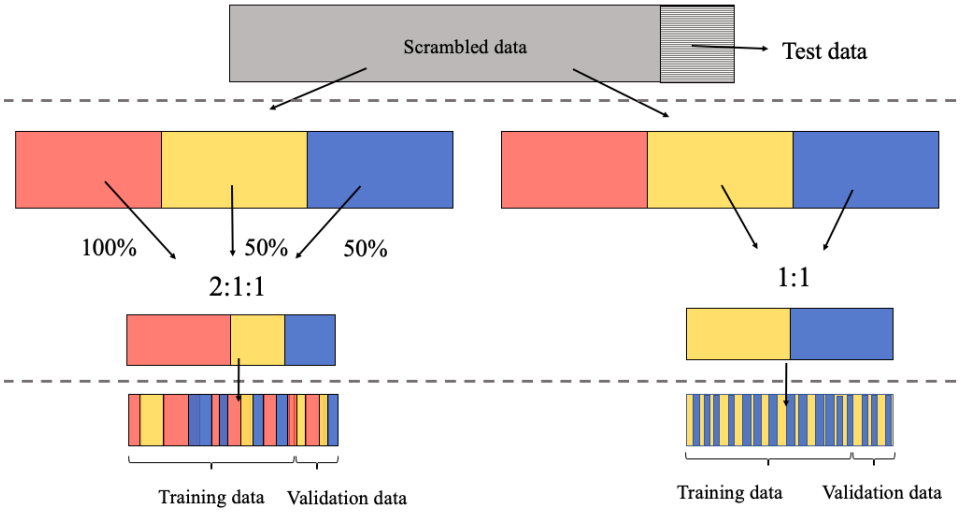

Figure 15: Data scrambling for Two step-model.

2.8.3 Managing uncertainty with a threshold

As mentioned previously, many tweets contain links and images and many of them are replies to other tweets. This means that a randomly selected tweet is likely to be taken out of context and may contain too little information when analyzed on its own. When manually scanning through 1000 tweets, it was discovered that only around 10 percent of these tweets contained enough political information to be classified in favour of a party. It would hence be wrong to classify all tweets that are scraped from Twitter since many of them would be completely irrelevant.

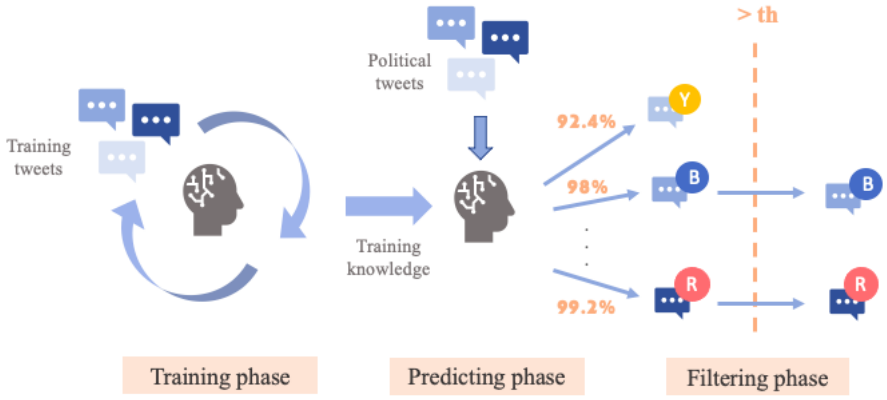

Figure 16: Filtering process.

To address this issue, a threshold is introduced. The model was trained on all tweets in the training data set and was then used to predict the opinion on the political data set from Twitter. The model generates probabilities that a tweet is in favour of the different party divisions. The threshold then works as a filtering function, and only allows tweets that the model is certain of to be classified. A higher threshold leads to more tweets being classified, but also means that tweets that the model is less certain of slips through. A scheme of this procedure is shown in figure 16.

2.8.4 Classifying with threads

The context of the tweets is hypothesised to add value to the classification. This hypothesis is tested by examining the impact of considering threads as a supplement to the classification. This is done by not only looking at each individual tweet but also examining the context of the tweet. The structure of the Twitter forum allows for debate in threads. Threads are created when

someone replies to a tweet, often with an opinion on the tweet in question. The tweet that generates the thread will from now on be referred to as the parent tweet. If someone tweets something positive about a specific party, it is likely to cause debate in the thread. People in favour of that same party are likely to react positively to that parent tweet and people opposing the party are likely to react in a negative manner. Sentiment analysis on the replies to an already classified tweet serves as a possibility to classify even further tweets. The relationship between threads and parent tweets is illustrated in figure 17. This is enabled by the scraping tool allowing gathering of both tweet ID and reply ID associated with each tweet, which is then used to group tweets into threads.

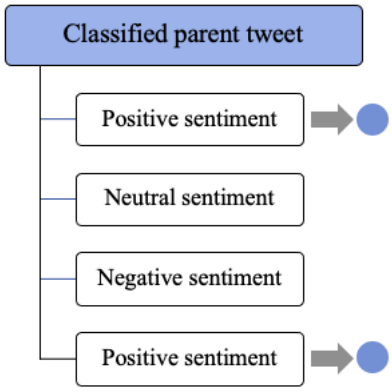

Figure 17: Classifying threads.

2.9 Alternative model

2.9.1 Lexicon-based opinion mining

A lexicon-based approach has been shown to outperform machine learning techniques in some cases of sentiment analysis (Hota, D. K. Sharma, and Verma 2021). Machine learning techniques are highly dependent on the set of training data, which is difficult to acquire. The VADER lexicon requires no training data and serves as an alternative to the deep learning techniques used in the *Base-model* and *Two step-model*. The Vader lexicon model, described earlier, is used to predict the sentiment of tweets containing mentions of parties or party leaders within the three divisions. In the political data set of 180 000 tweets, only the tweets mentioning leaders or parties were considered. Tweets that mentioned a division that was also labeled as positive by VADER, were classified as in favour of that division. Tweets labeled as negative were classified as against.

