M.Sc. Thesis
Master of Science in Engineering

Linguistic and Political Differences in Large Online Social Media Discourse

Jamie Neubert Pedersen (s113431)
Valentin Ibanez (s122994)

Kongens Lyngby
June 22, 2018

DTU Compute
Department of Applied Mathematics and Computer Science

DTU

Preface

This thesis was submitted as a final requirement for obtaining the degree of Master of Science in Engineering (M.Sc.Eng / Cand.polyt.) at the Technical University of Denmark (DTU). The thesis was prepared at the Department of Applied Mathematics and Computer Science at DTU. It was carried out from January to June, 2018. The project was mainly supervised by Sune Lehmann (Associate Professor, DTU, DTU Compute).

We want to thank Sune Lehmann for his supervision, motivating discussions and for allowing us to work in the offices of his research group. We would also like to thank Amalie Jensen, Sofie Holck Thorhauge and our mothers, brothers and fathers for their support.

Contents

Abstract

Unsupervised and data-driven language understanding methods makes it possible to quantify large amounts of unstructured natural language corpora. We collect and construct a dataset of 1848 Danish political actors along with their tweet histories, totalling 584, 156 tweets. We find linguistic differences across political groups and also demonstrate the diachronic nature of language, by showing the meaning of words shifting over time. Our dataset is used to construct word embeddings of political, semantic and syntactic similarities, which we use in machine learning algorithms to classify the political partisanship of text. We demonstrate that combining, context specific semantic and syntactic information with class representative word weights results in better accuracy, when classifying political affiliation of tweets from political actors. Through these analyses of the constructed dataset, we show the viability of using written text to find political patterns and linguistic differences. We demonstrate how these can be used for feature engineering in supervised machine learning algorithms and the possibility of classifying text's political partisanship.

Chapter 1

Introduction

Political discourse on social media has been a major topic in recent years. Social media has been a significant component in politics internationally and has given rise to terms such as fake news, troll factories and voter manipulation. Throughout the years social media has increasingly become a platform for politicians to influence voters. This is also true for Denmark, as the data presented in this thesis shows, the amount of political discourse generated by politicians on Twitter has increased in the last decade. This vast amount of publicly available information, can render it difficult for journalists and voters to navigate through the content. The authors of content and their agendas might not always be apparent in the world of digital media with the anonymity it entails. However, the increased amount of publicly available information also allows data analysts to construct a pointillist representation of patterns in political discourse.

These pointilist representations are however, not restricted to political discourse. In digital systems, the traces *we* leave are plentiful. All our interactions with digital systems can be logged, which allow capturing otherwise ephemeral information. This creates the potential to create a very fine-grained pointillist representation of us as well. A representation that can reveal a lot about our personality, what we agree with, our preferences and opinions towards different topics. Today, we are using social media more than ever, which offers information about our social interactions and communication patterns. There are many ways to communicate, both offline and online, on social media we communicate in a variety of structured and unstructured ways. We "like", company pages, brands, organizations, each others "status updates" and what events we are attending. We write said updates and "mention" our friends, topics and people we want to direct attention to. We send still pictures, or animated ones, but most of all we communicate in written natural language.

Natural languages. The languages that have evolved naturally without conscious planning are, in contrast to constructed languages such as programming languages, irregular. That we communicate in these irregular natural languages such as *English* and *Danish* is a testimony to the human brain's ability to interpret ambiguous information. The concept that natural languages evolve through time is known in linguistics as natural languages being diachronic.

The diachronic nature of languages imply that the meaning of words shift through time. On one scale, all our utterances are independent, we are unique in what we think and how we

perceive the world. The seeming independence and uniqueness of our utterances, is shown to exhibit patterns in both the temporal (diachronic) and the spatial scale. Changing these scales can reveal otherwise hidden patterns. Patterns that can be analyzed quantitatively and generalized across a group of people or over time. We show in this thesis the diachronic nature of language and the patterns of language use that belong to groups of individuals.

Even though natural language does not follow a strict and consistent structure, hence it is irregular, we can map aforementioned opinions about different topics. Opinions about topics reflects a political standpoint towards that topic. Some topics are more partisan than others but nevertheless there is political structure to communication.

Thesis goals. Our thesis addresses politics on Twitter through a range of analyses. Focusing on finding political patterns in natural language in a large dataset of unstructured text, we will surface some of the patterns underlying the way we communicate and what those can reveal about our political standpoints. We conduct these analyses on a dataset that we collected during the projects period. The dataset used is a collection of Twitter profiles, and their corresponding tweet history, of political candidates for the 2017 local election in Denmark, and the politicians in the Danish Parliament elected in 2015.

Contributions. The main contributions of this thesis are: showing that political discourse on social media, such as Twitter, carries signal that can be extracted; using different techniques to gain insights, into what words and phrases are considered partisan; and presenting the results in the context of Danish political discourse on Twitter.

We do not have domain expertise in the social sciences and we will refrain from concluding any causal relationships, as it is easy to mistake correlation for causality when we do not have deep knowledge in the area. We hope that our exploratory analysis of this subject can ease comprehension, of large corpora from social media discourse by political actors. Thereby, leading other researchers, with more domain expertise, to use this analysis to gain insights or prove hypotheses that previously only were backed by small qualitative studies or intuition.

1.1 Twitter, Politics and NLP

Twitter, politics and natural language goes hand in hand. Politicians use media to reach the greater public with their opinions. Twitter is a medium where people can express their thoughts using written text. Written text is often prose written in natural languages. We will in this section, highlight research that shows the viability of using data from Twitter, to find patterns in large scale unstructured natural language produced by politicians.

Twitter in politics. Numerous studies have looked at extracting valuable information from Twitter, and a comprehensive review of 115 studies regarding political use of Twitter highlights the political usage and what patterns there were across these studies [1]. The studies included in [1], were strictly about the use of Twitter by politicians, publics or comments regarding political issues and campaigns.

Politicians mainly use Twitter for broadcasting their own agendas and opinions [2, 3], and we can use their broadcasting to collect information about their use of language.

Political classification. In recent years much effort has been made to construct algorithms that can classify political opinions based on digital content from social media. This has been done using a variety of features, such as *hashtags*[1] [4], user behaviour and social network structure [5], *retweets*[2] [6, 7], *user-mentions*[3] [7], user votes [8] and of course text content [9, 5].

Data-driven language understanding. Quantifying and structuring natural language has a long tradition in linguistics, and is referred to as natural language processing (NLP). NLP deals with tasks such as part-of-speech (POS) tagging, text clustering, sentiment analysis, word-sense disambiguation, etc. Using text as data has the problem of ambiguity, as utterances in natural language allow for different interpretations. Nevertheless there is rich information to be mined from text, and it is possible to extract it. Capturing this rich information by hand, structuring it and putting it in knowledge databases is a time consuming and expensive process. To dig into the plethora of unlabelled and unstructured data there is in the world, we must employ techniques that reach beyond manually labelled and encoded knowledge. In [10] the distinction is coined as "open- and closed-vocabulary" approaches. Closed- and open-vocabulary approaches are different in how they construct the vocabularies, i.e. a lexicon of language. A *closed-vocabulary* is defined *a priori* manually which is useful in small datasets where it is hard to infer any meaningful structure. A *open-vocabulary* approach is defined by automatically deriving the vocabulary from the dataset, i.e. it is data-driven. The vocabularies can include information about the properties that are of interest to a specific analysis e.g. the linguistic features, semantics of words and phrases and categories they belong to. Again the closed-vocabulary would be manually created by experts, and the open-vocabulary would infer the properties from the data.

Using a traditional closed-vocabulary approach hinders one to find new characteristics. Whereas, using an open-vocabulary approach instead of the closed-vocabulary, where a fixed lexicon is defined a priori, allows one to find unexpected results [10]. Schwartz et al. analyzed a dataset of 15.4 million Facebook messages from 75,000 volunteers and found statistically significant characteristics of language use differences across several attributes such as gender, personality traits, age, etc. They find distinctive words and topics as functions of known attributes such as gender, age, location and psychological characteristics. The research by Schwartz et al. is in line with research published by Monroe et al. where they find distinctive words as a function of topic and political affiliation [11].

1.2 Motivation

The social sciences seek to understand and explain human behaviour, culture and society. As more of the interactions are conducted online, digital tools are needed for studying the complex dynamics of society, communication, political discourse, etc. We think that quantitative and qualitative approaches to understanding the world can be brought together. Our research methodology, reflects this by being a complex mixed method, to analyze qualitative data in a quantitative way. Putting it simply, one could say that it is often the case, that in works seeking to gain *insights* about phenomenons, an emphasis is placed on

[1] Twitter specific tags that can be applied to a tweet to connect the tweet to a topic or to emphasize a message

[2] Twitter specific action where a user re-posts a message that another user has posted

[3] Twitter specific mentions used to reference another Twitter user

qualitative data and results, neglecting data-driven approaches. This is in contrast to the research focused on *predicting* outcomes, where a data-driven approach is the norm, but here neglecting qualitative assessment of result's validity. We think that it may serve the works, seeking to gain insights, to appreciate the power of a data-driven approach and vice versa. This is a key motivation for showing the quantitative approaches, to gaining insights about the political landscape in large datasets of online discourse.

1.3 Central Research Question

As motivated by our hypothesis that there is a need for tools to inspect large corpora of text, since a lot of discourse happens online and that the social sciences need better tools to disseminate these large corpora, we formulate our research question as follows:

> How can we use natural language to surface patterns of linguistic differences in large scale corpora of unstructured text and how can we use these patterns to find and visualize political partisanship in language use?

The keywords here are *unstructured text*, *political partisanship* and *visualize*. Unstructured text is defined as all natural language communication, since it is not encoded in a structured format i.e. it is qualitative, and we seek to surface these patterns and structure in the unstructured text. Political partisanship refers to the political leaning, affiliation or bias of a text document. We visualize the features that we extract from our analyses, so it is easy to interpret the large scale structure of the text corpus.

1.4 Methods

In this thesis, the approach to gaining insights is through quantitative measures and models. The research conducted goes through different disciplines, from politology and social sciences to applied AI, showing the applications of using *learning algorithms* for understanding large unstructured corpora of natural language. We focus on conducting the analyses in such a way that the signal captured by the models can be interpreted and explained, which allows inspection by domain experts to test the validity of the results. Interpretability, is therefore a key driver and a great emphasis is put on working with methods that produce interpretable features and results.

1.4.1 Data Collection

The data used in this thesis is for the most part the tweet history of the Danish politicians as well as the politicians' party affiliation. The dataset has been collected for this thesis and was collected in the time period 25th of Januaray to the 15th of May. Another dataset that is used is an extract of all the Danish Wikipedia articles which is used in training the *word embeddings* in chapter 4.

1.4.2 Model Testing

We will distinguish between *training*, *validation* and *test* datasets in the modelling process, with the definitions of the terms following here. The training dataset is the sample used to fit the models, i.e. the dataset that the learning algorithms are trained on. The validation

dataset is the sample used to fine-tune the hyper-parameters of the model by allowing the model to validate the trained configurations on this previously unseen data. As the model will fine tune according to this dataset it can no longer be thought of as unbiased or out-of-sample. Finally we have the test dataset which will be the sample of data used to provide a final unbiased evaluation of the model that was trained. It is unbiased as the forecast will be *out-of-sample* and thus gives greater confidence to the performance of the model. This 3-step process reduces signal leakage from the training to the testing of the models, and the model should not be fitted again to yield higher results on the test set as that reintroduces the problem that was alleviated by this process. We split the original dataset by holding 20% for test and 80% for train and validation. The train and validation datasets are randomly split between runs of training to avoid overfitting and make the model generalize better, which is known as cross-validation.

1.4.3 Qualitative measurement of results

Qualitative measures are needed for certain aspects of this study as there are no quantitative measures for classifying political leaning of language insofar we know. To guide in this process Frederik Georg Hjorth has provided his assistance by validating the results with his intuition based on his experience in working with politology.

1.5 Report Organization

The thesis is divided into self-contained chapters that each revolves around a specific analysis of the dataset we are using. Our thesis spans from *Data collection* to *Data mining* and *Data visualization*, and we will in the following chapters present our collection, analysis and results. An outline of the chapters is provided in the following paragraph.

Chapters outline. Chapter 2 describes the dataset. Detailing how it is collected, what the properties are, it's basic characteristics and descriptive statistics. Chapter 3 details our analysis of word usage across political parties, which shows models for processing natural language to find linguistic differences in the text, to select features that are salient for further analysis. The analysis provides face-valid results, showing linguistic differences in political discourse analysis. Furthermore, the analysis of the political leaning in a temporal perspective, shows the diachronic nature of language and the limitations of only considering a synchronic view of language. Chapter 4 criticises the bag of words approach often used in natural language processing, and presents to the reader a theoretically founded hypothesis of statistical semantics. This paves the way for learning word vectors via word distributions in documents. The chapter presents the state of the art on learning these distributional semantics with word embeddings and compares the approaches, by constructing word embeddings from our dataset and detailing a qualitative analysis of the results. Chapter 5 brings the previous chapters together, by constructing different models for classifying the political leaning of text. Chapter 6 discusses the overarching results and possibilities for future work.

Chapter 2

Dataset & Introductory Analysis

There are several issues that are important for successful data mining: the type of data, the quality of the data and which preprocessing steps are used. It is thus important to describe the steps that have been taken in the process. In this chapter, we introduce the dataset that we have collected. We explain in detail our collection process, which span multiple sources and methods. We also describe how the data from multiple sources was brought together into one cohesive dataset. By describing our data collection and construction process, we also show the possible implications and bias the might contain. We will display the dataset's size and properties, however not all of these properties are used in the research presented. We will describe which properties are selected, for the various analyses in subsequent chapters. We also introduce a preliminary statistical analysis of the dataset, showing the distribution of data over time and across the different actors. From the statistical analysis, we can deduce the data's characteristics, qualities and nuances. Such a descriptive analysis is important, to show what considerations we have to take when using this data in further analyses. Furthermore, we present the preprocessing pipeline, detailing how we parse the text into elements that can be encoded. Finally we summarize the content of the chapter, highlighting the most important aspects of the dataset and the preprocessing.

2.1 Data Collection and Construction

We start by introducing what issues might arise, when performing data collection and data mining. Then we describe how the data was collected from Twitter and cross-referenced with other sources, to construct a dataset with properties that are useful for our analyses.

We have created a dataset consisting of Twitter profiles, and their corresponding tweet history[1], of political candidates for the Danish local elections of 2017 as well as the politicians in the Danish Parliament, as of January 25, 2018. The data we have collected stems from multiple sources. Our analyses presented in this research, are for the most part based on the tweet history of the Twitter profiles. To label the Twitter profiles to a political party, we

[1] A history of status updates a profile has "tweeted" (published). The tweet histories collected are not necessarily the complete set of their tweets because of restrictions in the Twitter API.

collected data from Altinget[2] and the website of the Danish Parliament. We have collected data up until May 26, this has however been an iterative process, due to rate limits in the Twitter API. Therefore, the collection has been carried out over several months.

We will first describe the collection process of the data from the local elections and then proceed to specify how the data from the parliament was collected. The local elections in Denmark consists of both a municipal and a regional election.

Local election candidates. For the data of the local elections we started with a Twitter list, which is a curated group of Twitter accounts, of political candidates. The list was published on a Twitter account named @KV17DK[3], which in the bio stated, that the account is managed by Troels Runge[4] and Camilla Rozenfeld[5]. People could request for a Twitter profile to be added to the list, by sending a Twitter profile of a possible political candidate to the two administrators. The administrators would then determine which profiles would make it on the list.

The list of Twitter profiles gathered from @KV17DK, was cross-referenced with the names of political candidates on a list published by the news agency *Altinget*. This allowed us to automatically link Twitter profiles to a political candidate and a party. All candidates with names that had duplicates were discarded, as there was no way of distinguishing them. We will for the most part of the thesis work with a subset of local candidates, namely those that are affiliated with one of the 9 parties that are also represented in parliament. This choice was made because local elections have a large amount of small and regional parties, which aren't well represented on Twitter and often have political agendas that are very specific to their municipality.

Politicians of the Danish Parliament. The website *twitterpolitikere.dk*, curates a list of Twitter profiles owned by Danish politicians. The list was mainly comprised of high profile politicians at the time we collected our data[6]. The names of the Twitter profiles we collected from this website, was then cross-referenced with names of parliament members and their corresponding party. The names and political affiliation of parliament members, was gathered from the Danish Parliament's website[7]. As was was done for the local election candidates, parliament members with names that had duplicates in our dataset were discarded.

Manually labelled politicians. In addition to automatically cross-referencing names of political Twitter profiles to lists of candidates from the local election and the parliament, we also manually labelled a subset of unlabelled Twitter profiles to a political party. The selection of profiles was done by sorting the unlabelled Twitter profiles from most to least tweets produced. We are thereby labelling the profiles which had the most tweets first to get as much data as possible. Selectively choosing which politicians to include based on some criteria introduces a sampling bias to the dataset. This concern and others will be addressed in the following paragraph.

[2]Altinget.dk is an independent online newspaper concerning Danish politics.

[3]KV17 is the abbreviation to denote the local election of 2017 and DK is an abbreviation of Denmark.

[4]PhD fellow at the IT University of Copenhagen

[5]Editorial Manager at TVSYD

[6]The list of politicians on the website *twitterpolitikere.dk* was later merged with the aforementioned list from @KV17DK, but this was not an actuality when we collected our data.

[7]www.ft.dk

Sampling bias. The dataset collected for this research is in different ways subject to sampling bias. One of the goals of this study, is to characterize how Danish politicians communicate on social media and what distinguishes members of the political parties. However, the list of politicians that was gathered for this research cannot be said to represent a random sample of politicians. Parties are unequally represented in this dataset, as some parties are better represented on Twitter or simply have more politicians affiliated with their party. Furthermore, some politicians are more active on social media than others, and therefore generate more content for us to analyze. This means that individual politicians also introduce bias into the dataset, as these politicians will have a bigger impact on the derived characteristics of their corresponding party than less active politicians. The decision not to adjust for these biases, is because re-adjusting to the least represented party or the least content generating politician, would significantly diminish the amount of information. This can be observed from the data description in section 2.3. As the methods applied in this research are quantitatively driven, this would reduce the possibility of constructing meaningful analyses and models.

In chapter 3, we also compare the results of our analysis to studies and opinion polls made on the general public and the political characteristics of the political parties in general. This might not be an equal comparison, as the politicians active on Twitter can not be said to be representative of politicians in general from the different parties. All of these aspects have to be taken into consideration when drawing conclusions from the analyses performed in this study.

Other considerations about the dataset. The dataset is partly comprised of high profile politicians, some of which have had the profession for a long time and are a part of the Danish Parliament. Others, are local candidates, some of which might be first time candidates for a political party. We will collectively refer to the set of profiles as *politicians*, *authors* or *actors*. We do not distinguish between the degree to which they are politicians as a full-time profession or are amateurs. This diversity can lead to varying political standpoints and ways of communicating that we believe strengthens the dataset. When the tweet history of a Twitter profile is gathered, we collect all data that is available through the Twitter API. However, as we have no way of distinguishing when a Twitter profile's political career begins or whether it has already ended, it might result in us having tweets from a profile which were created before or after it was part of a political party. As will be demonstrated in some of the following sections, most of the tweets gathered are recently produced. Furthermore, most of the analysis is conducted on a subset of tweets that is user specified as being politically related. The data gathered and used for this research is stored in a database, for a full description of the data storage see appendix A.

2.2 Properties

During the collection of the tweets, we included all *metadata*, but we only use some of it for the analyses performed in this thesis. The attributes that we use in our analyses can be seen in table 2.1, where a short description about the attributes are also provided. For a full reference of the attributes see appendix A.