3 Evaluation of model designs

3.1 Data scrambling and model initialization

The result of the training is dependent on which part of the data set is chosen as the training set and which is chosen as the test set. There is also a stochastic variation in the result as a natural consequence of neural network optimization. To receive trustworthy results, the classification was carried out several times, where the data was scrambled each run. This results in different parts of the data ending up in the training set and test set each run. The models were pre-trained on tweets scraped from 2021, with the same keywords. The weights from the pre-training were used to initialize the neural network in each run to receive a more robust result.

3.2 Evaluation metrics

There are several ways to measure the performance of a model or model augmentation. The outputs of a binary classification model are true positives, false positives, true negatives and false negatives, and will be referred to as *tp, fp, tn, fn* from now on (Lipton, Elkan, and Narayanaswamy 2014). Two important measures are precision and recall. Precision is defined by the share of all positive predictions that were true positives. Recall is defined by the share of all true positives that were predicted positively. Precision and recall are expressed in (3) and (4). Recall and precision are numbers between zero and one, and are both high in a good model. Both these measures are important performance metrics in classification. Precision only considers the predicted positives, and a high precision score means that a large share of the predicted positives actually were true positives. However, the model might have neglected to classify many of the true positives, and would then have a high recall score. Consequently, it is vital to consider both measures when evaluating the performance of a model.

$$Precision = \frac{tp}{tp + fp} \tag{3}$$

$$Recall = \frac{tp}{tp + fn} \tag{4}$$

The F1-score is the harmonic mean of precision and recall and is commonly used to combine both measures into one comparable measure (ibid.). The F1-score is expressed in (5), and is a number between zero and one. A high F1 score means that the model is performing well. By calculating an F1-score for each class, the measure is applicable also for multi-labeled classification.

$$F1 = \frac{2tp}{2tp + fp + fn} \tag{5}$$

4 Prediction of opinion distribution

The predicted opinion distribution on Twitter was generated by classifying users based on the classification of their tweets. This was done by scraping 180 000 tweets in 2022 and the keywords used were svpol, all party leaders, and all parties, to try to cover as much of the political debate as possible.

4.1 Classifying users

To address the issue that 10 percent of the users on Twitter are generating the majority of the tweets, the tweets are grouped by user. For each user, the number of classified tweets in favour of each party was counted and the largest count of classified division was used to determine the party division sentiment of the user. If a user had the same amount of tweets in favour of several parties, the user was left unclassified. The procedure is illustrated in figure 18.

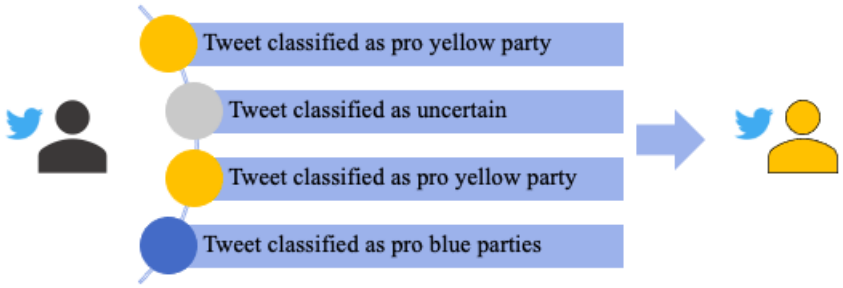

Figure 18: Classifying users.

5 Results and discussion

The models designed in the previous section were evaluated by letting them predict a test set, separated from the training data as described earlier. After the training process, each model predicted the test set and the results were compared to the true results. Following the evaluation, the models were used to predict the general political discussion. These results were evaluated based on the variance of the predictions.

5.1 Evaluation of model designs

The F1 measure is used to evaluate the model designs. The evaluation was carried out for all training sets, but only the F1 distribution for the training data based on the first set of keywords is presented as an illustrative example. The F1 distribution was calculated by training the models 40 times on 3600 tweets

and letting the model predict a randomly selected test set of 900 tweets. Recall, precision, and F1-score were calculated for each run with the sample size of 40 (Ganti 2023). The respective F1 distribution expressed in percentage for each class and model is presented in figure 19 - 21.

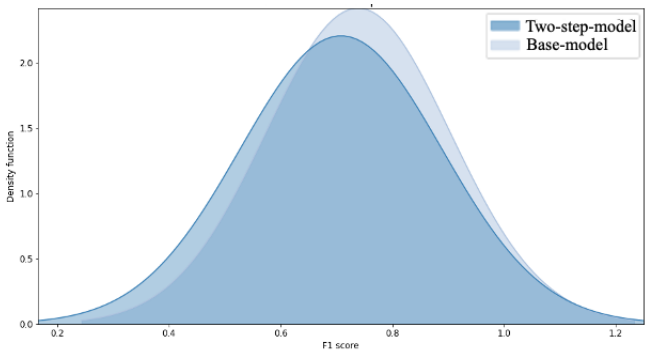

Figure 19: Distribution of F1-scores over blue parties.

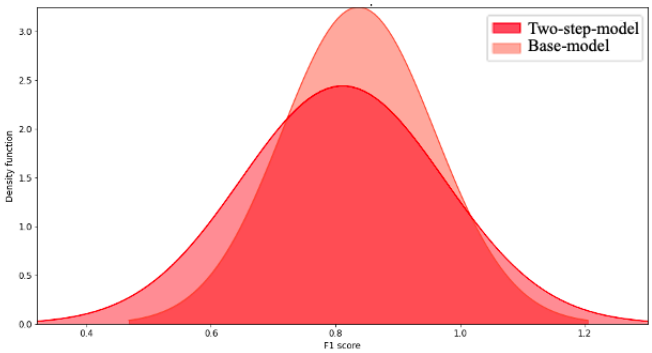

Figure 20: Distribution of F1-scores over red parties.

The results show that the Base-model surpasses the Two step-model in all classes based on the F1 score. The mean is both higher and the variance slightly lower for red parties and the yellow party. The most significant difference between the models appears in the division of blue parties. In general, both models are less accurate when predicting blue tweets. Although, the mean is higher and the variance is lower with the Base-model. That the score is subject to a higher variance for the Two step-model is likely due to the fact that the training and prediction is carried out two times each run. Since both classification steps are subject to a stochastic variation, the total variation is expected to increase.

The mean F1-score for the different classes and models is presented in figure 22.

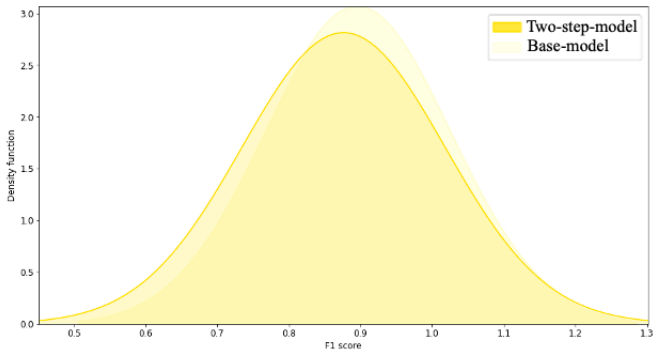

Figure 21: Distribution of F1-scores over the yellow party.

Considering the central limit theorem, the scores are assumed to follow a normal distribution. It is clear that both models are best at predicting tweets from the yellow party, followed by the red parties, and then the blue parties. The difference between the classes is most likely a consequence of the training data. The results also suggest that the Base-model is the better model at predicting the test set. However, the objective of this work is to predict the general opinion distribution on Twitter, and the best model is consequently the model that is best at predicting new unseen tweets. Both models are therefore used in the next section.

	Blue parties	**Red parties**	**Yellow party**
Base-model	69 ± 3%	80 ± 2%	84 ± 2%
Twostep-model	67 ± 4%	78 ± 2%	84 ± 2%

Figure 22: F1-scores.

5.1.1 Evaluation of threads

When examining the training data set scraped from keywords there are 90 tweets that belong to a thread created by a parent tweet in the same set. This shows that there is potential to classify more tweets if a parent tweet is already classified. As described earlier, this thread classification was carried out with the help of VADER. In the training data set, a total of 43 thread-tweets were classified with 35 of being classified correctly. This results in a total accuracy of 81 percent. In order to evaluate the performance of this classification for each category, the F1-score was calculated and is presented in figure 23.

	Blue parties	Red parties	Yellow party
Precision	100%	67%	100%
Recall	62%	100%	100%
F1	76%	80%	100%

Figure 23: Performance of VADER classification of tweets in threads.

This shows a potential to use threads as a way of classifying even more tweets, tweets that the neural network could not classify with certainty above the chosen threshold. Depending on the threshold, the neural network is on average able to classify around 600 tweets out of the 900 tweets in the test set. These new classified tweets represent an extra 7.5 percent of classified tweets, which could significantly impact the result when predicting unseen tweets.

5.1.2 Threshold analysis

The optimal threshold is a trade-off between the number of tweets classified and the accuracy of the prediction. The threshold plays an important role when generating an opinion distribution since many tweets need to be classified but with high accuracy in order to produce a reliable result. A threshold analysis was carried out to investigate the effect of the threshold on the accuracy and number of classified tweets. The results are shown below in figures 24 and 25.

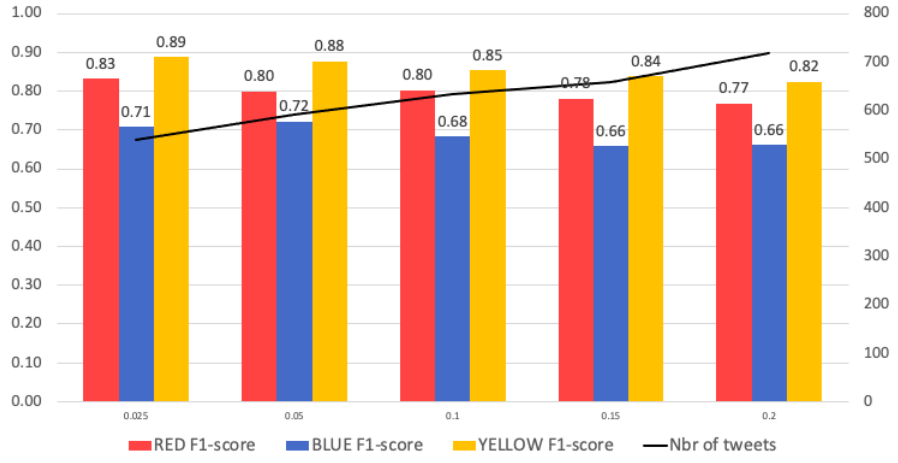

Figure 24: Threshold analysis Base-model.