Attribute	Type	Example	Description
author_id	number	124512412321	Unique identifier for the Twitter accounts
text	text	Husk at stemme	Text written in a tweet
party	text	Venstre	Name of a party
created_at	number	1524732445	Timestamp for when the tweet was created

Table 2.1: Dataset attributes - the attributes that we use in further analyses are included here with their corresponding type, an example and a short description.

2.3 Descriptive statistics

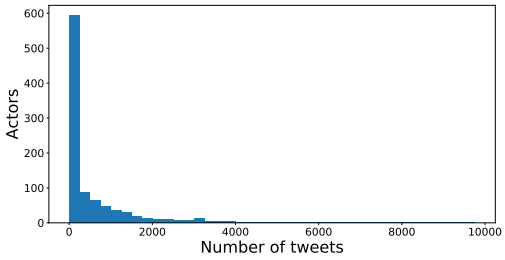
In this section, we report some descriptive statistics of the dataset. The dataset consists of 584, 156 tweets in total, generated by 1569 authors, over a period of approximately 10 years. Even though the total amount of Twitter profiles in the dataset amounts to 1848 politicians, 279 of the profiles haven't written any tweets. Out of the 1569 tweet generating profiles, 951 are in our dataset labelled as being part of one of the parties that are represented in parliament. These profiles account for 479, 899 of the total tweets. It is tweets from this subset of actors that will mainly be used in this thesis. However, tweets from the total set of political actors are used in chapter 4. The following descriptive statistics will be of the 951 parliament party politicians and their corresponding tweets.

Twitter activity across actors. There is a high degree of variation between the amount of tweets generated by the actors in the dataset. The number of tweets generated by an actor ranges from 1 to 9760. The mean tweet count for an actor is 372, which is artificially high, since there are a few outliers that skew the mean. The median tweet count is 106, which shows how the outliers skew the mean. The distribution of tweet counts amongst actors is reported in figure **??**. The distribution is both represented on a regular scale in figure 2.1a and a log-log scale in figure 2.1b. This is due to the long tail of actors that generate more tweets than others.

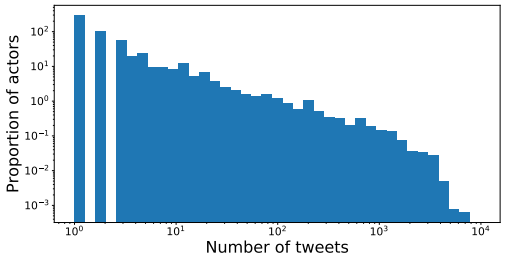
The distribution of tweet counts amongst actors implies that some actors are more represented in the dataset than others, and might have a bigger impact on the results of our analyses than less content generating actors. Adjusting for the disproportion in generated tweets, by reducing the amount of tweets from each actor, to the level of the lowest count or even the median, would greatly reduce the amount of information available. Contrarily, adjusting by removing all actors that have generated few tweets, would lead to the dataset being more dominated by fewer actors, introducing even further bias. Therefore we have chosen to keep the dataset the way it is, and keep this bias in mind when drawing conclusions.

Twitter activity across parties. If the dataset is to be used to define characteristics for each party, then it is important that the parties are represented equally to a degree, such that they are comparable.

The mean count of tweets generated by a party is 53, 322 and the median is 44, 134. Figure 2.2a shows the amount of tweets generated by each of the 9 parties represented in the Danish Parliament. Some of the parties have a larger amount of tweets generated than others, but not to a disproportionate degree, as seen across actors in figure **??**. If we look at the distribution of tweets generated across actors within each party, seen in figure 2.2c,



(a) The distribution of tweets produced by actors shown on a regular scale exhibits characteristics of a long-tailed distribution. The largest number of tweets produced by a single actor is 9760. The mean number of tweets generated by a political actor is 372, but the median number of tweets generated is only 106. The large difference can be attributed to the skewed distribution.



(b) The distribution of tweets produced by actors on a log-log plot with adjusted bin sizes. The proportion of actors producing a certain number of tweets almost show a linear relationship which suggests the distribution is more heavy-tailed than an exponential distribution. The bin values have been adjusted to the increasing bin size brought on by the x-axis' log scale and therefore can contain values that are smaller than 1.

Figure 2.1: Tweet count distribution on both a regular scale and a log-log scale

we can see that some parties are more uniformly represented by their members than others. However, most of the actors across parties lie somewhat within the same range.

Tweet attributes. The structure and general content of the tweets, is an important aspect to consider because this can have implications for some of the analyses performed in later chapters. Some of the methods applied in this thesis, such as Recurrent Neural Networks (RNN), require inputs that are of fixed length and we will therefore cover some general attributes of tweet texts.

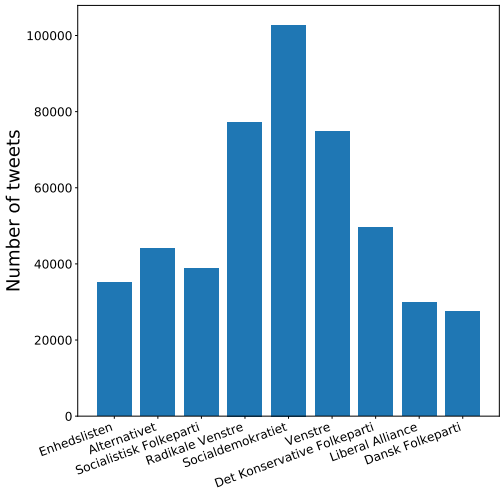
The maximum number of characters in a tweet, was up until the end of 2017 limited to 140, afterwards it was increased to 280. If the character limit is exceeded in a tweet, the tweet text is truncated to the limit.

Diving into the statistics of a tweet, the mean and median number of characters is 110 and 128 respectively. The proportional distribution of how many characters there are in a tweet can be seen in figure 2.3a. The figure shows that most of the tweets lie on the old limit of 140 characters. Few tweets in the dataset are beyond the old 140-character limit and the consequences of this is therefore neglected in our analyses.

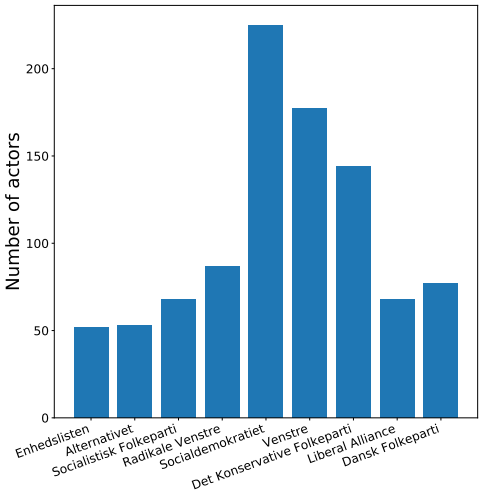
As a natural consequence of the character limit, the number of words in a tweet is also to some extent limited. Figure 2.2b, shows the proportional distribution of the number of words in the tweet texts. This is an appreciative quality of the tweets, because it allows us to transform the tweet texts into uniformly lengthened sequences, that we with some certainty, can assume won't remove any excess words.

2.4 Preprocessing

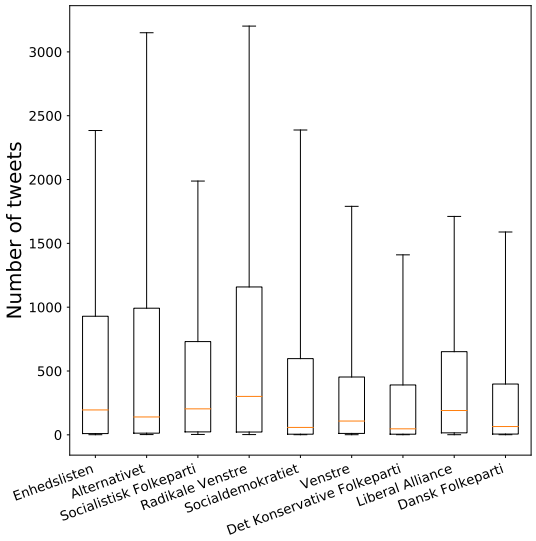
Since a tweet constitutes of written text, a large number of unique tokens are used. This is especially true for text gathered from social media, which in addition to traditional writ-



(a) Histogram over number of tweets produced by actors, but aggregated on the level of the actors party affiliation. This shows how many tweets there are in the dataset from the different parties.

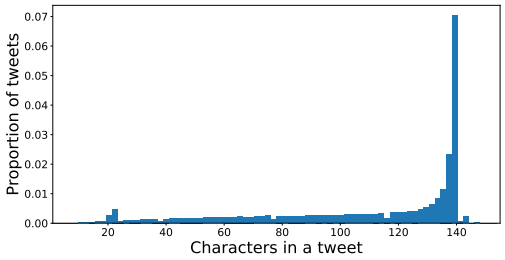


(b) Histogram over the number of members each party has in the dataset. Interestingly, *Radikale Vesntre* is around half the size of *Venstre*, but as seen on figure 2.2a, *Radikale Venstre* has produced more tweets.

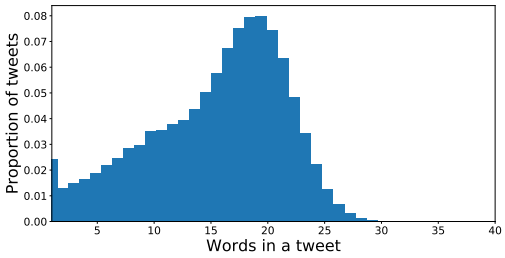


(c) Box and whisker plot showing how much actors from different political parties tweet. We see the median is relatively high for *Radikale Venstre* and the quantiles lie far apart. The extremes are the 5th and 95th percentile, outliers are not shown.

Figure 2.2: Tweet length distribution on word and character level. Actors are mapped to a party. We only include parties that are also represented in the Danish Parliament. The distribution of tweets is not uniform and certain parties are better represented than others



(a) Histogram of tweets with a certain number of characters in it. The maximum number of characters has been 140 up until the end of 2017 where it was increased. This explains the large peak around 140 on the x-axis and why there are few tweets with more than 140 characters. Notice the peak around 20 characters which corresponds to the length of a URL shortened automatically by Twitter.



(b) Histogram of tweets with a certain number of words in it. The character limit in a tweet results in a short right-tail in the probability distribution. There is a peak at one-word tweets, which might be attributed to the same tweets as the peak seen in 2.3a, at around 20 characters.

Figure 2.3: Tweet length distribution on word and character level.

ten language, also includes various symbols, emojis[8] and website links. Preprocessing this content allows for a more aggregated and manageable dataset. We will throughout the thesis adhere to the following preprocessing steps of the tweet texts, for the various analyses conducted.

First we separate the content of a tweet's text into 4 different categories, *user-mentions*, *hashtags*, *URLs* and the remaining content in the tweet, which we will refer to as *plain text*[9]. This is done partly because these elements need to be processed in differently and because these elements represent different sources of information, with different qualities. Therefore, it is reasonable to analyze these categories separately.

A note on networks. The content of our dataset is produced by a network of actors. Social networks have been shown to exhibit special dynamics, which are outside the scope of this thesis. Boutet et al. [7] show that these dynamics can be good predictors of political leaning. Specifically, hashtags, user-mentions, retweets and URLs carry a lot of the political signal there is in a tweet. Since our thesis for the most part, deal with linguistic differences and natural language processing, we exclude the hashtags, user-mentions and URLs from most analyses. The overall focus on using plain text for political classification, stems from the generalizability of the concepts to other domains and the interpretability of the results. However, we will also evaluate the potential signal gain from using these features in some analyses.

URLs. URLs are in tweets shortened and replaced by a Twitter link-shortener. The original URLs are however available in the Twitter object, along with information about the replacement link and its location in the text. This information is used to find the

[8]emojis are ideograms and smileys used in text

[9]By *plain text* we do not mean it in the literal computer science sense. We chose the name plain text as it relates to how we think about the structure of the tweet.

original URLs and separate the replacement URLs from the other categories. URLs comes in many forms, as a link for a webpage by default needs to be unique. This results in a large vocabulary with very few occurrences. Therefore, the URLs are shortened to their domain name, such that trailing path, specifying a webpage on a website, is removed.

Hashtags. The hashtags in a tweet text are also available in the Twitter object, along with their positions in the text. This information is used to extract and separate them from the remaining content. In some chapters of this thesis, we use a topic specific subset of tweets. This topic specific subset of tweets, is inferred via which hashtags are included in a tweet, as hashtags are used to subscribe a tweet to a certain topic.

User-mentions. Information about user-mentions, are also available in the Twitter object and are collected and separated from the general content, in the same way as the other elements.

From plain text to tokens. All plain text is lower-cased and stripped of all characters that are not numerical or alphabetical. Removing symbols and emojis from the tweet text also removes information that could be valuable, however, excluding them decreases the set of words in the resulting vocabulary. If a tweet has been truncated due to the character limit being exceeded, the last word in the tweet is removed, because it most likely will be an unintentionally incomplete word. The remaining plain text is then split on white-space into tokens.

Filters. We have throughout the development constructed various filters, to control which tweets should be included for the various analyses. We will cover some of the more important ones that are used in the analyses.

Topics - as mentioned, we use hashtags to extract tweets that are related to certain topics.

Timeframes - the date of which a tweet is created, is used to extract time specific tweets.

Language - non Danish tweets are removed because it is difficult to compare results across languages.

Retweets - in Twitter it is possible to re-post other users posts. The retweets are removed when training and testing classification algorithms, because they represent duplicates in the dataset.

To determine whether a tweet is written in Danish, we count the amount of plain text tokens that can be found in the Danish dictionary[10]. If the proportion of Danish tokens in a tweet is below 0.5, we discard the tweet. This is done because we in some sections of the thesis, try to distinguish which words are specific for the different political parties. Some parties are more inclined to write tweets in other languages than Danish, and it is difficult to compare linguistic differences across languages.

Counting all the unique tokens in the set of plain text elements and sorting by frequency, yield a distribution that can be seen in figure 2.4. The figure is shown on a log-log scale. From the distribution we can infer that the structure follows a generic distribution of word usage [12].

[10]The dictionary used is provided by the website *da.speling.org*

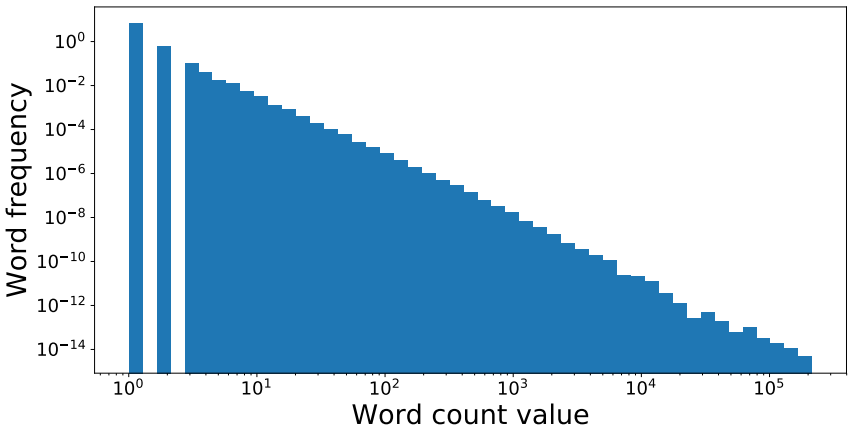


Figure 2.4: The histogram of word usage ranked from the most frequently used word to the least used, shown on a log-log scale plot. This shows a generic long-tailed distribution of word usage that is empirically found in many text corpora. That it doesn't show any special characteristics is good since it shows the text data is not special in this regard.

Temporal distribution of tweets. The amount of generated tweets in the dataset varies over time. In recent years, there has been a growth in the amount of tweets generated by the Twitter profiles in the dataset. In figure 2.5, we can see the accumulated tweet count for each party aggregated by month. The counts have been stacked onto two different wings, one which represents the governing and government supporting parties, another which represents the government opposing parties. The governing parties and supporting parties are marked by nuances of blue and have and have an accumulated count going downwards. The government opposing parties are marked by nuances of red and have an accumulated count going upwards. The figure shows that peaks in the number of tweets generated, correlates with certain events, e.g. there is an increase in tweets generated around the 18th of June 2015, which was the day of the last government election.

As mentioned previously, hashtags can be used to infer the topics of a tweet, but they are not static properties, hashtags can be trending and extinguish over time. We will in chapter 3, use a subset of tweets that are user specified as being related to politics, to characterize linguistic differences of political parties. These tweets are extracted, by filtering away all tweets that do not contain the hashtags, *#dkpol*[11], *#kompoldk*[12], *#dkpolitik*[13], *#kv17*[14] and *#fv15*[15]. These tweets amount to $97,676$, out of the $479,899$ produced by politicians that are member of a parliament party. Because political focus shifts over time, it is important that this subset of tweets is not restricted to a specific time period. Figure 2.6, shows the number of tweets that are generated per party on a monthly basis, which includes the aforementioned hashtags. We can see that the number of topic specific tweets generated, follow the same patterns as the overall amount of tweets generated. They are represented in most time periods except for the early years of Twitter. Therefore we conclude that they

[11] The *#dkpol* hashtag is an abbreviation for *Dansk politik* (Danish politics).
[12] *#kompoldk* is related to issues that concerns municipal politics.
[13] *#dkpolitik* is a hashtag related to Danish politics.
[14] #kv17 is related to the 2017 local elections.
[15] #fv15 is related to the 2015 government election.

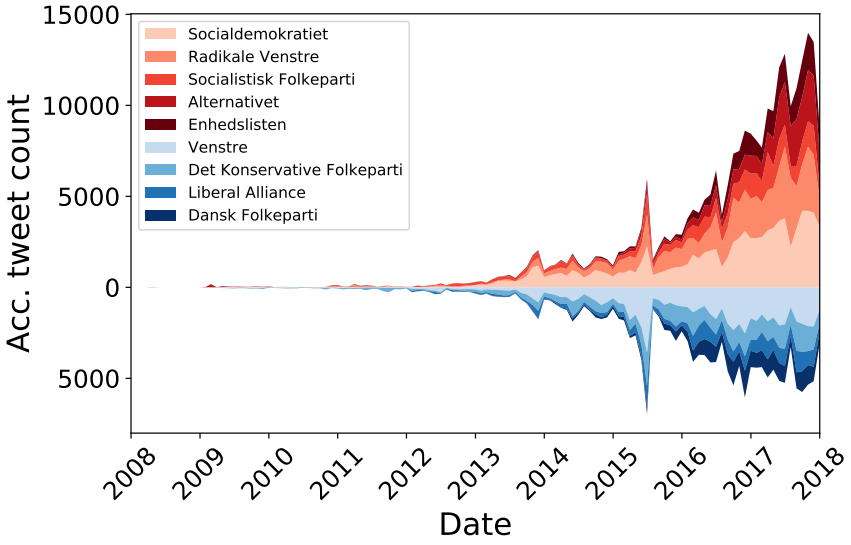


Figure 2.5: The y-axis displays the amount of tweets generated by a party, the counts have been stacked on each side of the political wing to show the accumulated tweets generated per wing. The x-axis displays the dates of the tweet counts which where aggregated on a monthly basis.

are a fair representation of political tweets across periods after 2012.

2.5 Brief Summary

We have now presented the Twitter dataset that we have collected for this project. The dataset consists of the tweet history of politicians in the Danish Parliament and political candidates for the 2017 local elections. We collected the dataset using the Twitter API. We automatically labelled most of the politicians' Twitter profiles by cross-referencing the authors of the tweets, with lists of political candidates. Furthermore, we manually labelled 100 politicians. There are possibilities of biases in the dataset that must be considered and we have described these biases and what they might mean for future analyses. We have also presented the qualities of the dataset and shown the properties that are interesting.

A brief summary about what is included in the dataset is listed here:

Source Collected from Twitter for this masters thesis.