The bars in the graph show how the F1-score for each class changes with the

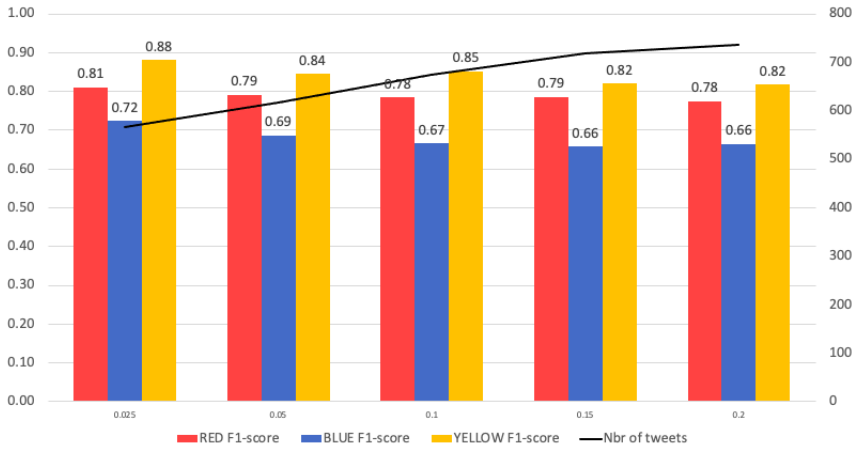

Figure 25: Threshold analysis Two step-model.

threshold along the x-axis. The trend line shows the number of tweets classified with each threshold. As expected, there is an increase in the number of tweets classified and a decrease in the F1-score as the threshold is lowered. The number of tweets classified has a relatively steady increase, and the F1-score has a steady decrease. The plots show no optimal value, but rather a trade-off between two important measures. Emphasis was placed on the F1-score since the accuracy of the classification needed to be high. Predicting general test data showed that a threshold of 0.025 resulted in an inadequate number of tweets being classified. A sufficient amount of tweets need to be classified, to minimize the impact of individual tweets on the opinion distribution. Since the loss in F1-score when increasing the threshold from 0.05 to 0.1 is quite marginal but achieves a substantial increase in tweets classified, it was decided to use 0.1 as the threshold in subsequent modeling.

5.2 Prediction of opinion distribution

When introducing a new set of tweets to the models, the aim is to accurately predict the party division sentiment of each tweet and to generate a distribution. Predicting an unseen set, generated differently than the training data, is a more difficult task for the model. Political data, consisting of 180 000 tweets scraped with the keyword *#svpol* or party or party leader mentioned in 2022, was predicted with the two different model layouts, Base-model and Two step-model. The predictions were done both with and without the augmented thread classification. Four different training data sets were used, "First", and "Second" which are annotated based on different keywords presented earlier, "Sentiment" which is annotated with the sentiment analyzer and "Mixed" which is a mixture containing equal parts of "First" and "Sentiment". The mixed data set was created to prevent the neural network from being trained on too specific tweets,

making the models perform badly when predicting general political discussion.

Deep-diving into the impact of different training sets shows how diverse the models are when trained on different training sets. The results for a random sample of tweets are shown in figure 26. It is clear that the models classify the tweets differently, and based on this sample entirely opposite. This analysis shows how crucial the training set is for the prediction of new unseen tweets. The difficulty of acquiring training data and the impact of the training data set again shows the difficulty of this problem.

As described in theory, a large variance is natural when fitting neural networks on complex problems and with small data sets. However, the variance encountered in the predictions is large, implying unreliable predictions. The same model with the same underlying training data, predicting the same unseen data can yield wildly different results. figure 27 describe the standard deviation from the model variations when predicting tweets from January only. The analysis was done for multiple months but January is used as an illustrative example.

Tweet	Model with first set of keywords	Model with first set of keywords
@eva_kristin77 Så rätt. Väntar på att gammal socialism med subventionerat mjölkpris ska upphöra. @Centerpartiet det står väl på agendan?	0.0945 **Right**	0.5905 **Red**
@katjanouch @socialdemokrat Usch, en korrupt socialist. Hon kan ta sina egna pengar och skänka till Ygeman och dom andra sosseprofessorerna.	0.2427 **Right**	0.8026 **Red**
@springhousese @socialdemokrat Okej. Vart har pengarna gått?	0.2559 **Right**	0.6948 **Red**
Fråga på nytt ångrar Ni Er socialdemokrater?	0.3520 **Right**	0.7402 **Red**
Nu när Löfven inte längre finns med i bilden, kan vi nu få kritisera islam eller ska vi fortsätta låta islamister styra debatten?	0.0429 **Right**	0.5616 **Red**
Det här är ett brutalt angrepp på äganderätten och livsmiljön. Allt för att miljöpartister socialdemokrater vänsterpartister och centerpartister i 08 området - som inte berörs av vindkraft - inte vill ha kärnkraft. Rösta bort dem i valet! #svpol #vindkraft https://t.co/OzFv7qzc7n	0.0426 **Right**	0.6886 **Red**

Figure 26: Illustration of the impact of the set of keywords

The results show that the Base-model, trained on the sentiment data set outperforms the other models in terms of average variance. This model shows the best results for the blue parties and the red parties. However, when only considering the yellow party, it seems as if the mixed data set is better. It is difficult to establish the reason for this. One hypothesis is that the sentiment set and

	Variance in users supporting blue parties	Variance in users supporting red parties	Variance in users supporting yellow party	Average variance
Base-model Threads				
First set	17%	16%	19%	18%
Sentiment	14%	10%	18%	14%
Mixed	28%	29%	9%	22%
Second set	18%	26%	26%	23%
Base-model No threads				
First set	19%	20%	21%	20%
Sentiment	15%	11%	24%	16%
Mixed	33%	32%	9%	25%
Second set	19%	30%	30%	26%
Two step-model Threads				
First set	21%	16%	18%	18%
Sentiment	15%	18%	20%	18%
Mixed	18%	27%	7%	17%
Second set	13%	18%	14%	15%
Two step-model No threads				
First set	25%	17%	23%	22%
Sentiment	18%	21%	23%	20%
Mixed	20%	29%	7%	19%
Second set	14%	21%	15%	17%

Figure 27: Variance in the predicted number of users in favour of each division with 20 runs for different variations of models.

the first keywords set complement each other better for the yellow party, e.g., by reducing the noise or amplifying patterns. For the red parties and the blue parties, the mixture might work in the opposite way.