Collection Period Until May 25, 2018.

Actors 1569 politicians and political candidates that have tweeted at least once, 951 of those are affiliated to a party represented in parliament.

Documents 584,156 tweets in total, 479,899 of those are affiliated to a party represented in parliament and contains a subset of 97,676 tweets that are extracted as being related to politics.

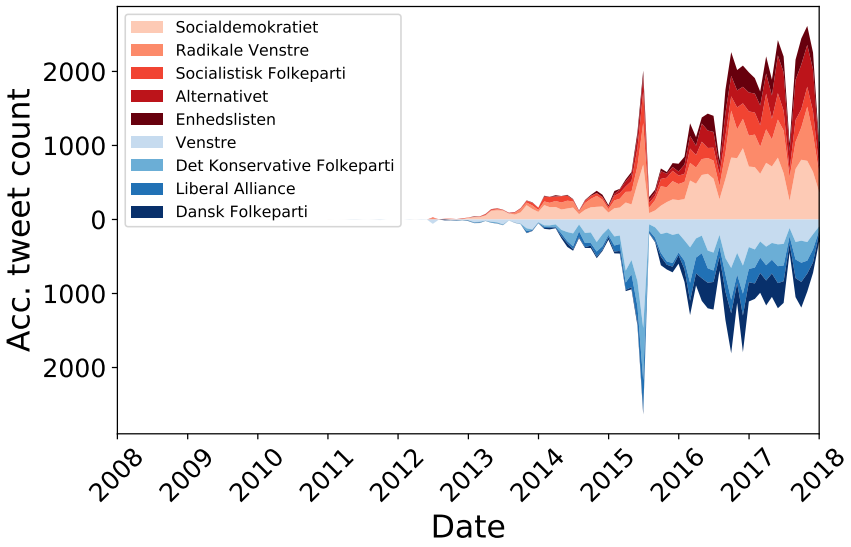


Figure 2.6: The y-axis displays the amount of tweets generated by a party, the counts have been stacked on each side of the political wing to show the accumulated tweets generated per wing. The x-axis displays the dates of the tweet counts which where aggregated on a monthly basis. The number of tweets have been reduced to tweets that contains the hashtags #dkpol, #kompoldk, #dkpolitik, #kv17, #fv15

We also introduced some descriptive statistics about the dataset that shows how the tweet distribution is across the political actors and parties. We can from those statistics derive that we have enough data from each party to do data-driven analyses of language.

Our preprocessing pipeline filters all hashtags, mentions and links away from the plain text. We also remove punctuation, emojis and other special characters. We split on white space and tokenize the text into lists with each entry representing a word. We do not remove words that occur frequently or only rarely.

Chapter 3

Linguistic Differences of Political Parties

We will in this chapter explore the possibility of creating a political weights for words, based on discourse of politicians on Twitter. We perform a step-wise analysis, from simple non-model based approaches, to more advanced model based approaches, using the dataset presented in chapter 2. The goal is to specify a mathematical model such that we quantitatively can attribute a partisan weight to words, based on the word usage of politicians on Twitter. These word weights will be used used in subsequent chapters, as features for machine learning algorithms. Some language modelling techniques, strive to describe the syntactic and semantic value of words. We will in this chapter, explore the possibility of capturing a words political value, given the current political context. One fundamental assumption to this idea, is that words can be attributed a political value in a meaningful way. We will also address this issue in this chapter, by relating to concepts and analyses of political discourse in other fields of science, such as sociology and politology, to argue why this approach is meaningful.

The analyses conducted in this chapter, is inspired by the work done in [11]. There are significant differences between the dataset used in [11] and our own, which we will also address and take into consideration.

3.1 Introduction

Attributing partisanship to words may seem meaningless, as the political affiliation of a word depends on the context in which it is presented. Possibly any word can emphasize different political opinions, depending on how it is used and the circumstances it is presented in. However, different parties have an agenda towards highlighting specific topics in the political debate. This concept is known as *issue-competition* and is due to the belief that voters place their votes, based on their standing on different issues, known as *issue-voting* [13]. Parties are trusted by the public as being good at handling certain issues, which is referred to as *issue-ownership*. The ownership of a certain issue can pertain to a party's history, such as the traditional socio-economic background of their voters [14]. However, issue-ownership is not a static property and can shift over time, either due to political events or due to efforts from a party to appropriate themselves the ownership of an issue, also called

issue-trespassing [15]. It is our hypothesis that issue-competition can be reflected through word usage, as political topics are often associated with a set of words. Political statements about immigration for example, often contain words that are associated to the issue, such as *refugees*, *EU*, or *border control*[1]. Using words to frame the issue could also be indicative of a political standpoint, for example using the term *convenience refugees* instead of *refugees* when debating immigration.

We will in the following sections, cover various methods to evaluate the partisanship of words. The basis for the analysis performed in this chapter, are bags-of-words (BoW) representations of the data mentioned in chapter 2. Many of the methods applied in this section originate from research done by Monroe et al. in [11]. In their work they try to identify words that distinguish Republicans and Democrats in speeches given on different subjects in Congress. These words are by Monroe et al. refered to as *fighting words*. There are some differences in the datasets presented by Monroe et al. and the dataset utilized in this research. The word counts presented in this thesis stem from aggregated tweets, whereas, the data gathered by Monroe et al. are speeches given on specific subjects by the two different parties. These tweets can however be limited to a subset of tweets, by selecting tweets that apply a certain hashtag. Thereby resembling a subject specific piece of text, but there are of course differences. More importantly, the Monroe et al. research revolves around the differences in word usage in a two party system, Republicans and Democrats. In order to apply the same methods in the context of Danish politics, which is a multi-party system, the methods must be extended. However, first we will present the results of applying the different methods on our dataset, transformed to a two party problem. This is done by aggregating all parties that are supporting the governing parties, and those standing in opposition, into two separate wings as seen in table 3.1.

Gov. parties	**Abbr.**	**Opp. parties**	**Abbr.**
Venstre	V	Socialdemokratiet	A
Konservative	C	Socialistisk Folkeparti	F
Liberal Alliance	I	Radikale Venstre	B
Dansk Folkeparti	O	Alternativet	Å
		Enhedslisten	Ø

Table 3.1: Grouping of parties into political wings, representing government supporting parties and government opposing parties. The parties included are all represented in the Danish parliament.

The governing parties Venstre, Konservative as well as the government supporting party Dansk Folkeparti, traditionally represent the right-wing in Danish politics [16]. The parties Socialdemokratiet, Socialistisk Folkeparti and Enhedslisten, traditionally represent the left-wing in Danish politics [17]. In addition to supporting this categorization, a more recent analysis conducted in [18], shows that the parties Radikale Venstre and Alternativet, represent voters that consider themselves a part of the political left-wing. Whereas voters consider the party Liberal Alliance, to be a right-wing party. Because of this, we will in this study refer to the grouping of parties presented in table 3.1, as the left-wing and right-wing.

To limit the set of tweets to those that contain political content, only tweets that include

[1]In danish the words (border control) and (convenience refugees) are 'grænsekontrol' and 'bekvemmelighedsflygtninge' and are therefore single words in our data set

the hashtags #dkpol, #kompoldk, #dkpolitik, #kv17 and #fv15 are included. There are many trending hashtags that are political, but these hashtags were chosen because we consider them to be non-partisan and encompasses all sorts of political issues. Some political hashtags are related to specific issues, such as: education, immigration, or the EU. Using these hashtags could be problematic, as hashtags can also be reflective of the aforementioned *issue-ownership* and has in studies been used to label stance on different subjects [9]. Because the goal of this analysis is to compare and distinguish word usage between the different parties, it is therefore important that the subset of tweets used, is not in itself politically biased. The subset of political tweets, can in the following algorithms be interpreted as the *topic*. The tweet texts have been separated into user-mentions, hashtags, URLs and the remaining content in the tweet, which we will refer to as *words*. The reason behind this distinction is that user-mentions and hashtags are restricted to the Twitter domain. URLs are also only present in a subset of textual contexts and are not as widely generalizible as words. Words are also more interpretable in relation to the previously covered concepts of issue-ownership and issue-competition. Therefore most of the analyses will be based on the words used in the tweets. However, the user-mentions, hashtags and URLs do also in themselves contain interesting information, which will also be presented in this chapter and will be evaluated as features for machine learning algorithms.

The chapter starts with a step-wise review of the different methods that we have applied on the dataset, as well as evaluations on the individual results. By beginning with the most naive approaches, we will highlight their shortcomings, to explain the gradual increase in complexity in the solutions. Lastly, the results of applying the most promising methods to a multi-party scenario, are presented.

General notation We will start by introducing some general notation, which we will adhere to for the remaining part of this chapter. General notation for the analyses in this chapter can be seen in the following list:

- Let $\mathbf{W}$ be the set of words in the corpus
- Let $w = 1, 2, ..., W$ be the index of words in the set of words $\mathbf{W}$.
- Let y denote the word frequencies in the set $\mathbf{W}$.
- Let $k \in K$ denote a topic in the corpus.
- Let $I = \{L, R\}$ where L denotes the left-wing parties and R denotes the right-wing parties.
- Let $i \in I$ be a partition of the documents in the corpus.

3.2 Non-Model Based Approaches

3.2.1 Word Frequencies

One simple way of measuring the importance of words in different text corpora, is to count the occurrences of words. How often a word is used by one party, or in our case a group of parties, is a naive way of interpreting which words that distinguish the two political wings.

In figure 3.1, we display a plot where the y-axis is the difference in word counts by each wing calculated as shown in equation (3.1).

$$y_{kw}^{(L-R)} = y_{kw}^{(L)} - y_{kw}^{(R)} \tag{3.1}$$

Where $y^{(L)}$ and $y^{(R)}$ are vectors of word frequencies of the left-wing and the right-wing parties. k is a given topic and w is a specific word. The x-axis of figure 3.1 represents the total count of a word's frequency, of both wings in a log scale.

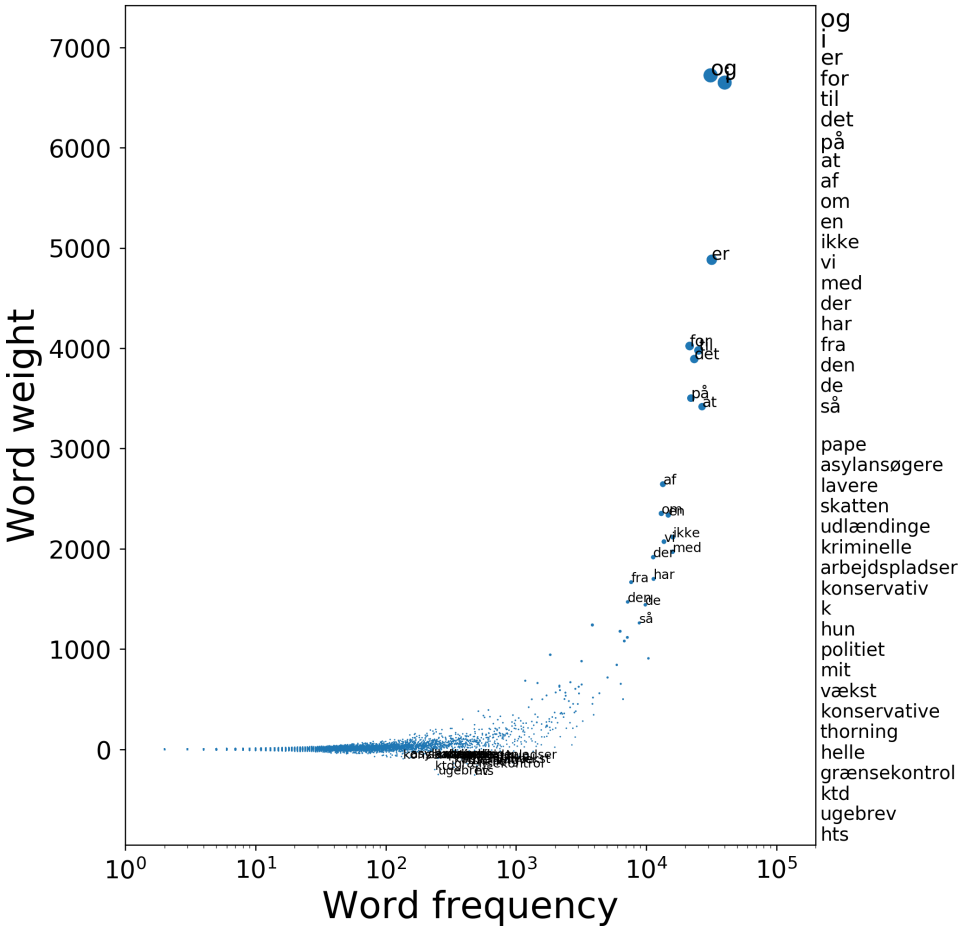


Figure 3.1: Word frequencies between left-wing and right-wing parties. The y-axis indicates partisan weight of a value where values $y > 0$ indicates left-wing partisanship and values $y < 0$ indicates right-wing partisanship. In this case the partisan weight is simply the frequency of the word left-wing parties subtracted the frequency of right-wing parties. The x-axis shows the total frequency of the word in both wings.

From figure 3.1, it can be seen that words that occur often in the total word count, also tend to score higher y-values. This is due to the disproportionate amount of words generated by the parties of the left-wing, compared to the right-wing parties. Therefore, words that have high occurrence score higher y-values. The difference is larger for words

that occur often, even though the words might be in equal proportion, compared to the total word count of each wing. This problem leads to the solution in the following section.

3.2.2 Word Proportions

Normalizing word frequency into word proportions, is a common way of making two BoWs comparable. In figure 3.2, the y-axis represents the difference in word proportions between left and right-wing parties. The values are calculated as shown in (3.2).

$$y_{kw}^{(L-R)} = f_{kw}^{(L)} - f_{kw}^{(R)} \tag{3.2}$$

Where the proportions are defined as, $f_{kw}^{(i)} = y_{kw}^{(i)}/n_k^i$ and n_k^i is the total word count of a wing. The x-axis of figure 3.2 represents the total count of a word's frequency, of both wings in a log scale.

Utilizing proportions removes the asymmetry brought on by the disproportionate amount of words generated by each wing. However, variations in high frequency words is larger between wings, than variations in low frequency words. Therefore, high frequency words dominates the extremes, even though they would not be considered particularly partisan.

3.2.3 Word Log Odds Ratio

Using log odds ratios instead of differences in proportions, has the advantage of being unbounded, making this measure more comparable between results [19].

Odds are calculated, as the amount of favourable outcomes to unfavourable outcomes ratio. In this context we calculate the odds of words being used by a wing as shown in equation (3.3).

$$O_{kw}^{(i)} = \frac{f_{kw}^{(i)}}{1 - f_{kw}^{(i)}} \tag{3.3}$$

The odds ratio between wings is calculated as shown in equation (3.4).

$$O_{kw}^{(L-R)} = \frac{O_{kw}^{(L)}}{O_{kw}^{(R)}} \tag{3.4}$$

This however leads to a division by zero, when a word is present in one corpus but not in the other. The simple solution to this is to add a small amount to all frequencies, such that $y'^{(i)}_{kw} = y_{kw}^{(i)} + \epsilon$. For this example we have chosen to add 0.5 to all word counts. The resulting values are bound to be in the range of $O_{kw}^{(L-R)} \in [0, \infty]$, calculating the logarithm of the odds, results in values that are unbounded in the range of $\log(O_{kw}^{(L-R)}) \in [-\infty, \infty]$ and symmetrical around zero. The resulting plot can be seen in figure 3.3, where the y-axis is calculated as shown in equation (3.5).

$$y_{kw}^{(L-R)} = log(\frac{O_{kw}^{(L)}}{O_{kw}^{(R)}}) \tag{3.5}$$

The x-axis represents total word occurrence in a log scale. Figure 3.3 shows that the problem with this approach, also relates to the difference in variability between high and low frequency words. However, in this case it is the low frequency words are over represented

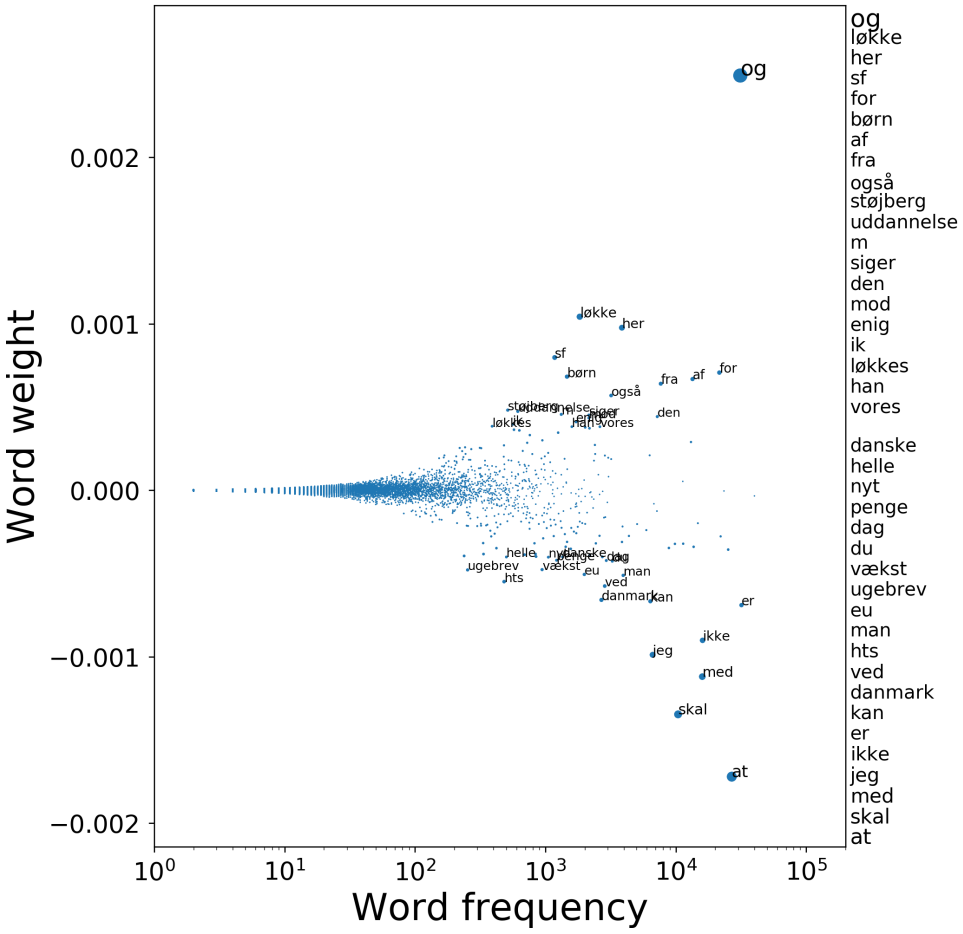


Figure 3.2: Word proportions between left-wing and right-wing parties. The y-axis indicates partisan weight of a value where values $y > 0$ indicates left-wing partisanship and values $y < 0$ indicates right-wing partisanship. As opposed to calculating the difference in frequencies, the difference in proportions is symmetrical, thereby adjusting for the disproportionate amount of tweets between wings. The x-axis shows the total frequency of the word in both wings.

in the extremes of the y-axis. This is due to the problem stated previously, words that occur in one group of parties but not in the other, result in high values even though they occur rarely and are not representative of partisanship.