For the Two step-model, the second training data set seems to perform the best in terms of average variance. Although, the same reasoning goes for the mixed data set for the yellow party. There is no clear winner between the Two step-model and the Base-model, but the lowest overall average variance is achieved with the sentiment training set and the Base-model. Generally, model prediction with threads augmentation achieved a lower variance than without.

Regardless of model, variation of training data, and with or without threads, the variance is still large. A common issue when experiencing high variance is that the training data is not sufficient, e.g., not fitting the data it will later be used to predict, too small, or too noisy. A small training data set could cause the model to overfit, which means that the model learns the noise of the training data. The model then performs seemingly well on the training set, whilst unreliable in predicting new tweets.

If the training data is too noisy, the model may be incapable of recognizing patterns, which would also lead to high variance. Low quality of the training data may be caused by incorrectly annotated tweets, or tweets annotated even though they in fact carry too little information. A too noisy data set would suggest that this way of gathering training data is not sufficient, and that manual annotation might be the better choice. To establish which underlying issues cause the bad performance, the variance is further analyzed.

5.2.1 Analyzing the variance

One approach to attempt to reduce that variance is to utilize the same simple layer structure but introduce a dropout layer to further exploit bagging. This adjustment is only reasonable to implement if the number of hidden neurons is also increased. The simple layer structure only contains 10 hidden neurons, and introducing a dropout layer would not make sense with this few hidden neurons. Hence, the number of neurons was increased to 100. The adjustment was implemented with the lowest variance model, Base-model with threads trained on the Sentiment-data set. Opposite of the desired effect, the average variance instead increases as can be seen in figure 28.

Base-model Threads	Variance in users supporting blue parties	Variance in users supporting red parties	Variance in users supporting yellow party	Average variance
Sentiment	14%	10%	18%	14%
Sentiment with droplayer	14%	22%	21%	19%

Figure 28: Impact on variance when introducing a drop-out layer

Another hypothesis is that the large variance is due to the training data set being too small, which causes the model to overfit. To examine this, the training data set was halved and the results are shown in figure 29. When the data set was halved, the model was unable to classify any tweets as blue. This could imply that the model would benefit from a larger data set than the original one to increase the reliability of the blue classifications. On the contrary, the variance in the classification of red and yellow users was reduced by halving the data set, marginally for red but more so for yellow users. The difference in the number of users classified as each party division is shown in figure 30. Many of the users previously classified as blue are now classified as yellow. This is probably the reason for the reduction in variance, not classifying any as blue implies an easier guess and thus, a reduction in variance. Therefore, the reduction of variance, in this case, is not an improvement in model performance.

Base-model Threads	Variance in users supporting blue parties	Variance in users supporting red parties	Variance in users supporting yellow party	Average variance
Sentiment	14%	10%	18%	14%
Sentiment halved-dataset	N/A	9%	9%	

Figure 29: Impact on variance when halving the training data set

Base-model Threads	Average number of users classified as Blue	Average number of users classified as Red	Average number of users classified as Yellow
Sentiment	1165	1343	1283
Sentiment halved-dataset	0	1244	3093

Figure 30: Impact on number classified from halving the data set

As seen in figure 27, including threads contributes to lower variance. Due to the lexicon-based classification being deterministic, including threads is expected to lower the variance. The evaluation of thread classification was optimistic and showed potential. However, an issue with the current methodology is that not all threads are analyzed. Only threads of classified parent tweets are considered, which potentially simply leverages the party division with most classified tweets. Thus, utilizing the current form of thread classification lowers the variance, but might skew the resulting distribution.

5.2.2 Opinion distribution with lexicon-based model

When using sentiment analysis on tweets where party leaders and parties are mentioned, it is observed that the red parties are most frequently talked about. The results are shown in figure 31. In total, 9399 tweets were classified as either in favour of or against a party division. These tweets are generated by 5307 users, of which 51 percent tweet the most about red parties, 38 percent tweet about blue parties, and 12 percent tweet about the yellow party. Overall, tweets with a negative sentiment outnumber those with positive sentiment for all party divisions.

Since it is generally not possible to draw conclusions about party affiliation based on negative sentiment, one approach to yield an opinion distribution is to base it only on the users with a positive sentiment. To ensure that the distribution concerning users in favour of the different divisions adds up to 100 percent, the results are normalized by the total number of users in favour of a division. The results are shown in figure 32. Naturally, the distribution of users against

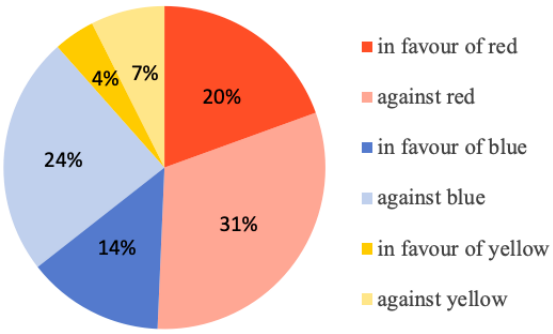

Figure 31: Opinion distribution with VADER lexicon-based model in January.

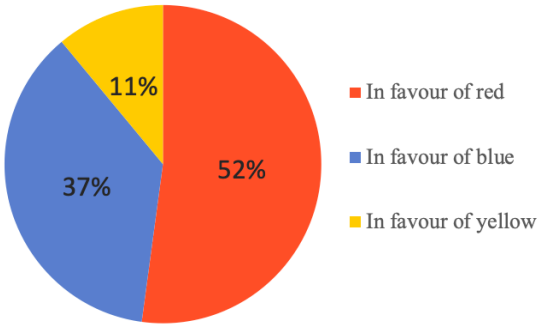

Figure 32: Positive opinion distribution with VADER lexicon-based model in January.

a division also carries significant information about the opinion distribution on Twitter. Although, these results are more ambiguous and more difficult to use when creating an opinion distribution.

To get a sense of how to interpret the results, it is interesting to further examine the share of users classified as either in favour of or against the respective party divisions. The results are shown in figure 33. These results suggest that information about opinion distribution is also captured in the share of users against a division. Equivalently to how the positive opinion distribution is generated, a negative distribution can be created. Even though one can not establish which party the negative user support, the result can be used as a variable in a regression analysis. The regression analysis aims to explore if there is a relation between results from opinion polls and the predicted distribution of users in favour of and against.

The analysis showed that there was no significant relation between the opinion polls and the predicted distribution with the VADER sentiment model. This

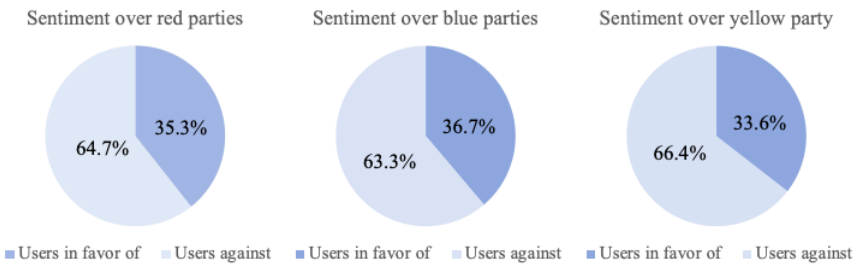

Figure 33: Distribution of positive and negative users by party division.

can depend on several factors, e.g., that the way of scraping based on mentions of parties is not sufficient to comprehensively determine the opinion distribution or overarching factors discussed in the next section.

5.3 Method limitations

While the fault could lie in how the data is gathered or other limitations previously discussed, the problem could also lie entirely in the proposed method. The Swedish political landscape is not nearly as polarized as in the US (Oscarsson et al. 2021). Perhaps the political discussion climate is not polarized enough in Sweden to successfully use the classifying methods that worked in US and Italy. The US two party-system also simplifies the problem significantly. Meanwhile, in Sweden, there are many parties that try to have their own differentiated political ideology, and collaboration between parties within different divisions occurs. Therefore, sometimes the split proposed in this work may not be accurate and cause misclassification.

In addition, there is no guarantee that conversations on Twitter reflect the overall political views of the general population. The proposed method includes using regression as a means to weigh the predictions of each month to fit the distributions of opinion polls. The purpose of this is to map the opinion poll and the Twitter distribution with weights since the hypothesis is that these do not perfectly align. Although, the problem could lie in that Twitter is not at all representative of the general political landscape and no opinion trends can be discovered on the platform that also translates to the opinion polls. For example, the number of users that decides to tweet one month might depend on a certain political event and has the potential to severely impact the distribution. This methodology relies on the assumption that Twitter users tweet regularly to capture changes in distribution.

Many of the conversations on Twitter are heavily context-based. Often a picture or a link to a news article is the conversation starter, and the replies are in regard to what the picture or link is about. The thread classification was an

attempt at taking more of the context into account when classifying but is not comprehensively capturing all of the contexts. This is likely one of the limitations affecting the results of the current method.