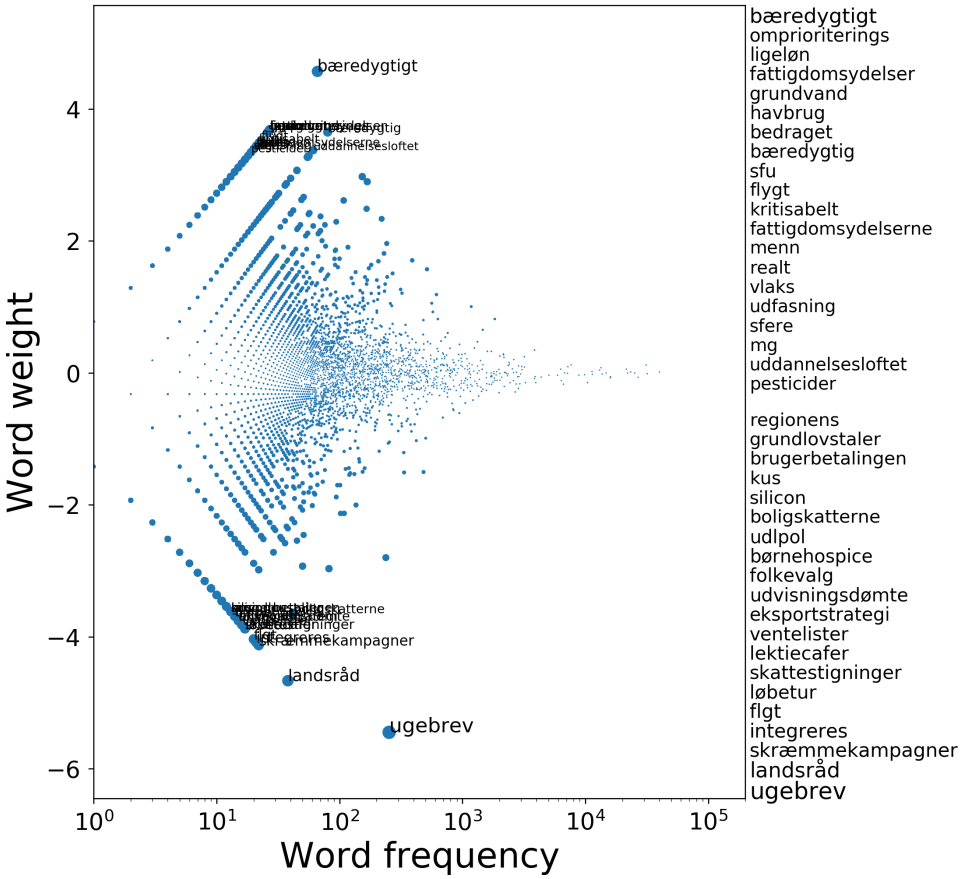


Figure 3.3: Word log odds ratio between the left and right-wing parties. High positive values on the y-axis indicate that a word is weighed as being left-wing and low values on the y-axis means a word is weighed as being right-wing. The log odds ratio has the property of being unbounded as opposed to difference in proportions.

3.3 Other Applied Methods

We will in this section, cover the results of other methods that have been applied to this dataset, but do not relate to the final solution.

Tf-Idf. Tf-idf is a commonly used algorithm in computer science and is intended to highlight words that are specific for a corpus in a collection of corpora [20]. The strengths of this algorithm lies in its simplicity and has proven to be applicable in a wide range of cases. In regards to machine learning algorithms, tf-idf can be used for feature extraction because

it emphasizes words that are salient for different corpora. However, due to its inability to account for the relative word frequency between corpora, it has not shown to be useful on this dataset. A more thorough explanation of the tf-idf algorithm used on the dataset as well as the results can be found in appendix B.

Wordscores. Wordscores is a method that was developed specifically to be used on political [21]. Wordscores is a way of extracting policy positions from political texts and is also used to classify new documents. The method is a two step process, where new documents are assigned scores based on word counts from previously known political texts, by utilizing calculated wordscores. We have in our analysis, only evaluated the potential of using the calculated wordscores to distinguish words that are partisan. We have not applied the second step, which is to score newly introduced documents. Even though wordscores do seem to correctly emhpasize words that are partisan, it tends to favour low frequency words and especially words that only occur in one corpus. This bias is problematic, as low frequency words that only occur in one corpus might not be particularly partisan, but simply be rarely occurring words or misspellings. This is especially true when working with data from a source like Twitter, which endorses quickly written, short messages and where the character limit encourages the users to be creative about their word abbreviations. Furthermore, by neglecting high frequency words, a lot of useful information is lost. A more thorough explanation of the wordscores algorithm used on the dataset as well as the results can be found in appendix C.

3.4 Model Based Approaches

The basis for the models presented in the following sections, is a calculation of the log odds ratios in a probabilistic manner, using equation (3.6).

$$\delta_{kw}^{(L-R)} = log(\frac{y_{kw}^{L} + \alpha_{kw}^{L}}{n_k^{L} + \alpha_{k0}^{L} - y_{kw}^{L} - \alpha_{kw}^{L}}) - log(\frac{y_{kw}^{R} + \alpha_{kw}^{R}}{n_k^{R} + \alpha_{k0}^{R} - y_{kw}^{R} - \alpha_{kw}^{R}}) \tag{3.6}$$

This equation bears resemblance to the log odds ratio presented in section 3.2.3, except for the addition of a prior α and the use of word frequencies instead of proportions. The prior α is a word vector of size W where $\alpha_{kw} > 0$ and $\alpha_{k0} = \sum_{w=1}^{W} \alpha_{kw}$. To account for low and high frequency words dominating the extremes, certainty is applied to the model by calculating an approximation of the estimate's variance.

Note *The following paragraph concerning variance, is our own interpretation of how variance is estimated in the model. This is because, there is no explicit explanation as to how the formula for variance estimation is derived in reference [11].*

Variance. It is known that the variance can be estimated from a normal distribution by maximising the likelihood. Recognise that the proportion $f_{kw}^{(i)} = y_{kw}^{(i)}/n_k$ is also a multinomial distribution of probabilities, which we will denote as π. The log odds can then be written as a function of the probabilities as $\Omega_{kw}^{(i)} = \frac{\pi_{kw}^{(i)}}{1-\pi_{kw}^{(i)}}$ also known as the logit function [22]. We can inversely transform the log odds back into a probability as, $\pi_{kw}^{(i)} = \frac{\exp \Omega_{kw}^{(i)}}{1+\exp \Omega_{kw}^{(i)}}$.

By using the logit transformation, we can write the log likelihood function of the odds as,

$$\sum_{w=1}^{W} y_{kw}^{(i)} log(\frac{\exp \Omega_{kw}^{(i)}}{1+\exp \Omega_{kw}^{(i)}})$$

The benefit of calculating the log likelihood instead of the likelihood, is that the joint distribution can be summed up instead of being multiplied. Especially in the case of working with natural languages, since the probabilities of a specific word being used, is often relatively low. The variance estimates can then be analytically derived from the log likelihood function. The approximation of the variance is shown in equation (3.7). The resulting model is the z-scores of the log odds ratios, as shown in equation (3.8).

$$\sigma^2 = \frac{1}{y_{kw}^L + \alpha_{kw}^L} + \frac{1}{y_{kw}^R + \alpha_{kw}^R} \tag{3.7}$$

$$\zeta_{kw}^{(L-R)} = \frac{\delta_{kw}^{(L-R)}}{\sqrt{\sigma^2(\delta_{kw}^{(L-R)})}} \tag{3.8}$$

This approximation of the variance is based on two assumptions, the first is that $y_{kw}^i << n_k^i$. The other is that $\alpha_{kw}^i << y_{kw}^i$. By applying certainty to the log odds ratios, the resulting extremes are no longer dominated by low frequency words as the example in figure 3.3. In figure 3.4, the prior is a symmetric Dirichlet distribution, also called an uninformative prior, since the prior is not based on any previous information.

3.4.1 Multi-Party Extension

We explore two different ways of extending the model based approach to a multi-party model. One approach is to calculate the word weights between each party, to do this the equations (3.6), (3.7) and (3.8) could be extended with arbitrary party variables *i,j*. This is done in equations (3.9), (3.10) and (3.11).

$$\delta_{kw}^{(i-j)} = log(\frac{y_{kw}^i + \alpha_{kw}^i}{n_k^i + \alpha_{k0}^i - y_{kw}^i - \alpha_{kw}^i}) - log(\frac{y_{kw}^j + \alpha_{kw}^j}{n_k^j + \alpha_{k0}^j - y_{kw}^j - \alpha_{kw}^j}) \tag{3.9}$$

$$\sigma^2 = \frac{1}{y_{kw}^i + \alpha_{kw}^i} + \frac{1}{y_{kw}^j + \alpha_{kw}^j} \tag{3.10}$$

$$\zeta_{kw}^{(i-j)} = \frac{\delta_{kw}^{(i-j)}}{\sqrt{\sigma^2(\delta_{kw}^{(i-j)})}} \tag{3.11}$$

Utilizing the method with n parties would result in a $n-1$ simplex for each word, where each edge would represent a word's weight between two parties. The number of weights for each word, would then be the number of edges in the simplex, $\frac{n\cdot(n-1)}{2}$. In our case where there are 9 parties the result would be 36 weights for each word.

Another approach is to calculate the word usage between a party and the accumulated word counts of all the other parties. The benefit of this approach is that the dimensionality of the resulting weights is equal to the number of parties i.e. 9 dimensions. The reduced dimensionality from the second approach, improves the interpretability of the results. Therefore we will only display the results of using the second approach. However, both methods will be used in subsequent chapters. In table 3.2, we show the 6 most representative words and in table 3.3 we show the 6 least representative words for each of the parties.

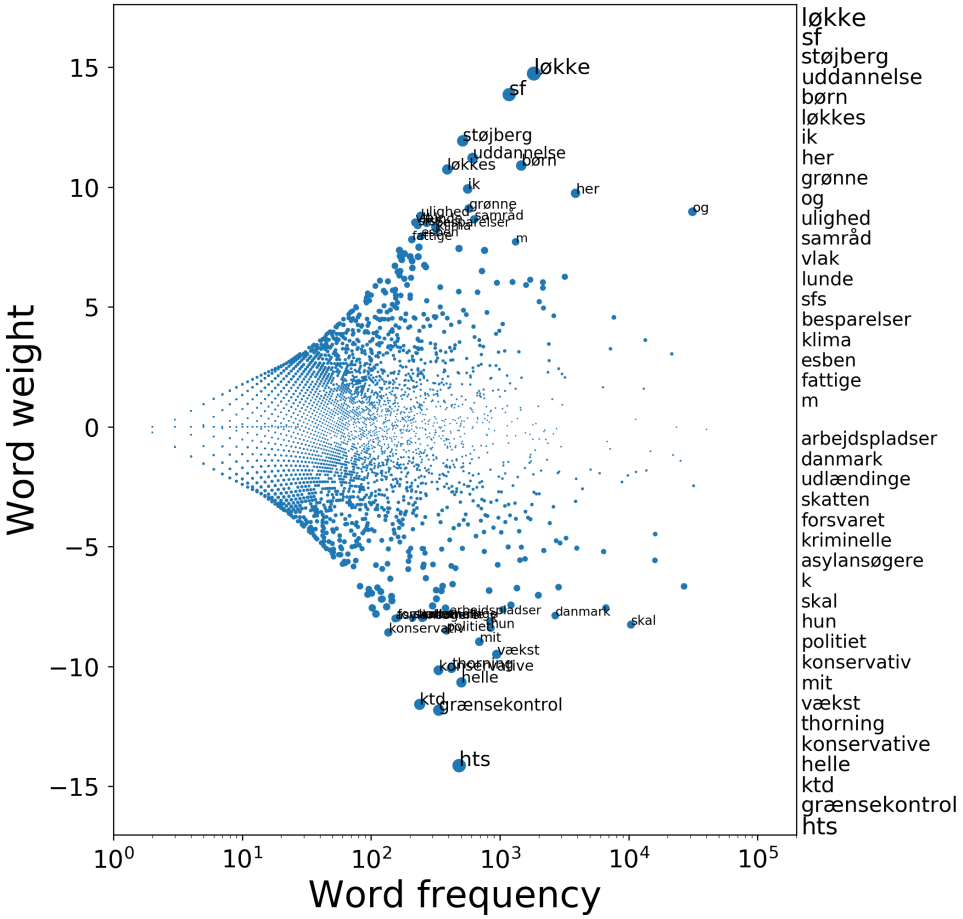


Figure 3.4: Resulting word weights of using a model based approach with a prior value of 1.0e−15. High positive values on the y-axis indicate that the word is representative of the left-wing. High negative values on the y-axis indicate that the word is representative of the right-wing. The x-axis shows the words total frequency in both corpora. By applying certainty to the log odds ratios the model does not over emphasize low frequency words.

	1	2	3	4	5	6
A	løkke	mette	løkkes	vlak	s	eb
B	radikale	uddannelse	ikk	børn	radikal	støjberg
C	k	konservative	pape	konservativ	boligejerne	grundskylden
F	sf	sfs	gratis	m	arbejdstid	psykologhjælp
I	la	liberal	frihed	alliance	regist. safgift.	assens
O	df	grænsekontrol	ktd	dfs	mit	permanent
V	hts	helle	venstres	thorning	venstre	hun
Å	her	alternativet	vi	grønt	københavn	tak
Ø	enhedslisten	el	mærsk	handicap	skoledage	samråd

Table 3.2: The 6 most partisan words for each party, when scored between each party word count vs the aggregated word counts of all other parties. Party abbreviations have been used to minimize table space.

	6	5	4	3	2	1
A	skal	flygtninge	eu	at	sf	vi
B	skattelettelser	hts	s	løkke	sf	mette
C	løkke	venstre	mette	sf	df	s
F	ik	s	her	dansk	mette	vækst
I	her	venstre	løkke	v	regeringen	df
O	skat	du	vækst	venstre	løkke	og
V	børn	støjberg	la	her	vores	df
Å	m	venstre	sf	v	løkke	df
Ø	er	man	det	så	tak	vi

Table 3.3: The 6 least partisan words for each party, when scored between each party word count vs the aggregated word counts of all other parties. Party abbreviations have been used to minimize table space.

3.5 Evaluation

We evaluate our results qualitatively, by comparing the most representative words for each wing, according to the values presented in figure 3.4, to issue-ownerships of parties according to opinion polls conducted during the 2015 election [23]. The results of the opinion polls, show that some parties are considered by voters to be better at handling certain issues than others. Even though there is some variation between left and right-wing voters, there is a general agreement of issue-ownership across voters. The right-wing parties are considered most competent at solving issues that involve *securing law and order*, *ensuring low taxes* and *competitiveness*, whereas the left-wing parties are considered adept at solving issues, such as *the environment*, *unemployment* and *wellfare*. Furthermore, the study also showed that even though the right-wing weren't considered by voters to be better at ensuring a reasonable immigration's policy, the large amount of media coverage that this issue received may still have worked in favour of the right-wing parties. This could be due to many voters feeling that the right-wing stance on immigration were closer to their own [23]. In the results from the model seen in figure 3.4, we can see that some of the high scoring words are consistent

with the results in reference [23]. Words like *grønt, grønne* (green), *klima* (climate), *ulighed* (inequality), *fattige* (poor), *uddannelse* (education) and *besparelser* (spending cuts) have high positive values, therefore being more representative of the left-wing in the model. Whereas words like *udlændinge* (foreigners), *asylansøgere* (asylum seekers), *grænsekontrol* (border control), *skatten* (the taxes), *kriminelle* (criminals), *politiet* (the police), *forsvaret* (the defence/military), *vækst* (growth) have high negative values, therefore being more representative of the right-wing in the model. However, many of the words located in the extremes of figure 3.4, aren't associated to any particular issue. Instead, many of the words are names of politicians or political parties. These results are important to consider when trying to generalize these results to other contexts, as the data that this analysis is based on is biased towards the political agenda of the parties. This is in essence also what the model is intended to capture through the concept of issue-ownership. However, the question remains, whether these word weights can be generalized to other contexts, such as media articles, blogs or other digital text content.

In addition to applying the model on the words used in the tweets, we have also explored possibility of applying the model on the hashtags, user-mentions and URLs. The results and analyses of this exercise can be seen in appendix D and offer interesting insights. However, they do not confer any further understanding the topics covered in this chapter and has therefore been left out of the main part of the thesis.

The model that was presented in this section and the calculation of an appropriate prior, are the fulcrum of the analysis in the following section. As well as a cornerstone in the training of machine learning algorithms, in chapter 5.

3.6 Ephemeral Aspects Of Political Discourse

In this section, we explore the temporal aspect of attributing political weights to words. The section consists of a step-wise introduction to different solutions and a review of their results. The goal of the following examinations, is to construct a solution that is adaptive to changes in the political domain. However, as the intention is to use these word weights as features in machine learning algorithms, consistency is also important. If the weights fluctuate too rapidly due to noise or the influence of other variables. The machine learning algorithms might have difficulties extracting patterns from the resulting features.

Dynamicity of word partisanship. The partisanship of words is a dynamic property as it is a product of the political context. As politicians shifts focus from one topic to another, based on events and public opinion. For a model to produce accurate word weights it would have to reflect and adapt to the changes in the political domain. By considering the temporal aspect of the data, the model can better reflect the actual word partisanship. To evaluate the development of word weights across time, we split our dataset into timeframes. However, because smaller timeframes results in more noisy data, the model might over-emphasize words that are not particularly partisan, or vice versa. The lack of information in the model can however be alleviated by introducing prior values based on previous word counts.

Informative prior. In some cases it may be useful to use prior knowledge about word distributions, to prevent the model from overfitting on words that score high values due to random variations. This is not the case in the example presented in figure 3.4, as these

results are a product of running the model on the entire dataset. However, in this section, we will be analysing subsets of the dataset divided into timeframes. In this case, the amount of information which the models are based on is smaller, this can lead to high fluctuations in the resulting word weights. We can alleviate the increase in variability, by using prior knowledge about word distributions. This is however a trade-off, on how adaptable the model becomes to new word distributions. Consider figure 3.5 where the model scores for the words *frihed* (freedom) and *ulighed* (inequality) are plotted based on word counts from 3-month time frames. We can see on figure 3.4 that these words score as being highly partisan when calculated from the entire dataset but their partisanship is not represented in every timeframe.

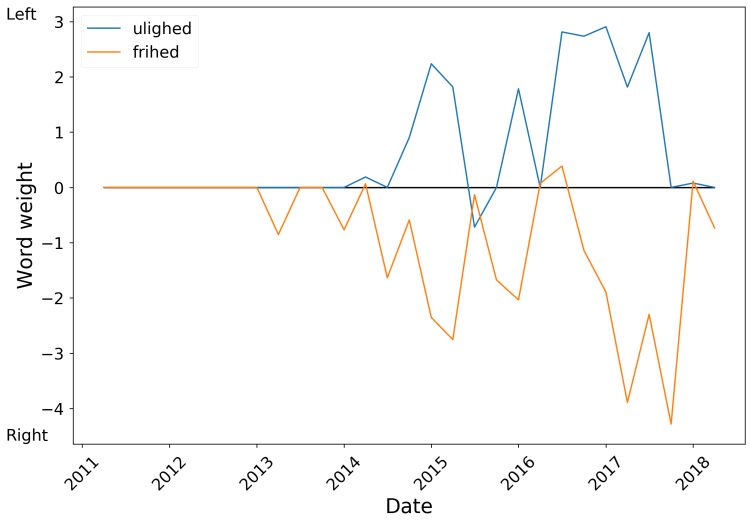


Figure 3.5: Values produced for the words *ulighed*(inequality) and *frihed*(freedom) when applying the model based approach to word counts, divided into 3 month timeframes.

because of the small time frames on which the model results are based, the model over adapts due to the lack of information.

Splitting on time. By applying a prior based on previous information, the model can maintain its adaptability while remaining stable. Figure 3.6, shows the result of using the model with a prior based on word counts from the previous 3-12 months. Because the amount of tweets can fluctuate between time frames, the prior is scaled to the values of the current period's word count, such that the prior does not dominate the current period's results. This is done multiplying the priors word proportions, with the total word count of the current period. This is shown in equation (3.12), where $f12_k^i$ is the word proportions of the previous 3-12 months.

$$\alpha_k^i = n_k^i \cdot f12_k^i \tag{3.12}$$

This method ensures that prior word counts are weighed equally to current word counts, even though the amount of tweets generated might fluctuate across periods. But inconsistencies in word counts between time frames, also needs to be addressed between periods. Consider figure 3.7, which shows the development of the words *løkke* (right-wing prime minister candidate), *helle* (left-wing prime minister candidate) and *regering* (government).

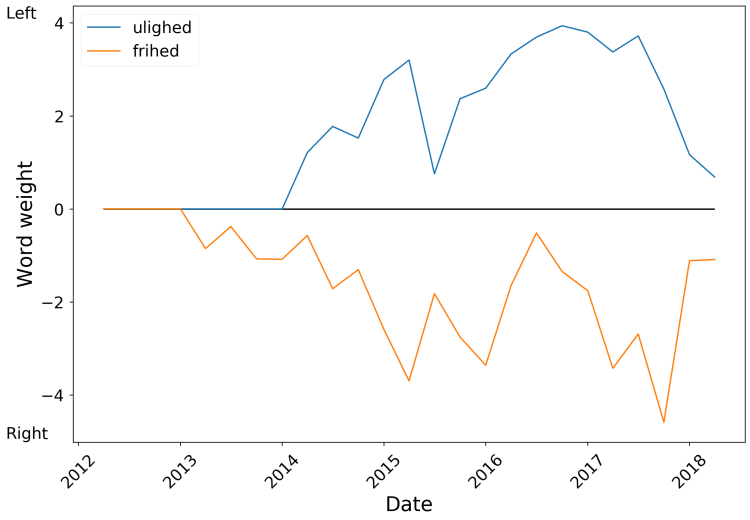


Figure 3.6: Values produced for the words *ulighed*(inequality) and *frihed*(freedom) when applying the model based approach to word counts, divided into 3 month timeframes and a prior based on word counts from from the previous 3-12 months.

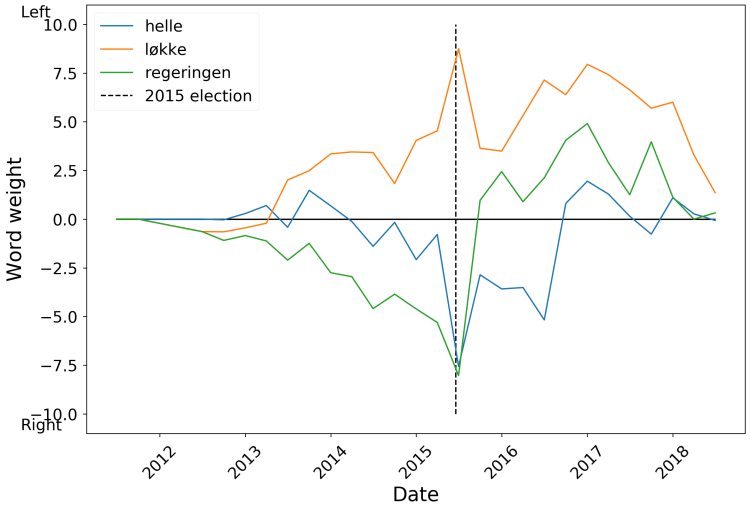


Figure 3.7: Model values produced for the words *løkke*(right-wing prime minister candidate), *helle*(left-wing prime minister candidate) and *regering*(government) when using word counts from 3 month timeframes with a prior based on the previous 3-12 months

These results show a polarization between the two wings during the 2015 government election. the candidates from each wing gain more extreme values during the 2015 election, indicating that each wing is increasing its focus on the opposing candidate in their tweets. Also the word weight of *regering* (government) shifts from being scoring high negative values during a left-wing government, to high positive values when the right-wing takes the governing role. However, if we plot the variance of all the words in the same periods, we can see how spread out the values are for all the words, which indicates how polarized the debate is overall. Figure 3.8, shows the normalized values of, the total word count and the variance of the word weights, for each period.

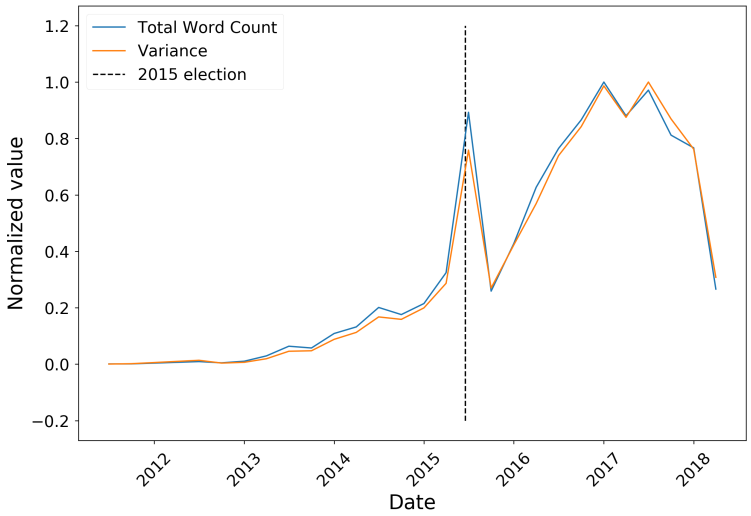


Figure 3.8: Normalized values of the variance of the weights and the total amount of words generated across periods. The periods consists of 3 month timeframes with a prior based on the previous 3-12 months.

It is apparent that the amount of tweets generated in a period affects how high or low the resulting weights become. This can also be derived from equation (3.8) in the model, where the variance dominates the outcome of the z-scores. Periods with many tweets yields less noise than periods with fewer tweets, therefore the certainty for a word being left or right-wing increases. To account for this, so that the results are comparable between periods, the timeframes need to be divided into equally large sets of data.

Splitting on tweets counts. By dividing the dataset into timeframes with an equal amount of tweets, we create windows with dynamic time sizes, but with a more consistent amount of information. Having equally sized periods, also eases the implementation of more controlled smoothing operations, such as moving average and exponential smoothing. Figure 3.9, shows the values of the same words as previously, but with time frames of 2000 tweets and a prior based on 4 previous timeframes with an exponential smoothing of 0.1. These results show that the polarization during the 2015 election, was exacerbated by the increase in certainty, stemming from the increase in tweets in this period.

Note *It is important to note that the resulting curves are dependant of the size of the timeframes, the degree of smoothing and number of prior periods, therefore the results may greatly vary by using other parameters.*

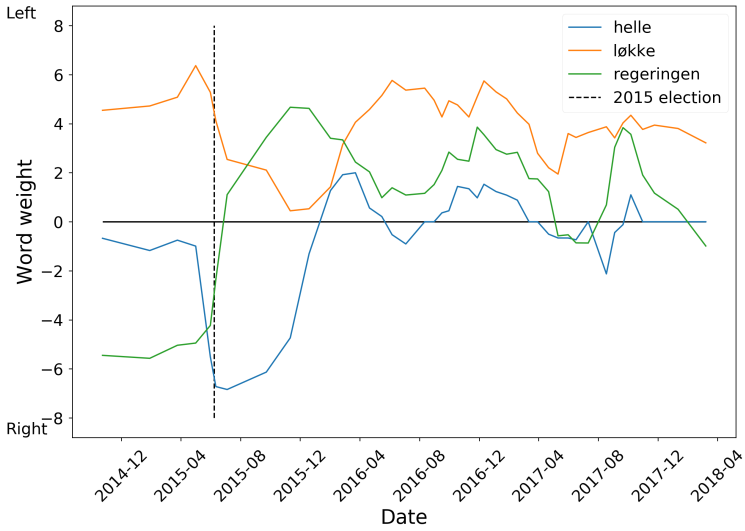


Figure 3.9: Values produced for the words *løkke*(right-wing prime minister candidate), *helle*(left-wing prime minister candidate) and *regering*(government) with a prior based on the 4 prior periods (8000 tweets) but with exponential smoothing on prior periods and a smoothing factor of 0.1

If we again consider the development of the variance, we can see that the values are a better reflection of the actual polarization of the periods, because the amount of words per period is more uniform. However, some periods still contain more words than others, even though the amount of tweets is constant. Adjusting for this would be too cumbersome for too small a gain, therefore we choose to use the current solution in subsequent chapters.

3.7 Summary

We have in this chapter displayed and assessed various methods, of evaluating how representative a word is for different corpora. The chapter has progressively moved from basic formulas that subtracts the word occurrences, to more complex model based approaches that performs much better at finding salient words. We have analyzed the word usage of politicians from different parties, in political discourse on social media. We have shown that these model based approaches, can be used to attribute words with weights that are seemingly representative of partisanship. In order to qualitatively evaluate our results, we have compared some of the most representative words for each political wing, with opinion polls conducted in 2015. By relating our results, to concepts and research from the field of sociology, we have tried to asses the validity of our results. In addition, we examined how the 2-class model, can be transformed into a multi-class solution. In order to produce a model that better reflects, the multi-party scenario that our dataset originates from. Furthermore

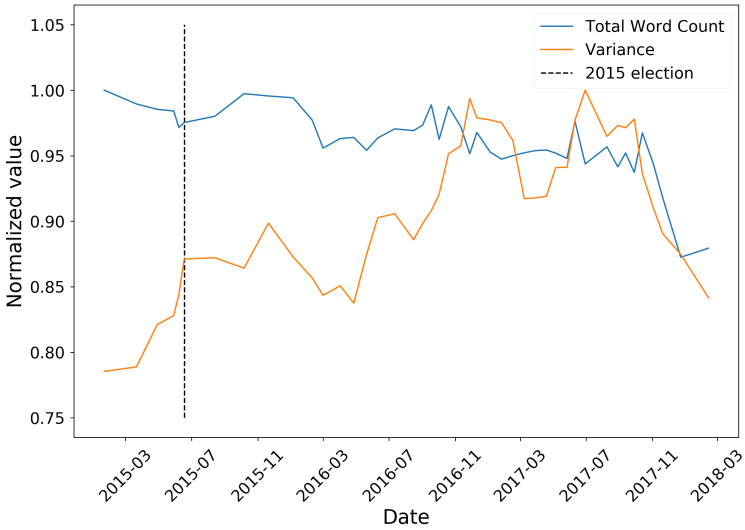


Figure 3.10: Ratios for each equally distributed period of the following variables, the total word count, the sum of the absolute values of the word weights, the variance of the word weights and the total amount of tweets of the period.

we have explored the temporal aspects of our dataset and how this affects the outcome the model based approach. Subsequently, we have proposed solutions to some of the problems that entails, from analyzing the data in a temporal manner. The results produced in this chapter, will be utilized as features for machine learning algorithms, in chapter 5.

Chapter 4

Vector Space Models of Natural Language

In the previous chapter we introduced the concept of representing words in a multi-dimensional vector space, in which each coordinate axis represent a political dimension. We introduced 2 vector spaces, one representing political wings, and a 9-dimensional vector space representing the political parties. Building upon that concept, we will in this chapter introduce the concept of word embeddings and vector space models (VSMs) and present ways of constructing them. Since words carry more information than political partisanship, we will in this chapter focus on capturing information that the previous analysis was unable to. The information we focus on capturing, is the semantic meaning and syntactic structure of natural language. We also show that the semantics and syntactic structure of natural language varies between domains and that constructing the word embeddings with domain specific data improves them.

We begin by examining some challenges encountered when quantifying natural language and we present the linguistic distributional hypothesis, which will carry over to the concept of encoding words. There are numerous ways to encode natural language and a selection of those methods are presented, listing some of the qualities and shortcomings of the different approaches. Afterwards, a thorough introduction into the history and current state of the art for word embeddings is presented, comparing three algorithms, namely: *Word2Vec*, *GloVe* and *fastText* which are some of the most used. We compare these three algorithms, by training on both the Twitter dataset and a data extract of all Danish Wikipedia articles.

4.1 Challenges with text

There are numerous challenges in dealing with text as input to a mathematical system. So far we haven't discussed methods of text representation and what pros and cons there are. This section describes and discuss some problems we must consider, when encoding unstructured text into mathematical models.

Encoding words. Transforming text into input for use in computers is a difficult problem and many methods have come up over the course of the years, when trying to do exactly that. Two used methods are *label encoding* and *one-hot encoding*. The former indexes words

and encodes them into integers, and the latter constructs a vector that is the size of the feature space, with only one element being `true` and the others `false` for every feature.

Curse of dimensionality. The number of data points needed for acceptable performance grows exponentially in the number of dimensions e.g. if a one-dimensional interval needs N data points for acceptable performance, the corresponding 2-dimensional feature space would require N^2 data points [24]. A large feature space is thus not favourable due to this *curse of dimensionality*. In natural languages the feature space is especially large since the number of words in a language often exceeds 50.000 [25], and by some counts exceeding 188.000 words for Danish [26]. This poses a problem in the encoding phase, since every word can be seen as a feature, especially since some words are used far less than others.

Both one-hot encoding and label encoding are problematic The problem with one-hot encoding, is that it is sparse and treats data as categorical. The sparsity is a problem because each word $v(w) \in W$ is a vector the size of the feature space V, which we just described can exceed 50.000. The problem with the categorical nature of one-hot encoding, is there is no notion of similarity, which as we will show, words do indeed possess. Label encoding creates a seemingly ordinal structure by encoding the data as integers which poses a problem when there is no meaningful ordering of the data, as is the case with words.

Word-sense disambiguation (WSD). WSD is the task of identifying the sense (i.e. meaning) of a word when it has multiple senses (polysemy), and the task of identifying multiple words that have similar meaning (synonymy). This problem of disambiguation was first introduced in 1949 [27]. The SemEval competition [28] which have been held for over a decade shows this is still an open problem. The SemEval competition deals with computational semantic analysis, and is an ongoing series. WSD has the potential to improve many natural language processing (NLP) tasks [29], since these tasks rely on being able to know what the meaning of words are.

4.2 The Distributional Hypothesis

Words in language should not be treated as atomic units, as they are defined by the context they are used in. That we can model language in a statistical sense builds upon the hypothesis that there is a distributional structure, to the way words occur and co-occur in different semantic contexts. This distributional hypothesis has its roots in linguistics [30], and a popular way of referring to the context dependent nature of word distributions is: "You shall know a word by the company it keeps" (Firth, J. R. 1957:11) [31].

Similarity between words can rightly be defined as a measure of how often they occur together. These co-occurrence statistics can be used to model some distribution of how similar words are. As words have multiple degrees of similarity, such a measure would require many dimensions to capture their many similarities. This problem, is captured nicely by the WSD task when evaluating the similarity in a specific context [29]. WSD is a task, that we know from everyday interactions with other people, that the human brain is quite adept at solving, but algorithms have long been unable to adequately solve this problem. The problem, is the encoding of words into a format that is interpretable by the computer, and a format we can perform mathematical operations on.

4.3 Word Vectors

Other methods than one-hot-encoding and label encoding have to be considered to overcome their limitations. To incorporate the knowledge about the statistical, context-dependent and distributional nature of language, another approach to encoding words, known as *word embedding*, can be considered. The process of word embedding is to construct a dense vector representation of the feature space by mapping a vocabulary of words to vectors. The notion of creating vectors to represent words goes back several decades to early information retrieval systems [32, 33, 34]. A dense vector representation where words that are close in semantic values, lie close in the vector space representation has been proposed by [35, 33] and later used in the context of learning these representations using neural networks [36].

Vector space models (VSMs) have a rich history in NLP, but we will focus on a subset of the methods. Some ways of creating word embeddings that are not considered in this thesis are the pointwise mutual information (PMI) matrix, singular value decomposition (svd) and other *count based* methods. The reason for the exclusion is simply that the algorithms have been shown to underperform or perform equally well to current neural network language models (NNLM) [37, 38], which have the added benefits of *online learning*.

Word vectors learned through word embedding algorithms have shown to be useful in NLP applications, ranging from information retrieval tasks, to topic modelling and document classification.It has become possible to train complex distributed word representations with neural networks [36], and the techniques are feasible to run on very large corpora with upwards of 600 billion tokens [39]. Word embeddings has recently seen a surge in popularity after Mikolov et al. published the Word2Vec article in 2013 [40]. The results of which have been widely reproduced in new ways to learn all kinds of VSMs, with specific purposes such as WSD, document classification, etc. [41, 42, 43, 44].

4.4 Word2Vec, GloVe and fastText

Different ways of learning these VSMs have been proposed in [45, 40, 46]. An emphasis on predicting the local context has been the main focus in *Word2Vec* [40] and *fastText* [46]. Whereas, learning the global statistical properties of word co-occurrences in the corpus have been the emphasis in GloVe [45].

Word2Vec was the first of the methods, introducing the continuous bag of words (CBOW) and skip-gram approach in constructing word embeddings [40, 47]. The skip-gram training objective have been shown multiple times to be superior to the CBOW objective [40, 46, 47, 48]. The GloVe method was introduced after Word2Vec and criticizes the opaque introduction of the training objectives. Pennington et al. argues that GloVe finds long term structure in text that only looking at local contexts misses, which the CBOW and skip-gram approaches do [45]. The fastText method is the latest of the three methods and builds upon lessons learned from Word2Vec [46]. fastText does a couple of things different than Word2Vec. The model takes morphology of language into account by representing words as a sum of their character n-grams.

However, looking at the training objectives of the algorithms shows the approaches are not so different. The GloVe algorithm trains a weighted least squares regression model on a word-word co-occurrence matrix with word counts. The cost function is introduced as

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j + \log X_{ij})^2 \tag{4.1}$$

noting that all unsupervised models that learn word vectors, use word co-occurrence statistics of the corpus [45] and thus claim through a derivation of equations that the skip-gram objective is equal to the cost function in equation (4.1). Interestingly another article states that the skip-gram with negative sampling implicitly factorizes a word-context PMI matrix [37]. The objective of the skip-gram model is also stated in [46] where it is given as maximizing the log-likelihood probability of observing a context word.

$$\sum_{t=1}^{T} \sum_{c \in \boldsymbol{C}_t} \log p(w_c \mid w_t) \tag{4.2}$$

The objective presented in (4.2) both reads easier than (4.1) and allows us to define the probability in various ways. In the article introducing fastText, they use the same skip-gram model with negative sampling introduced in Word2Vec [46], shown here for completeness:

$$\log(1 + e^{-s(w_t, w_c)}) + \sum_{n \in N_{t,c}} \log(1 + e^{s(w_t, n)}) \tag{4.3}$$

Transfer Learning. The process of reusing a knowledge learned in one setting to a different setting is called *transfer learning*. In a machine learning context it is, repurposing a model trained on one task to another task. A benefit of transfer learning is the large amount of data, outside the domain we are interested in, that can be used in training and then reapplied in the domain of interest. Both Word2Vec and fastText support *online learning* which we can use for transfer learning. The implementation of the GloVe algorithm does not support online learning, however an algorithm named *Mittens* published in [49] mitigates that by vectorizing the objective function of GloVe. The vectorization allow Mittens to have an additional term that penalizes the squared euclidean distance from an existing to a new embedding.

Evaluation. An evaluation metric of the learned embeddings can be constructed as presented in the 2012 SemEval competition[50], where some word pair $a : b$ is similar to another word pair $c : d$. This can in other terms be thought of as "a is to b as c is to d" [51]. The evaluation measures used by Word2Vec [40] are: a derivation of the second task in the 2012 SemEval competition, and a syntactic word analogy task. The word analogy task consisted of comprehensive test sets, where the goal was to evaluate the syntactic and semantic word relationships [51]. Insofar we know, no such tasks have been created for the Danish language. Since our dataset consists of Danish tweets, there are no commonly used datasets to evaulate the word embeddings that we construct.