5.4 Comparison to opinion polls

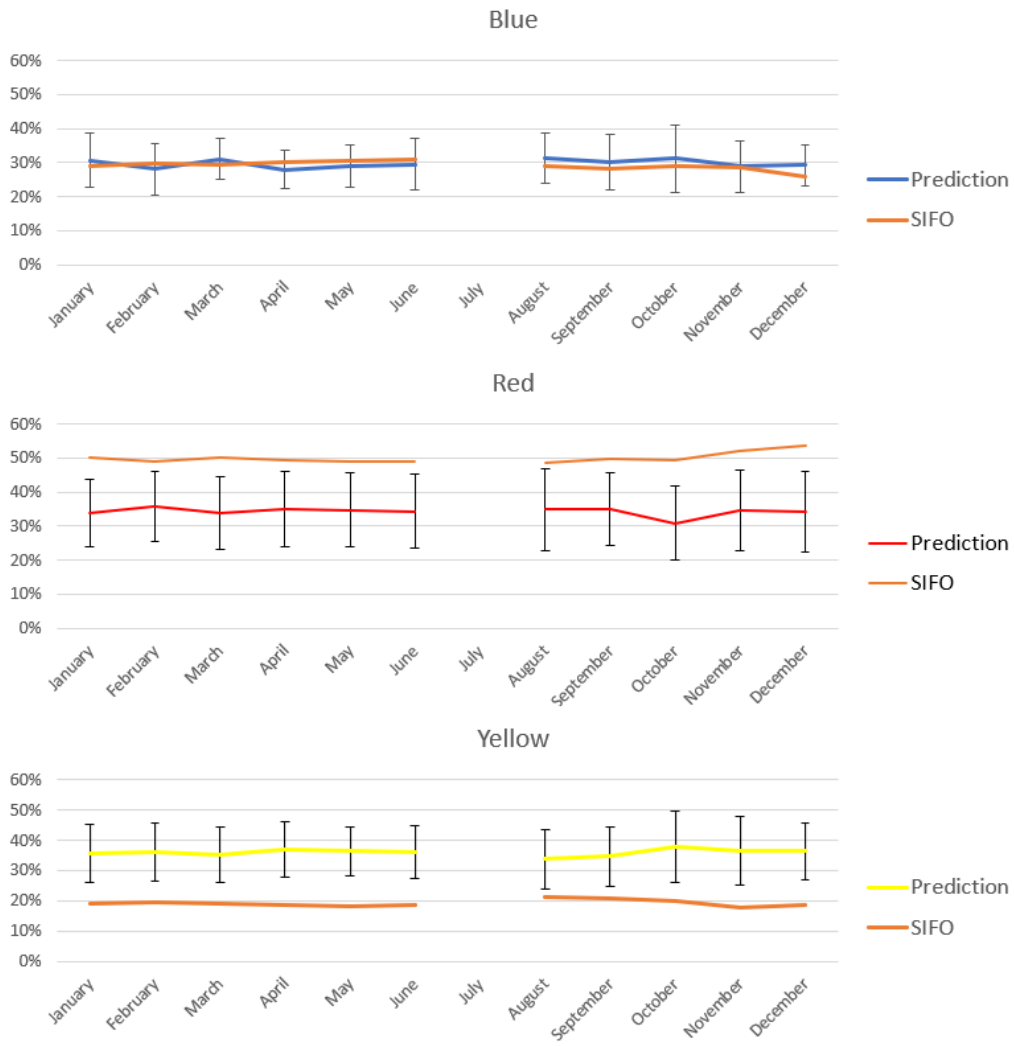

Figure 34: Opinion distributions with standard deviations predicted with the Base-model, trained on the sentiment set and with thread augmentation.

Due to the high variance observed when predicting with the different neural network models, there is no surprise that performing regression on the results

shows no significant relationship between the predicted opinion distribution and the opinion polls. Since the predictions are so diverse for each run, there is no point in trying to find a regression that fits the predictions to the opinion polls. As illustrated in figure 34, the variability is too large when dealing with such small changes in the true opinion distribution each month. The figure shows the mean distribution predicted with the Base-model, trained on the sentiment set and with thread augmentation since this was the model with the lowest variance. July is not predicted since there is no opinion poll data for this month to compare the predictions with.

The result show that the predicted results in November could for example differ between 23% and 47% within one standard deviation for red parties. A regression would then be based on one random run, and even if this regression would show significance, the results would not be reliable. The next run would generate completely different results.

In the case when using the deterministic lexicon-based model to predict, there is no issue with variance. However, a regression analysis showed that there was no significant correlation between the opinion polls and the predicted distributions using the lexicon-based model.

6 Conclusion

This work attempted to predict political opinion polls using Twitter data. When building the classification model, the neural network was quite successful in predicting tweets collected based on specific keywords. Classifying tweets regarding Sverigedemokraterna achieved the highest F1-score whilst the tweets by parties affiliated with Moderaterna were more difficult to predict. Moreover, considering the context in the classification by using the structure of tweet threads proved a rather effective method.

The prediction of unseen tweets, resulting in an opinion distribution proved to be a more difficult task. Many different model variations were created in an attempt to reduce the prediction variance. The model with a three-way prediction, trained on data annotated with a sentiment analyzer and considering the context with threads achieved the lowest variance, however, still too high to make any reliable predictions.

The underlying issue of the prediction variance was investigated, but no definite answer was found. The investigation is pointing towards the need for more training data of higher quality. However, the problem could also lie in the proposed method. To accurately predict opinion polls might not be achieved even with perfect training data due to the structure of Twitter, or the political landscape. The context of the tweets might not be considered enough to classify tweets in affiliation with a party, Swedish politics might not be polarized enough

to accurately make sense of vague tweets or the opinions expressed on Twitter may not at all be representative of the general population.

7 Ideas for future research

Most feature extraction algorithms are typically designed for English-language data, and it can be less efficient to apply these algorithms directly to Swedish text. One solution is to first translate the Swedish text into English and then apply the English feature extraction algorithms to the translated text. It is important to note, however, that there may be differences in the grammatical structure and vocabulary of Swedish and English that can affect the accuracy of feature extraction. Feature extraction algorithms that would be valuable to experiment with are BERT and Word2Vec. Another possibility is using the large language model ChatGPT for sentiment analysis.

In this work, the classified tweets were linked to the user that posted them to create the distribution. Another possible approach is to deepen the analysis by considering the number of followers the user has or how many times the tweet has been viewed to get a sense of the reach.

Furthermore, Twitter is only one of many possible sources of data, subsequent work could explore the possibility of doing similar analysis on other outlets of opinions. For example, newspapers or Facebook could be a source of opinion data that is more representative. Another alternative is to combine multiple sources to gather more of the political discussions.

This work also discovered that a majority of the tweets expressing an opinion were negative. Perhaps a better way of predicting opinion would be to use negative tweets to determine the distribution. However, utilizing negative tweets in a three-party system poses a difficult challenge since negative views toward one party do not necessarily reflect a person's voting intention. Another potential possibility is to use the negative opinion information to discover trends that can be exploited to predict changes in opinion.

To further explore where the main limitations lie within this work, it would be interesting to run the algorithm on political data from the US.

8 Bibliography

References

Belcastro, Loris et al. (2020). "Learning political polarization on social media using neural networks". In: *IEEE Access* 8, pp. 47177–47187.

Börjeson, Love (Feb. 2020). *KB Tillgängliggör Kraftfulla modeller för språkförståelse*. URL: https://www.kb.se/samverkan-och-utveckling/nytt-fran-kb/nyheter-samverkan-och-utveckling/2020-02-04-kb-tillgangliggor-kraftfulla-modeller-for-sprakforstaelse.html.

Brownlee, Jason (Aug. 2019). *What are word embeddings for text?* URL: https://machinelearningmastery.com/what-are-word-embeddings/.

Chollet, François et al. (2015). *Keras*. https://keras.io.

Dalle, Peter (Nov. 2021). *Svensktext/stoppord.csv at master · Peterdalle/Svensktext*. URL: https://github.com/peterdalle/svensktext/blob/master/stoppord/stoppord.csv.

Delua, Julianna (Mar. 2021). *Supervised vs. unsupervised learning: What's the difference?* URL: https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning.