Dimensionality reduction. To be able to visualize the word embeddings we need a way to reduce the number of dimensions, since VSMs need a high number of dimensions to capture the richness of language. t-SNE is a dimensionality reduction algorithm that produce good mappings of data from high dimensionality to a much lower dimensional space (2 to 3 dimensions) [52].

4.5 Experimental results

To evaluate the different approaches of Word2Vec, GloVe and fastText we construct word embeddings with each method. To capture and show the context dependent nature of lan-

guage we train three word embeddings on two datasets. The first embedding is trained on all the Danish articles from Wikipedia, the second embedding on the Twitter dataset, and the third embedding is trained first on the Wikipedia articles and then retrained with the Twitter dataset. The hyper-parameters that are configured for the model training are: number of dimensions for the vectors $\vec{x}$, iterations of training i, context window size c, deletion of rare words and the learning rate. The size of the vectors $\vec{x}$ alters how many dimensions that can encode a words syntactic and semantic information, where fewer dimensions would superposition the information stored in a dimension with other information, and more dimensions would allow a more unique representation but also increases complexity. The context window c alters how many "neighbouring" words that are allowed to provide information about the target word, where a more narrow context window loses information about the sentence structure and a too broad context window picks up unrelated information. Deleting rare words can help us reduce the vocabulary size and remove words that are misspellings, but we also construct sentences that are unnatural since we remove some words. Although the unnatural sentences are in the context of a large dataset negligible. Changing the learning rate and number of iterations, only affects how quickly and how much the model is allowed to train for learning the embeddings. Therefore, we will just follow sane defaults from the original authors. The word vectors for each algorithm are constructed with similar hyper-parameters to provide a fair comparison. We use the *Gensim* software framework [53] for training the Word2Vec and fastText embeddings. Gensim does not expose the power exponent α, described in GloVe [45], which is why it is excluded from our hyper-parameter configuration. The hyper-parameter configuration we have chosen for evaluating the 3 models are listed in the following. The number of dimensions for the vectors are $||\vec{x}|| = 300$, the number of iterations is $i = 20$ and the context window is $c = 5$. We only include words that occur more than 50 times in the corpus and the learning rate is 0.1. For Word2Vec and fastText, we have used skip-gram with 10 negative samples. These hyper-parameters were chosen, since those have been shown multiple times to perform well when learning embeddings [40, 46, 45] and because of hardware considerations.

Wikipedia data. Wikipedia is often used as a source of data, given its loose copyright license that allows people to reuse their work. We downloaded a dump of the Danish Wikipedia[1] articles, which is hosted under the Creative Commons license (CC BY-SA 3.0)[2]. The dump was downloaded from the Wikimedia website[3]. We have removed all paragraphs containing non-roman characters and all MediaWiki markups so only the natural language persists.

Similarity measure. To quantitatively calculate similarities between words, a similarity measure must be introduced. The similarity of the words will be calculated using the word vectors from the constructed VSMs. Some of the more common similarity measures that have been shown to perform well for text are: *Jaccard coefficient*, *averaged Kullback-Leibler divergence* and *cosine similarity*, all of which perform comparably on different datasets [55]. Since we use the GloVe, Word2Vec and fastText algorithms we choose to use the cosine similarity measure, due to it being the measure used by Word2Vec, GloVe, and fastText in the original publications [45, 40, 46]. The cosine similarity is calculated as shown in equation

[1] https://da.wikipedia.org/wiki/Forside
[2] https://creativecommons.org/licenses/by-sa/3.0/
[3] The specific subsite the data was downloaded from is the wikimedia dumps listed in [54]

(4.5) which is derived from the dot product shown in equation (4.4).

$$\vec{a} \cdot \vec{b} = \|\vec{a}\|\|\vec{b}\| \cos\theta \tag{4.4}$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|} \tag{4.5}$$

The range of the output from the calculation of the cosine similarity lies in the closed interval $[-1, 1]$, where -1 corresponds to exactly opposite and 1 corresponds to exactly the same.

4.5.1 Word similarity tables

We will now present similarity tables of the three algorithms trained on the two datasets. The tables show the 10 most similar words to a input word. The results of this analysis is used to choose which algorithm we will use in further analyses.

A similarity between a input vector and the rest of the vectors in the given vector space is calculated using the cosine similarity from equation (4.5). In tables 4.1, 4.2 and 4.3 the most similar words to the input word: *rød* (red) is shown along with their corresponding similarity value. The results are given as tuples (w, s) where s is the similarity and w is the word.

Wikipedia VSM. In the case of training only on the Wikipedia dataset the most similar words to the input word *rød* for each algorithm is *kornel* (Cornus), *gul* (yellow) and *gul* (yellow), and the rest of the words are colour related as well as seen in table 4.1.

GloVe	Word2Vec	fastText
(kornel, 0.72)	(gul, 0.76)	(gul, 0.80)
(gul, 0.71)	(blå, 0.70)	(sort, 0.71)
(svingel, 0.71)	(sort, 0.69)	(blå, 0.70)
(blå, 0.68)	(hvid, 0.67)	(hvid, 0.70)
(hvid, 0.67)	(grøn, 0.63)	(grøn, 0.66)
(grøn, 0.65)	(rødt, 0.60)	(rødt, 0.65)
(sort, 0.63)	(farverne, 0.58)	(lyserød, 0.63)
(farverne, 0.63)	(farve, 0.57)	(farverne, 0.60)
(farven, 0.61)	(blåt, 0.56)	(farve, 0.60)
(lyserød, 0.61)	(farver, 0.55)	(blåt, 0.58)

Table 4.1: The 10 most similar words to *rød* for the three different algorithms. These results come from training only on the Wikipedia dataset. The words are all related to colour and flowers.

Twitter VSM. The similarity results obtained when training on the Twitter dataset are seen in table 4.2. The "colour" words from the Wikipedia VSM, stand in contrast to the similar words found when training on the Twitter dataset, where the most similar word for all three algorithms is *blok* (block) and the other words are a mixture of words describing colour and politics.

The clear differences in what words are considered similar, show the contextual nature of language. Words that are unique to the different algorithms that we find interesting from a political perspective are *regering* (government) for GloVe, *socialdemokratisk* and *sf* for Word2Vec and *borgerlig* (bourgeois) and *meningsmåling* (opinion poll) for fastText. These are words that are considered salient in a political setting, and thus the algorithms do not show large differences.

GloVe	Word2Vec	fastText
(blok, 0.67)	(blok, 0.67)	(blok, 0.72)
(blå, 0.54)	(blå, 0.61)	(blå, 0.64)
(regering, 0.39)	(bloks, 0.44)	(bloks, 0.46)
(stue, 0.36)	(stue, 0.40)	(røde, 0.41)
(jakke, 0.31)	(gul, 0.40)	(stue, 0.39)
(grøn, 0.31)	(socialdemokratisk, 0.37)	(rødt, 0.38)
(gul, 0.29)	(voxmeter, 0.36)	(voxmeter, 0.37)
(røde, 0.28)	(røde, 0.34)	(borgerlig, 0.36)
(sort, 0.27)	(rødt, 0.34)	(meningsmåling, 0.35)
(bloks, 0.27)	(sf, 0.33)	(gul, 0.33)

Table 4.2: The 10 most similar words to *rød* for the three different algorithms. These results come from training only on the Twitter dataset. The words mostly relate to politics and we see words such as *meningsmåling* (opinion poll) and *regering* (government).

Wikipedia VSM + Twitter VSM. Going from vectors trained only on the Wikipedia dataset, to vectors trained only on the Twitter dataset, showed that language use differs between contexts. In the similarity table for the retrained vectors in table 4.3, the most similar words are again shown. These vectors were first trained the Wikipedia dataset and afterwards retrained on the Twitter dataset. The results of which shows that new embeddings can be improved by including pretrained vectors. The most similar words have shifted from colours to also include political words. The political words of the Twitter dataset are not lost and we gain information from a much larger dataset in the process.

Training the GloVe, Word2Vec and fastText algorithms on the Danish Wikipedia dataset yielded word vectors that closely match our intuition about semantically equivalent words. When training only on the Twitter dataset, the resulting word vectors look different from the vectors trained on the Wikipedia dataset. Here the similar words match our intuition about political words. Finally, when retraining the vectors on the Twitter dataset after initially training on the Wikipedia dataset, the result is showing a juxtaposition of the two previous embeddings along with some new results.

4.5.2 Visualizing high dimensional VSMs

Using the fastText word embeddings and the t-sne algorithm, we will show a representation of the words shown in the similarity tables along with other related words. The word vectors are reduced from 300 dimensions to only 2 dimensions. The result is shown in figure 4.1, which shows the words being clustered into several distinct clusters. On the face of it there are three clusters concerning politics, three clusters are about colours and one cluster about

GloVe	Word2Vec	fastText
(blok, 0.82)	(blok, 0.71)	(blok, 0.70)
(blå, 0.76)	(blå, 0.64)	(blå, 0.60)
(stue, 0.55)	(bloks, 0.49)	(stue, 0.48)
(bloks, 0.53)	(gul, 0.46)	(bloks, 0.47)
(regering, 0.51)	(stue, 0.45)	(rødlig, 0.44)
(grøn, 0.47)	(voxmeter, 0.42)	(rødlige, 0.41)
(finanslov, 0.46)	(lyseblå, 0.40)	(rødbrune, 0.41)
(gul, 0.45)	(grøn, 0.39)	(gul, 0.41)
(splittet, 0.45)	(rødt, 0.39)	(lyseblå, 0.41)
(farverne, 0.44)	(kløver, 0.38)	(borgerlig, 0.39)

Table 4.3: The 10 most similar words to *rød* for the three different algorithms. These results come from training on the Wikipedia dataset and retraining on the Twitter dataset. In this case there are more words describing colour than when only training on the Twitter dataset but there are also political words which show the vectors have shifted towards the Twitter context.

flowers. The three clusters of politics have distinct interpretations, where one is about opinion polls, another about ideology and one about office and government. Interestingly, the colour clusters have been separated, where the one containing colours and words that are also used to define political wings in Danish politics, is separate from other colours.

4.6 Discussion

We have now demonstrated the context dependent nature of language and the importance of using domain specific data, when constructing word embeddings with unsupervised algorithms. In this section, we discuss the differences and similarities of the algorithms, and argue for the selection of algorithm for the subsequent analyses in the thesis.

The results show that fastText weighs morphological representations of words. This is in contrast to GloVe and Word2Vec, both of which are restricted to process whole words. The difference is demonstrated when looking at different conjugations of *rød* (red) which are *røde* and *rødt* as well as words containing *rød* such as *rødlig* and *rødbrune* being more similar for fastText than the GloVe and Word2Vec algorithms. This is due to the subword information calculated in the fastText algorithm, which incorporates intra-word information into the processing of text corpora instead of strictly focusing on inter-word information, as is the case with GloVe and Word2Vec. Since the fastText algorithm calculates subword information it can provide an estimated word vector even for words that are out-of-vocabulary (OOV).

The performance results of the three algorithms are not adequately different to call for further analysis. However, fastText allows OOV words to be inferred, thus constituting a significant advantage over GloVe and Word2Vec. Hence, we will use fastText in our further analyses.

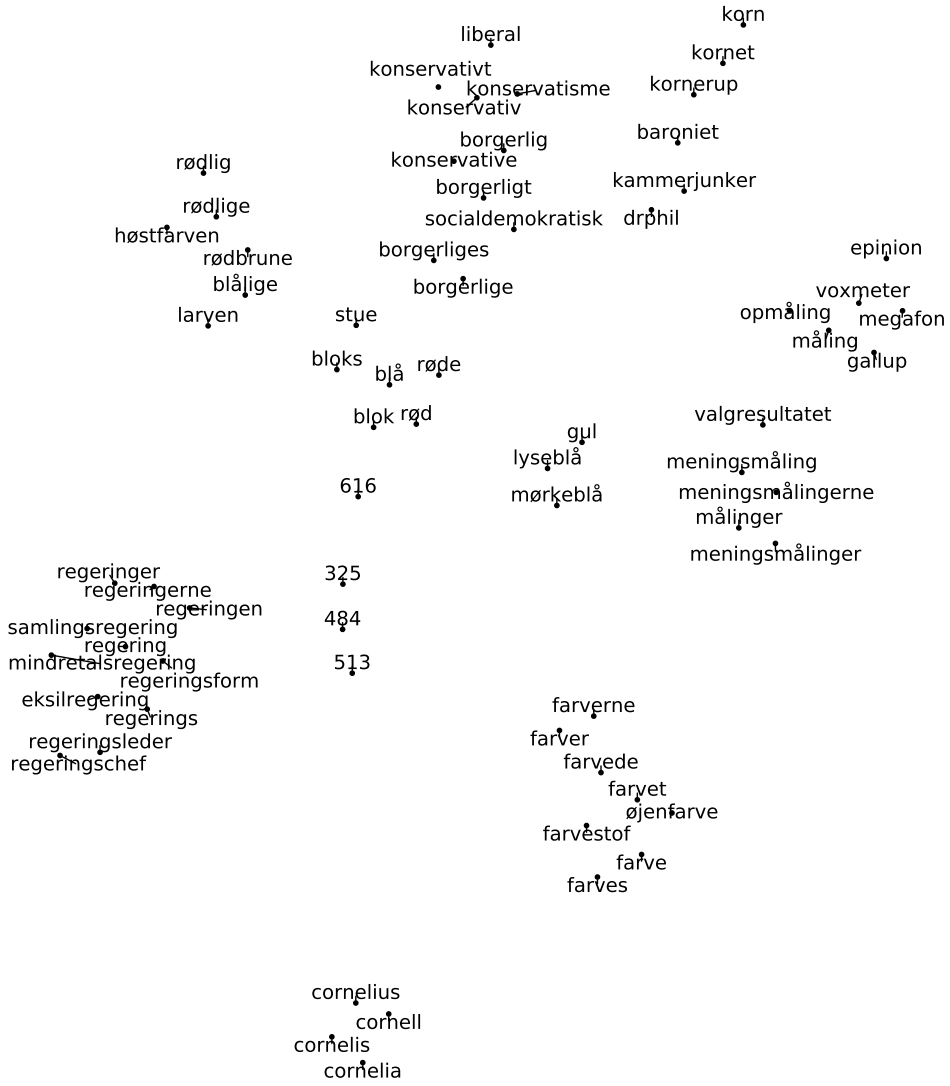


Figure 4.1: 2-dimensional representation of the final 300 dimensional vector space showing the words relative position. The representation is created with t-SNE and the plot shows a number of clusters, with *regering* (government) like words on the left, colours and flowers in the bottom and political words in distinct clusters on both the right and in the top. The selection of words that are presented here are chosen from the similarity tables along with other words that had high cosine similarity to the selected words.

4.7 Summary

In this chapter we introduced the concepts of vector space models (VSMs) and word embeddings. Word embeddings are useful in natural language processing (NLP) tasks, because they are able to capture the context-dependent nature of language, which alleviates some of the challenges in other methods of text encoding. Similarities between words can be found using co-occurrence statistics of large text corpora, which is useful for unsupervised data-driven language understanding. In the previous chapter, we constructed a political VSM of words. The theory and results of this chapter demonstrates the viability of constructing such a model.

We presented a comparison between three algorithms, *Word2Vec*, *GloVe* and *fastText*. We concluded that they mostly perform on par, but fastText allows for out-of-vocabulary (OOV) words to be inferred from subword information, which is why we use fastText in our further analyses.

Large, freely available, online corpora of text allow us to use transfer learning for training more robust word embeddings, which can then be fitted to learn the specific domain of a given dataset.

We also presented a way to visualize the word embeddings by using a dimensionality reduction algorithm such as *t-SNE*, and demonstrated how the visual results mapped to the similarity tables' results.

Chapter 5

Classifying Political Text

In the previous chapters, we showed ways to extract face-valid features of linguistic differences in text documents. Until now, the validity of these features have been based on qualitative assessments. We will in this chapter, test these features on learning algorithms, such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN), to classify the political leaning of a tweet. We show the validity of the features developed in the previous chapters, by including and excluding the features in the classification algorithms. We assess the features by showing the difference in classification accuracy, precision, recall and F1 score. Lastly, we will be exploring the diachronic nature of our dataset in an online learning scenario.

5.1 Introduction

Using learning algorithms for classifying political text, allows us to create models that infer the political affiliation from a text document, but the challenge of using machine learning algorithms to find these complex structures, is the potential loss of interpretability. Machine learning algorithms are notorious for being black box processes, in which, assessing what drives the resulting outcome is opaque. This is problematic in our case, because it is important to be able to asses which features determines the output when working with classification of political partisanship. Transparency is important for ethical reasons, but also because the validity of the results are partly determined by their ability to be qualitatively assessed. The strength of machine learning algorithms, is that they can detect patterns and correlations which are inconspicuous to most people. However, these correlations might not be representative of what would generally be considered, as being indicative of political leaning. The level of opaqueness depends entirely on the chosen algorithm, and some algorithms offer more clarity, regarding which features that drive the results. A selection of popular and interpretable algorithms are: *logistic regression*, *decision trees* and *neural networks*. Logistic regression has coefficients that can be analyzed to determine the effect of a feature. Decision trees can be plotted, and their thresholds evaluated, to see which feature values leads to the different results. However, these methods becomes less interpretable as the amount of features increases. NLP problems often have a large number of features, and the transparency of these algorithms quickly becomes diluted, by the complexity brought on by the many features.

Neural networks have a method of highlighting important aspects of input, via a mechanism called attention. This method is not intended as a way of visualizing the most important features, but it can be used to do so both in images [56] and text [57], in a way that is intuitive and interpretable. Neural networks have also shown in recent years to perform well on a variety of classification tasks. These improvements stem from several reasons, but one of the most influential changes, is that the computational power on parallelizable problems, has increased significantly in the last decades. As every neuron in a layer of a neural network, can compute its output independently, the problem can be parallelized onto different *warps*[1] or *thread blocks*[2] on a GPU.

Neural networks have also received a lot of attention in recent years, because new solutions have solved some of the issues, that the algorithms faced in earlier years. These different solutions and adaptations of neural networks, has led it to become a large and malleable class of machine learning algorithms, which can be modelled to solve different sorts of problems, using a large variety inputs. This is both a curse and a gift, as the amount of different hyper-parameters and model designs, increases the complexity of finding an optimal solution.

We will in the following paragraphs, introduce the two types of neural networks that are used for the tests in this chapter, namely Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN).

RNNs. The concepts of RNNs, were first introduced by Hopsfield [58] in the 1980's, but have until recently not found much use, due to the problem of *vaninshing gradients*[3]. Vanishing gradients is a predicament that affects deep neural networks in general. However, RNN's are especially subject to the vanishing gradient problem, because every time step in an RNN, corresponds to a layer in a traditional feed-forward neural network. This causes them to have many layers in most use cases. In figure 5.1, we show a graphical representation of how a RNN, models the input data in sequence. At each timestep t, the RNN takes as input both x_t and h_{t-1}, computes h_t, which is fed to h_{t+1}.

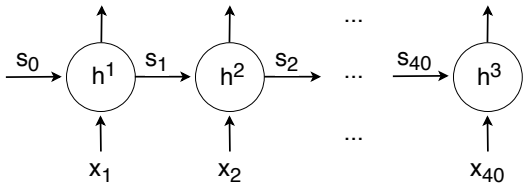


Figure 5.1: A graphical representation of a RNN where we "unroll" the recursive time-step. The input $\vec{x}$ is processed by incorporating the information in the state $\vec{h}$ for each time-step where the information in state h_{t_0} is passed to h_{t_1} in the next time-step. The output of the RNN at each iteration is omitted for brevity.

[1]Warps are the smallest unit in a GPU that can be assigned a specific computation, lower level threads all need to perform uniform calculations.