Eger, Steffen, Paul Youssef, and Iryna Gurevych (2019). "Is it time to swish? Comparing deep learning activation functions across NLP tasks". In: *arXiv preprint arXiv:1901.02671*.

Ganti, Akhilesh (Mar. 2023). *Central limit theorem (CLT): Definition and key characteristics*. URL: https://www.investopedia.com/terms/c/central_limit_theorem.asp.

Goularas, Dionysis and Sani Kamis (2019). "Evaluation of deep learning techniques in sentiment analysis from twitter data". In: *2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML)*. IEEE, pp. 12–17.

Gu, Weiwei et al. (June 2021). *Principled approach to the selection of the embedding dimension of networks*. URL: https://www.nature.com/articles/s41467-021-23795-5.

Hidayatullah, Ahmad Fathan, Siwi Cahyaningtyas, and Anisa Miladya Hakim (Nov. 2020). *Sentiment Analysis on Twitter using Neural Network: Indonesian Presidential Election 2019 Dataset*. URL: https://iopscience.iop.org/article/10.1088/1757-899X/1077/1/012001.

Hinton, Geoffrey et al. (2012). *Lecture 10a Why it helps to combine models*. URL: bit.ly/3LTbJxw.

Hota, H.S., Dinesh K. Sharma, and Nilesh Verma (May 2021). *Lexicon-based sentiment analysis using Twitter data: A case of covid-19 outbreak in India and abroad*. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8989068/.

Hutto, Clayton and Eric Gilbert (2014). "Vader: A parsimonious rule-based model for sentiment analysis of social media text". In: *Proceedings of the international AAAI conference on web and social media*. Vol. 8. 1, pp. 216–225.

Jain, Anil K, Jianchang Mao, and K Moidin Mohiuddin (1996). "Artificial neural networks: A tutorial". In: *Computer* 29.3, pp. 31–44.

JustAnotherArchivist (2018). *snscrape: A social networking service scraper*. URL: https://github.com/JustAnotherArchivist/snscrape.

Kantar (2023). *Väljarbarometern*. URL: https://www.kantarpublic.com/se/undersokningar-rapporter/valjarbarometern.

Kavlakoglu, Eda (Nov. 2020). *NLP vs. NLU vs. NLG: The differences between three Natural Language Processing Concepts*. URL: https://www.ibm.com/blog/nlp-vs-nlu-vs-nlg-the-differences-between-three-natural-language-processing-concepts/.

Kingma, Diederik P. and Jimmy Ba (2017). *Adam: A Method for Stochastic Optimization*. arXiv: 1412.6980 [cs.LG].

Kukreja, Harsh et al. (2016). "An introduction to artificial neural network". In: *Int J Adv Res Innov Ideas Educ* 1, pp. 27–30.

Lipton, Zachary Chase, Charles Elkan, and Balakrishnan Narayanaswamy (May 2014). *Thresholding classifiers to maximize F1 score*. URL: https://arxiv.org/abs/1402.1892.

Mahmood, Hamza (Nov. 2018). *Softmax function, simplified*. URL: https://towardsdatascience.com/softmax-function-simplified-714068bf8156.

Malmsten, Martin, Love Börjeson, and Chris Haffenden (2020). *Playing with Words at the National Library of Sweden – Making a Swedish BERT*. arXiv: 2007.01658 [cs.CL].

Nielson, Michael (Dec. 2020). *3.1: The cross-entropy cost function*. URL: https://eng.libretexts.org/Bookshelves/Computer_Science/Applied_Programming/Book3A_Neural_Networks_and_Deep_Learning_(Nielsen)/033A_Improving_the_way_neural_networks_learn/3.013A_The_cross-entropy_cost_function.

Nilsson, Thomas (Dec. 2019). *Det twittrade svenskarna om under 2019*. URL: https://www.resume.se/insikt/trend/det-twittrade-svenskarna-om-under-2019/.

Oscarsson, Henrik et al. (Oct. 2021). *Polarisering i sverige – demokratirådets rapport 2021*. URL: https://www.sns.se/artiklar/demokratiradets-rapport-2021-polarisering-i-sverige-2/.

Prechelt, Lutz (1998). "Early Stopping - But When?" In: *Neural Networks: Tricks of the Trade*. Ed. by Genevieve B. Orr and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 55–69. ISBN: 978-3-540-49430-0. DOI: 10.1007/3-540-49430-8_3. URL: https://doi.org/10.1007/3-540-49430-8_3.

Qin, Zhenyue, Dongwoo Kim, and Tom Gedeon (2019). "Rethinking softmax with cross-entropy: Neural network classifier as mutual information estimator". In: *arXiv preprint arXiv:1911.10688*.

Sathyanarayana, Shashi (July 2014). "A Gentle Introduction to Backpropagation". In: *Numeric Insight, Inc Whitepaper*.

Saxena, Sawan (Feb. 2021). *Understanding embedding layer in Keras*. URL: https://medium.com/analytics-vidhya/understanding-embedding-layer-in-keras-bbe3ff1327ce.

Severyn, Aliaksei and Alessandro Moschitti (Aug. 2015). *Twitter Sentiment Analysis with Deep Convolutional Neural Networks*. URL: bit.ly/3LLyIdN.

Sharma, Sagar, Simone Sharma, and Anidhya Athaiya (2017). "Activation functions in neural networks". In: *Towards Data Sci* 6.12, pp. 310–316.

Uzila, Albers (Nov. 2022). *All you need to know about bag of words and word2vec-text feature extraction*. URL: https://towardsdatascience.com/all-you-need-

to-know-about-bag-of-words-and-word2vec-text-feature-extraction-e386d9ed84aa.

Master's Theses in Mathematical Sciences 2023:E36
ISSN 1404-6342

LUTFMS-3477-2023

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/