[2]Thread blocks are a collection of warps on a GPU

[3]When neural networks become deep enough, they suffer from a concept known as vanishing gradients. Vanishing gradients occurs because non-linear activation functions scale the input to a fixed range, when the chain rule of differentiation is applied through many layers, the activation functions 'vanishes' the gradients. This hinders the later layers to correct for gradients in earlier layers and stops the neural network from learning.

However, since the notion of gated units was introduced [59], RNNs such as the Long-short-term-memory (LSTM) and Gated-Recurrent-Units (GRU), have proven to be increasingly interesting for solving classification of time sequence data. These types of models contains gated units, which allows the model to learn what types of information is important to discard or propagate throughout the network. Because the LSTM and GRU models can propagate important information unchanged through the network, they can learn dependencies that are several time steps apart. This property, has made these kinds of neural networks very useful in many cases where there is a temporal aspect to data sequences. However, because these types of neural networks process data sequentially, they are not parallelizable to the same degree as feed-forward neural networks. However, parallel processing power can be leveraged by training on *batches*[4] of data, instead on one input at a time.

CNNs. Historically, CNNs were applied to analyzing visual imagery, but recent work shows they are effective at classifying text [9, 60, 61, 62]. The CNN architecture is similar to a standard feed-forward neural network, but with two additional kinds of layers: *convolutional* and *pooling*. In contrast to the RNNs, the CNNs do not model state temporally. Instead, a CNN computes on the entire input at once, which is well suited for capturing the information in an image, or other synchronic input.

Why CNNs perform well at text classification, is explored by Collobert et al. [63]. Conceptually, we can think of the input text as pictures. If we model the text as a 2-dimensional input, we can exploit the CNNs ability to find abstract patterns. Modelling the input text as a fixed-size $m \times n$ matrix, we can in an abstract sense represent the text as a picture. A downside to this approach, is that the input text must be a fixed size, restricting us to have equal length input. Why a restriction to the input being equal length is a downside, is the varying length of the text input e.g. in our case a tweet can range from 1 word to 40. In figure 5.2, we show the concept of representing text as pictures by illustrating a 40×300 matrix, where each row correspond to a encoded word and the columns each represent a dimension the word is embedded in.

5.2 Embeddings

We have throughout the previous chapters, developed several features that we will test in this chapter. The features will be evaluated as word embeddings, in both RNN models and CNN models. For comparison, we will evaluate the results against models, using a pre-trained embedding for the Danish language, which is trained on texts from Danish Wikipedia pages [46]. Lastly, we will explore the possibility of combining both the semantic and syntactic information from traditional word embeddings, with word embeddings constructed from the weights, analyzed in chapter 3.

Note *We will use the term 'political' loosely in this chapter. We will sometimes use the term to distinguish between what the different features are intended to capture. We do not assume that the features referred to as political are absolute or universal reflections of politics. We would also like to state, that the method applied to calculate the 'political weight' of a word, is not restricted to a political setting. In fact, this method could be applied to any*

[4] Batches are randomly picked subsets of the data

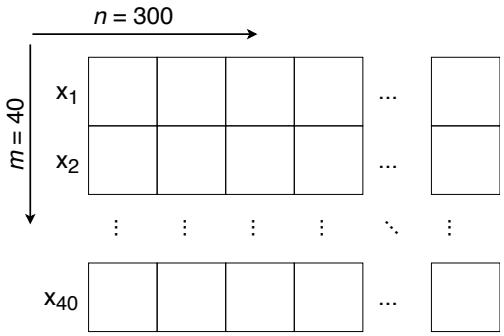


Figure 5.2: Matrix representation of a text input x after embedding. The matrix is a 40×300-dimensional $\mathbb{R} \times \mathbb{R}$ matrix where each cell corresponds to a word's value along a dimension. The number of columns are only 300 in the case where we use the fastText embedding algorithm. The number of columns in the *fighting words* embedding, calculated in chapter 3, are one of $\{2, 9, 49\}$ depending on which relation we show. when using our embedding.

scenario where corpora could be attributed to a class, and one would like to calculate how representative a word is for each class, in corpora.

In the following, we list the different embeddings that will be tested in this chapter and how we will reference them. We have shortened fastText to *FT* and fighting words to *FW* for brevity.

- *Pretrained FT* - A pre-trained Danish word embedding published by the fastText authors.
- *Retrained FT* - An embedding trained on Danish Wikipedia pages using the fastText algorithm and subsequently retrained on tweets from politicians.
- *FW:One vs All* - An embedding constructed from political tweets of politicians, as described in chapter 3, calculated between a party and all other parties.
- *FW:One vs One* - An embedding constructed from political tweets of politicians, as described in chapter 3, calculated between each party pair.
- *Combined embedding* - combined syntactic/semantic embedding with a political embedding.

For the following comparisons, fighting word values are only calculated for plain text elements, discarding the potential signal gain from other elements such as URLs, hashtags and user-mentions. This decision stems from the desire to make this a fair comparison between embeddings, as the fastText pre-trained embedding does not contain information on the aforementioned elements. Therefore both the fighting word weights and retrained fastText embeddings are only based on the plain text elements of tweets.

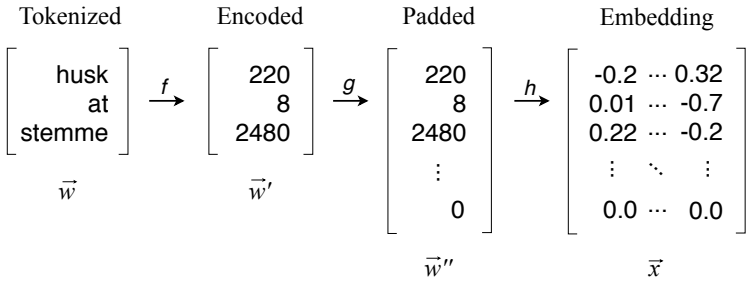


Figure 5.3: Depiction of how we transform a vector of tokens $\vec{w}$ into a word embedding $\vec{x}$. The word embedding is a matrix of size 40×300, which is mapped from the vector $\vec{w''}$. Note that a vector of tokens is always transformed into a 40×300 dimensional matrix, where values from the padded vector $\vec{w}''$ that are 0 are all mapped to 0.0 in the word embedding's vector space.

From text to embedding. Our data transformation pipeline, from tokenized text to a word embedding matrix, as depicted in figure 5.3 is described as follows. Let $\vec{w} \in \mathbf{W}$ be a word vector in the set of words, let $f : \mathbf{W} \to \mathbb{N}$ be the function $f(w_i) = \mathbb{N}_i$ that encodes a vector $\vec{w}$ into a natural number. We encode $\vec{w}$ by mapping the words in $\mathbf{W}$ into integers, and we define $\vec{w'}$ as the encoded vector resulting from applying $\vec{w} \xrightarrow{f} \vec{w'}$. We construct $\vec{w''}$ as a fixed-size vector of size 40 by a mapping $g : \mathbb{N} \to \mathbb{R}^2$:

$$\vec{w''} \begin{cases} \vec{w'} \frown \vec{v} & \text{if } ||\vec{w'}|| < 40 \\ (w_1, w_2, ..., w_{40}) & \text{if } ||\vec{w'}|| \geq 40 \end{cases}$$

where $\vec{v} = \vec{0}$ is a zero-vector of size $40 - ||\vec{w'}||$. As a final step, we define a mapping $g : \mathbb{N} \to \mathbb{R}$ such that $\vec{y} \xrightarrow{g} \vec{x}$. Where g operates on a already constructed word embedding in our set of word embeddings. The word embedding $\vec{x}$ is visualized as a matrix in figure 5.2 and this is how the input to the neural networks looks like.

Since we have multiple word embeddings that have been constructed in the different chapters we explain in the following paragraphs which word embeddings are used in the neural networks.

Fighting words. In the following section, we will explore the potential of using the word weights developed in chapter 3 as word embeddings. There are two different classification tasks that will be examined in this chapter: predicting the political party of a tweet, or predicting the political wing of a tweet. Depending on which of the aforementioned scenarios is being examined, the dimensions of the word embedding will vary. For classifying parties, the fighting word model is extended to accomodate the multi-party problem, using both methods described in chapter 3 section 3.4.1. The two different methods will be transformed into individual embeddings and evaluated separately. Depending on which multi-party extension is used, the dimensions will be either n or $\frac{n \cdot (n-1)}{2}$, where n is the number of party labels to classify. The fighting word vectors are scaled to values between -1 to 1, such that they can be processed efficiently by the models.

Retrained word embedding. We briefly recap the final word embeddings that were constructed and shown in chapter 4. The fastText word embedding we use in the neural networks is the one trained on Wikipedia + Twitter. This retrained embedding is referred to as *Retrained FT* in the results tables. We showed that the retrained embedding had a more domain specific context than the embedding learned only from Wikipedia articles. The benefit of using the retrained embedding over the Wikipedia embedding is two fold. One, we gain a more domain specific embedding that has better semantic and syntactical information for the current domain. Two, we expand the vocabulary to include more words, which is a large benefit due to fewer words being out-of-vocabulary (OOV).

Combined embeddings. We hypothesize that using both of the previously mentioned embeddings, *fighting words* and the *retrained FT*, as a layer in the neural network would improve the model. Utilizing the benefits of an unsupervised syntactical/semantic embedding, with a supervised politically weighed embedding, the resulting models will be able to capture patterns that would otherwise be more complicated to detect. We show the results of combining these embeddings in the experimental results.

5.3 Test Setup

In the following section we explain how the experimental results are obtained, with what models and parameters. The classification task is run on the Twitter dataset, with the objective of correctly classifying which political party a tweet belongs to. We go into detail on how we created the models and explain why we have chosen the architectures for the neural networks that we have.

5.3.1 Data

The dataset that is used throughout this chapter, is the subset of user-specified political tweets described in detail in chapter 2. The subset of political tweets is extracted by using a set of hashtags that we have defined as being political, but not biased towards any specific party. We choose to operate on this dataset, as the purpose of these tests is to compare whether the features developed in chapters 3 and 4, can be used to predict political leaning of text. Each tweet is labelled to the same party as its author, and the authors political affiliation is therefore used as the ground truth of the political leaning of the tweets. The labels used for the classification tasks presented in this chapter, is therefore based on some assumptions, about the relation between politicians and the content of their tweets, that might not always hold true. There is no guarantee that a tweet from a politician, is a reflection of the corresponding party's ideology, nor is it bound to be politically polarizing or even political at all. Some tweets are related to politics in a broader sense, but are however non-partisan. While working with this dataset, we have encountered examples, such as tweets where political candidates encourages users on Twitter, to vote for the local elections. These types of texts cannot be considered representative of ideological leaning or partisanship, and the dataset therefore contains some noise.

The dataset has for the following analyses been split into a 80% training and 20% test set, using random sampling. We have decided not to use a sampling method that ensures an even distribution of classes in the train/test-split, such as stratification. This decision is due to the relatively large amount of samples, compared to number of classes in the dataset. As

we have shown in previous chapters, the distribution of samples within each class is equal within an order of magnitude. Therefore, by random sampling, all classes should be well represented in both the train and the test set of the data.

The weights of the different embeddings have been created using the train data, of the train/test-split. To ensure that the dataset does not contain duplicates, all retweets have been removed, reducing the dataset to $62,393$ unique tweets. Unlike previous analyses, hashtags user-mentions and URLs have not been separated from the remaining content, in order not to deconstruct patterns in the tweet texts. However, when constructing the fighting word embedding and retraining the fastText embedding, we do separate Twitter specific elements and URLs from the dataset. This means that even though these elements will be present in the training corpora, they will be given a neutral value by the embeddings. The aforementioned elements are not included into the embeddings, because they are Twitter specific elements that the pretrained Wikipedia embedding most likely does not contain in it's vocabulary. If we were to include these elements in the embeddings, it would make the comparison less legitimate.

5.3.2 Model Designs

We will in this section, cover some of the choices that was made for the models used to compare the different embeddings. Some of the choices made applies to both RNN and CNN, however some of the hyper-parameters and architectural designs differ, as these are two different neural network types. Optimization of hyper-parameters is also approached in two separate ways for the two algorithms. We will first, list some of the design choices that are common for both algorithms, then move on to the differences between the two model types.

The neural network models will both use the *rmsprop* optimizer [64], this optimizer belongs to a category of adaptive gradient descent algorithms that has been shown to handle diminishing learning rates well. There are others, more recently developed, types of adaptive gradient descent algorithms, such as the *Adam* optimizer [65], which have shown to outperform the rmsrprop algorithm. However, we have chosen to use rmsprop for the following comparisons, because it is a more established algorithm that we are more familiar with.

For both neural networks the loss will be calculated using *categorical cross entropy*, which is the most appropriate loss function in a multi-class classification task of nominal classes. We could categorize the political parties as ordinal, since they are most often represented on a 1-dimensional political spectrum. But for the purposes of this classification task we will not consider constructing a ordinal version of the categorical cross entropy.

The approach that was used to construct the architectures of the CNN and the RNN were different. For the RNN we specify a simple model design that is used to compare the different embeddings. For the CNN we have chosen to follow the architectural design and recommended hyper-parameters shown in [66], which has shown good results on classifying text. There are several reasons for these different approaches which we will cover in the following paragraphs.

RNN. We decide to keep the design simple with the RNN model, because the sequential processing that is inherent to these types of networks, leads to long training times. Therefore, by keeping the design simple, the hyper-parameter optimization and tests can be performed, without being too resource demanding. RNNs have in many cases shown to be able to solve

text classification problems, therefore we asses that even a simple model can be used to evaluate the validity of the features that we will test.

The RNN design used is a simple model as illustrated in figure 5.4. The first layer of the model, consists of an embedding which will be exchanged between tests. The second layer, will consist of an LSTM network that processes the embedded input sequentially. The output is then fed to a dense layer with a softmax activation function with units equal to the number of classes to predict, in this case 9. This final dense layer then gives a probability distribution of the class prediction.

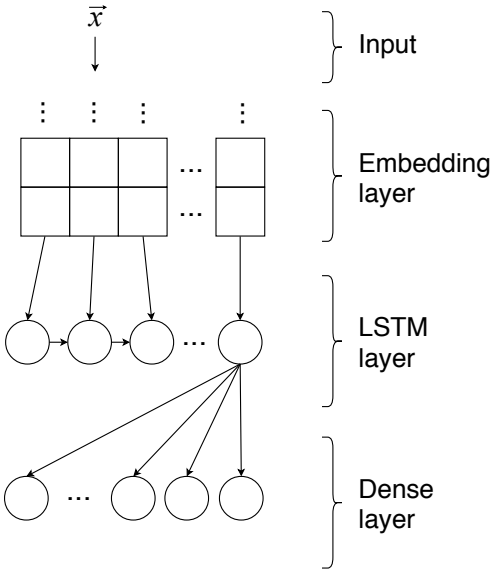


Figure 5.4: Representation of the test model structure. For the different test there will be used different embeddings, followed by an LSTM of equal size for each test, at last a dense layer with a *softmax activation function* will produce a prediction score from 0-1, for each label.

CNN. Convolutional networks are traditionally not associated with NLP problems and is often linked with other classification tasks, such as image processing. However, as mentioned in previous sections, these types of neural networks has in some cases also been applied on text classification and shown good results. Because the processing of text is less intuitive for CNN models, we have chosen to base the testing model on designs that have shown to work on text classification problems. We use an architecture for the CNN, which has been shown empirically to perform well on text classification tasks [66]. These types of neural networks also allow for shorter training times compared to RNN models, due to the parallelization of the processing. Using CNNs therefore allows for more complicated model structures, without being too time consuming.

Hyper-parameters. Fine tuning the hyper-parameters for a machine learning algorithm, has by some researchers been referred to as a "black art that requires expert experiences" [67]. As previously mentioned, this is especially true for neural networks that have

many hyper-parameters. Some research has tried to create guidelines for some model structures, given some specific type of task. For example in [68], where they try to establish rules of thumb, for hyper-parameters and design choices on deep LSTM networks, on a set common linguistic sequence tagging tasks. However, it is difficult to establish universally applicable guidelines to every parameter, as they conclude in [68] "hyperparameters can influence each other and the individual best options must not necessarily be the global optimum". This is exacerbated by the fact that every dataset and learning task is different and might entail other results.

To compare the different embeddings on a fair basis, we will do a *grid search* on the two parameters, *number of hidden states* and *number of epochs*, on the simple LSTM model visualized in figure 5.4. For the CNN, we will only perform a single dimensional grid search, as the only variable that needs to be optimized is the number of epochs. Performing this grid search is time consuming and we have therefore made two decisions to limit the amount of resources spent on hyper-parameter tuning.

- Hyper-parameters are only optimized for the pretrained FT embedding model, but used on all models, as this is the model that we evaluate our results against.
- We will not find optimal values for the *batch size* hyper-parameter, instead we will use a batch size of 128.

The decision not to optimize the batch size, is due to batch sizes having a big impact on training times. Furthermore, adding a new dimension to the grid search, would grow the amount of runs needed, significantly. However, it has in some cases been shown that batch size can have an effect on performance and that large batch sizes can lead to sub-optimal solutions [69]. This is due the "noise" brought on by adjusting the weights based on a small amount of samples, can help the algorithm escape local minima. In [69] however, they establish this as being an issue with batch sizes larger than 512, whereas we use a batch size of 128. However, as previously stated, these thresholds could vary between dataset, design choices and tasks.

RNN. In table 5.1, we display the resulting mean accuracies and standard deviation of performing a grid search, using a 5-fold cross-validation on the simple LSTM model. The epochs are increased by 5 for each iteration and the amount of hidden states is doubled per iteration The ranges have been chosen based on what we asses as being reasonable, given our knowledge of working with the dataset and hardware limitations. Iterations with the lowest scores have been omitted due to limits in the size of the table.

The resulting hyper-parameters chosen for the comparisons, is an LSTM with 128 hidden states, which is trained over 15 epochs. 15 epochs is quite a low amount for such a complex problem. However, keep in mind that the model for this comparison is quite simple and is therefore prone to quickly overfitting to the dataset. If the model was consisting of multiple layers with dropout or regularization, the model could probably train for longer and possibly achieve better results. However, for a comparison and evaluation of the constructed embeddings' performance, we asses that it is best to keep the design simple. The results of training the models with different embedding layers on the train dataset and evaluating their performance on the test set, is covered in section 5.4 and can be seen in table 5.4.

CNN. We perform a 5-fold cross-validation on the number of epochs for the CNN. This is the only hyper-parameter that we will fine-tune. We run the hyper-parameter search in

epochs	16	32	64	**128**
10	25.9% (+/- 0.7%)	27.5% (+/- 0.4%)	29.1% (+/- 0.2%)	29.8% (+/- 0.4%)
15	25.9% (+/- 1.3%)	28.9% (+/- 0.6%)	29.7% (+/- 0.6%)	31.0% (+/- 0.5%)
20	27.0% (+/- 0.5%)	29.1% (+/- 0.8%)	30.1% (+/- 0.6%)	30.3% (+/- 1.2%)
25	27.1% (+/- 0.7%)	29.3% (+/- 0.3%)	30.0% (+/- 0.8%)	29.6% (+/- 0.4%)
30	27.6% (+/- 0.6%)	28.9% (+/- 0.5%)	29.2% (+/- 0.6%)	29.2% (+/- 0.3%)
35	27.2% (+/- 0.8%)	28.9% (+/- 0.5%)	29.1% (+/- 0.6%)	28.4% (+/- 0.6%)

Table 5.1: Mean Accuracy and standard deviation of running a 5-fold cross-validation with increasing epochs and number of hidden states. The best performing parameters are highlighted in bold font. Performing the grid search took several days, therefore doing this operation for every model or with more hyper-parameters was not possible given the hardware and time available.

the range $[5, 50]$ in increments of 5. The difference in accuracy across the different epochs is low, suggesting that either a large increase in the number of epochs is needed or that training is saturated. The accuracy for a specific number of epochs can be seen in figure 5.2

epochs	accuracy
5	27.53% (+/- 1.62%)
10	27.77% (+/- 0.99%)
15	28.35% (+/- 1.12%)
20	28.26% (+/- 1.24%)
25	28.38% (+/- 0.58%)
30	27.76% (+/- 1.17%)
35	**28.91% (+/- 0.79%)**
40	28.45% (+/- 0.53%)
45	28.79% (+/- 0.52%)
50	28.22% (+/- 0.77%)

Table 5.2: Mean Accuracy and standard deviation of running a 5-fold cross-validation with increasing epochs. The best performing parameters and result are highlighted in bold font.

5.4 Experimental Results

This section covers the results of evaluating the CNN and RNN models on the test data set, using the various embeddings. The tests have been performed using the previously specified hyper-parameters and model design choices. Common for both CNN and RNN, is that we will evaluate each of the embeddings separately. Subsequently, the highest scoring semantic/syntactic embedding, is combined with the highest scoring politically weighed embedding, to *explore*[5] the potential of concatenating these embedding types.

[5]The choice of combined embedding, is based on the test results. Therefore we employ the term explore, because this decision is subject to data leakage.

Note *We are aware that choosing a candidate for the comparison, based on the test results, borders on data contamination. However, time restrictions did not allow us to run every combination of embeddings.*

In table 5.3, we display the distribution of the classes in the test dataset. By simply guessing the majority class, a model could achieve 21.83% accuracy.

Party	support	percentage
Alternativet	908	7.29%
Dansk Folkeparti	900	7.23%
Enhedslisten	747	5.99%
Liberal Alliance	892	7.16%
Radikale Venstre	1790	14.37%
Konservative Folkeparti	1471	11.81%
Socialdemokratiet	2719	21.83%
Socialistisk Folkeparti	809	6.5%
Venstre	2216	17.8%
total	12452	

Table 5.3: The distribution of classes in the test set. Some classes are more represented in the dataset than others. However, the support for each class does not differ differ by more than one order of magnitude

RNN. The results of evaluating the RNN model on the test set, can be seen in table 5.4. The table displays the mean accuracy and standard deviation of performing 10 tests with the different embeddings.

Embedding layer	Test accuracy	Train accuracy
Fighting words:One-vs-All	30.17% (+/- 0.47%)	36%
Fighting words:One-vs-One	30.96% (+/- 0.47%)	36%
Pretrained FT	31.84% (+/- 0.65%)	43%
Retrained FT	33.05% (+/- 0.57%)	47%
Retrained FT + FW:One-vs-One	33.93% (+/- 0.83%)	53%

Table 5.4: Test results from different embeddings using an RNN model. The accuracies depicted are the mean accuracy and standard deviation of running 10 tests, where the model is trained on the train set and evaluated on the test set.

If we exclude the combined embedding, the results show that the RNN model scores the highest results when using the retrained word-embedding. But that there is added gain, by combining the potentials of using an unsupervised syntactic/semantic embedding, together with a supervised class weighed embedding. The combined embedding has the highest mean accuracy, but also the highest amount of variation and training accuracy. This suggests that the model is more prone to overfit to the training set, compared to the other models. The models using Fighting-Word embeddings, had lower test accuracies, than any of the other

embeddings. However, these models also have the lowest training accuracies. This indicates that these models could have continued training for more epochs, without overfiting to the data. Remember that the amount of epochs was determined by a 5-fold cross-validation on the Pretrained FT embedding. Because the Retrained FT embedding is partly based on the same corpus as the Pretrained FT, using the same algorithm, it is likely that they would converge somewhat at the same rate. As opposed to the Fighting-Word embeddings, which represent another type of approach to word-embeddings.

The results also tend to favour the higher dimensional embeddings, which in turn, also suffer from the highest degree of overfitting. The added complexity of higher dimensional input, allows for the model to detect more patterns, at the cost of potentially overfitting to false signals.

However, accuracy is not always an appropriate metric, as the accuracy can be falsely skewed by the class imbalances. In table 5.5, we display: F1-score, precision and recall for each of the classes. The measurements stem from best scoring model, the combined *Retrained FT + FW:One-vs-One* model, and represents the mean values and standard deviation of the 10 tests performed. From these metrics we can see that the model has a

Party	F1-score	Precision	Recall
Alternativet	0.26 (+/- 0.01)	0.29 (+/- 0.02)	0.25 (+/- 0.04)
Dansk Folkeparti	0.28 (+/- 0.03)	0.35 (+/- 0.09)	0.25 (+/- 0.05)
Enhedslisten	0.37 (+/- 0.01)	0.46 (+/- 0.08)	0.32 (+/- 0.05)
Liberal Alliance	0.25 (+/- 0.03)	0.33 (+/- 0.06)	0.23 (+/- 0.08)
Radikale Venstre	0.31 (+/- 0.03)	0.33 (+/- 0.05)	0.32 (+/- 0.08)
Konservative Folkeparti	0.30 (+/- 0.02)	0.34 (+/- 0.03)	0.27 (+/- 0.05)
Socialdemokratiet	0.43 (+/- 0.02)	0.37 (+/- 0.03)	0.55 (+/- 0.11)
Socialistisk Folkeparti	0.18 (+/- 0.01)	0.26 (+/- 0.05)	0.15 (+/- 0.03)
Venstre	0.35 (+/- 0.04)	0.41 (+/- 0.06)	0.34 (+/- 0.10)
avg.	0.33 (+/- 0.01)	0.36 (+/- 0.01)	0.34 (+/- 0.01)

Table 5.5: In depth metrics of the highest accuracy scoring RNN model. The measures are the mean of 10 tests alongside the standard deviation of the test results.

tendency to favour the larger classes over the smaller ones. yet, the smallest class, *Enhedslisten*, has the highest precision, and a recall that is close to the mean. suggesting, that it does not predict this class particularly often, but when it does it often predicts correctly. Whereas, the second smallest party, *Socialistisk Folkeparti*, yields both the lowest recall and precision. Indicating that the model rarely predicts this class and with low accuracy. This is interesting because the party *Socialistisk Folkeparti*, is ideologically close to the most represented party *Socialdemokratiet*, which is the class that yields the highest recall. One answer could be that the model often picks *Socialdemokratiet*, because *Socialdemokratiet* is more represented and it has a hard time distinguishing the two classes from each other. An analogy of Danish politics.

CNN. In contrast to the RNN, the CNN does not exhibit any pattern to whether the fastText embedding or the fighting words embedding is better. One pattern we do see, is the fastText embeddings have a much higher training accuracy, which suggests the model

can overfit to these values. The embedding that performs the best is again the combined embedding, and the results can be seen in figure 5.6.

	Test accuracy	Train accuracy
Pretrained FT	29.72% (+/- 0.43%)	96%
Fighting words:One-vs-One	30.25% (+/- 0.71%)	43%
Retrained FT	30.97% (+/- 0.65%)	97%
Fighting words:One-vs-All	31.61% (+/- 1.12%)	43%
Retrained FT + FW:One-vs-All	31.97% (+/- 0.62%)	97%

Table 5.6: Test results from different embeddings using an CNN model. The accuracies depicted are the mean accuracy and standard deviation of running 10 tests, where the model is trained on the train set and evaluated on the test set.

As the RNN model, the CNN model also has the highest precision for Enhedslisten. In contrast to the RNN model, the CNN model has low recall for Enhedslisten. Socialdemokratiet still has the highest F1-score which might be due to that class being the most prevalent in the class distribution.

	F1-score	Precision	Recall
Alternativet	0.23% (+/- 0.01%)	0.27% (+/- 0.03%)	0.20% (+/- 0.03%)
Dansk Folkeparti	0.24% (+/- 0.02%)	0.33% (+/- 0.04%)	0.19% (+/- 0.04%)
Enhedslisten	0.31% (+/- 0.02%)	0.42% (+/- 0.04%)	0.25% (+/- 0.04%)
Liberal Alliance	0.21% (+/- 0.02%)	0.30% (+/- 0.05%)	0.17% (+/- 0.04%)
Radikale Venstre	0.28% (+/- 0.04%)	0.32% (+/- 0.03%)	0.26% (+/- 0.08%)
Konservative Folkeparti	0.30% (+/- 0.01%)	0.30% (+/- 0.04%)	0.33% (+/- 0.08%)
Socialdemokratiet	0.40% (+/- 0.01%)	0.35% (+/- 0.03%)	0.49% (+/- 0.11%)
Socialistisk Folkeparti	0.16% (+/- 0.02%)	0.24% (+/- 0.04%)	0.13% (+/- 0.04%)
Venstre	0.36% (+/- 0.05%)	0.36% (+/- 0.04%)	0.40% (+/- 0.12%)
avg.	0.30% (+/- 0.01%)	0.33% (+/- 0.01%)	0.32% (+/- 0.01%)

Table 5.7: In depth metrics of the highest accuracy scoring CNN model. The measures are the mean of 10 tests alongside the standard deviation of the test results.

The results of comparing the word embeddings across different learning algorithms suggest, that the values calculated in chapter 3 are indeed indicative of political affiliation. Modelling the domain-dependent features of the language also provide valuable information to the algorithms, which we can see by using the retrained fastText embedding. Combining the word embedding of the political fighting words embedding with the unsupervised retrained fastText word embedding, we gain an even higher accuracy. We argue that this indicates a confirmation of our hypothesis that: *utilizing the benefits of an unsupervised syntactical/semantic embedding, with a supervised politically weighed embedding, the resulting models will be able to capture patterns that would otherwise be more complicated to detect.*

In summary, the best perfoming model is the RNN with: the fighting words embedding with weights between all classes, and the retrained domain specific semantic/syntactic word embedding.

5.5 Online learning

The following section, will explore the possibility of learning patterns in political tweets gradually, in a chronological order. Thereby resembling a more lifelike scenario, where tweets are gradually recorded, then used for training a model. Which is in turn used to classify future tweets. This analysis will not adhere to the same principles of a controlled experiment, as the previous section. This is because the results are not intended to be comparable with the results of the previous section, it is therefore important that they are not perceived as such. For this scenario, we will employ both a semantic/syntactic embedding, *Retrained FT*, and a politically weighed embedding, in the test models. For each iteration, the politically weighed embeddings will be updated according to the recent tweets, thereby becoming a dynamic word embedding.

Note *The approach presented in this section cannot be classified as online learning in the strictest sense, as this would suggest that both the model and the embedding would be updated for each new input. This is not the case in our test scenario. Instead, the word weights and neural network model, are updated/trained on all tweets from the current timeframe, in order to classify tweets in the subsequent timeframe.*

As described in chapter 3, the political weight of a word can change over time. In this example we will examine the potential of having an adaptive embedding, in a neural network model. The neural network is gradually trained on chronologically ordered tweets, using a dynamic political embedding and a static syntactic/semantic embedding.

To create such a scenario, the dataset has been separated into chronological, equally sized, timeframes of tweets. Each timeframe is used to further train a model, which will then try to classify tweets from the following timeframe. On each iteration of timeframes, a new fighting word embedding is constructed, using current word counts and prior word counts from previous timeframes.

The amount of tweets stemming from a specific party can vary between timeframes and can result in some parties not being represented in certain timeframes. Adjusting for this, would to some extent, defeat the purpose of performing a lifelike test scenario. However, to ensure a more equal distribution of labels across timeframes, the following tests are performed on a 2-class classification problem. The party labels of the dataset are transformed into wings[6]. For this test we include all elements of a tweet in the political embedding, thereby attributing word weights to hashtags, user-mentions and URLs. We do this because this test is a Twitter specific scenario, where the purpose is to construct a model that can predict the political wing of a tweet author.

In figure 5.5, we show the mean accuracy and standard deviation of running 10 runs using timeframes of 1000 tweets. A new model is initiated for each run and trains on the timeframes in a chronological order. On each training iteration, it predicts the tweets of the next timeframe, which is the measured accuracy.

The measures show that the models gradually learn the more timeframes they iterate through. However, we can again see a correlation between our results and the 2015 government election. After the government election, where the governing party lost to the opposition, the model drops in accuracy. In chapter 3 section 3.6, we explored the shifting

[6]For a definition of the political wings, see chapter 3, table 3.1

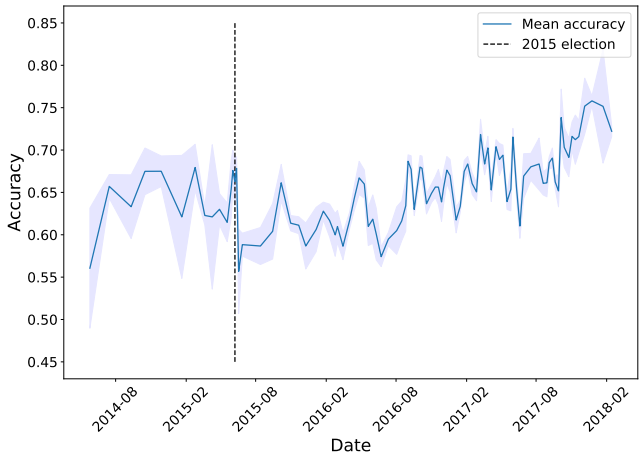


Figure 5.5: The mean accuracy and standard deviation of running 10 iterations of an online learning model with timeframes consisting of 1000 tweets. The fighting words are calculated based on the word counts from the current timerame and a prior based on previous 5 time frames with a smoothing factor of 0.1.

political weight of words across timeframes. This analysis showed that the government election correlated with an increase in polarization. This polarization, was followed by a shift in political weight for the word *regering* (government) and the names of the prime-minister candidates. Since the models employ an embedding with weights that are constructed from the same algorithm, it might not be so surprising that the model is affected by this change in the political domain. However, this indicates that the model relies on signal that is dependant of the political context. As stated previously, it can be difficult to interpret which features, a machine learning model derives it's predictions from. The patterns that determine the output of the model, might not be desirable depending on what we intend the model to capture. That the model is affected by and adjusts to shifts in the political domain, indicates that the model is emphasizing features that are relevant. We can also see from the results that even though the mean accuracy fluctuates, the standard deviation does not vary so much. This suggests that the fluctuations are consistent across test runs, therefore the fluctuations are most likely due to partisanship of tweets being more predictable in some periods than others.

5.6 Summary

In this chapter, we introduced different machine learning algorithms and argued why we have chosen neural networks as the basis for our tests. We introduced two neural network architectures: Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN). We have elaborated on their strengths and weaknesses and why these algorithms were chosen. We have constructed word embeddings using methods covered in chapter 3 and 4. We have subsequently, compared these embeddings with a pretrained word embedding, using both RNN and CNN models, in order to evaluate their performance. In order to compare the embeddings on a fair basis, we have optimized model hyper-parameters for pretrained embedding through a 5-fold cross-validation. We have also explored the poten-

tial gain of combining an unsupervised semantic/syntactic embedding, with a 'politically' weighed embedding. Our results show that by retraining a word-embedding, with context specific corpora, one can increase a models potential to detect patterns, and achieve better results. We have also shown that by employing an embedding, based on how representative a word is for each class, models can also achieve approximately as good results as traditional word-embeddings. However, the best results yield from combining a traditional semantic/syntactic word embedding, with a class representative embedding. From our results, RNN models seem to model the complexity of language better than CNNs for predicting political text on Twitter. However, this analysis would require a more in-depth comparison between different and more complex architectures to really evaluate the differences in RNN and CNN performance. The model that scored the highest accuracy was an RNN model utilizing both a retrained semantic/syntactic domain specific word embedding, concatenated with a class representative word embedding. This model scored an accuracy of 33.93%. However, in depth analysis of other metric, such as precision, recall and F1 score, showed that the model had a tendency to predict the larger classes in the dataset, due to class imbalances.

We have also exemplified how a model could perform in an online learning scenario. We have shown that such an online model, with a dynamic word embedding, can gradually learn and adapt to changes in the domain it categorizes. The analysis of the online learning model showed that when the government changed after the 2015 election, the accuracy dropped which suggests that the model indeed picks up on the political values in the fighting words embeddings. Having models that can adapt to new input without retraining the entire model are advantageous to forecasting and classification in a lifelike scenario.

Chapter 6

Discussion & Future Research

This chapter concludes our research for this thesis, which has examined patterns in natural language usage of politicians on Twitter. Throughout this thesis we have demonstrated every step of our pipeline, from gathering, labelling and storing large amounts of politically related data, to preprocessing, analyzing, feature extracting, classifying and evaluating our results. By handling every aspect of the process, we have been able to develop specifically tailored, datasets, features and machine learning algorithms for our purpose. Even though our goal has been to examine quantitatively driven methods to analyze political discourse on social media, we have focused on keeping our results qualitatively interpretable and thereby hope that our results can be used by researchers, from other fields than our own. In this chapter we will present the main takeaways and results from the previous chapters of this thesis and discuss what the results can be used for. We also outline what a path for future research might look like.

6.1 Conclusion

Here, we present a brief summary of the previous chapters, to give an overview of what has been covered in this thesis. In chapter 2 we introduced the dataset's characteristics and the process of collecting it. In chapter 3 we showed the linguistic differences across the political groups by using a bayesian maximum-likelihood model. In chapter 4 we showed how we can construct a dense vector space model of natural languages in a unsupervised fashion, and adapt the vector space to domain specific tasks. In chapter 5 we used the results of the previous analyses as features in machine learning algorithms to classify political content. In the following paragraphs, we highlight the most important points made in the chapters that were presented throughout this thesis.

Dataset and statistics. We created a dataset of 1848 Twitter profiles along with their tweet history. The Twitter profiles are politicians from the Danish Parliament and political candidates for the 2017 local elections. The dataset amounts to $584,156$ tweets, of which we have automatically and manually labelled $479,899$ tweets in total, from 951 politicians. The subset of tweets that is user specified as being politically related, amounts to $97,676$ tweets. We introduced a descriptive statistical analysis of the dataset's characteristics, demonstrating the viability of using this dataset, as well as explaining the potential biases

and pitfalls using it. We explained in detail the preprocessing steps performed and various filtering methods that are applied on the dataset.

Linguistic differences in political discourse. Based on political discourse on Twitter of politicians from different parties, we examined different methods to attribute partisan weights to words. Beginning with naive approaches, we progressively moved from non-model based approaches, to more complex models. Gradually, we highlighted the problems that each method entailed and showed solutions to those problems. To evaluate the final model, we compared our results to concepts and research from the field of sociology. We showed that the model is able to attribute political weights to words, that correlates with theory about political strategies and opinion polls conducted on voters. Finally, we examined the temporal aspect of our data, by applying the model to different periods of time. We demonstrated the diachronic nature of the developed partisan word weights, with the focus of employing them as dynamic embeddings. In doing this, we outlined some issues and solutions, to applying the model to data in a temporal manner.

Vector space models. Encoding words as sparse or categorical discards valuable information about natural language, and we showed that utilizing a dense vector representation of words is advantageous. The vector space models (VSMs) builds upon the distributional hypothesis from linguistics, which states that words that co-occurrence of words can be a measure for their similarity. We compared three word embedding algorithms Word2Vec, GloVe and fastText. Demonstrating that constructing VSMs on large text corpora is feasible. The algorithms performance are mostly on par, but fastText has the beneficial feature of being able to use out-of-vocabulary (OOV) words. Furthermore, we introduced the concepts of; transfer learning, which allow us to re-purpose models, that are trained outside of our domain, to our domain; and online learning that allows us to update our models iteratively, which is useful for transfer learning.

Classifying political texts. To asses the features constructed in the previous chapters, we created machine learning models to classify the political party of a tweet. We introduced several machine learning algorithms and presented arguments for the selection of utilizing neural networks. We introduced two types of neural networks, namely: Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). We reasoned why these neural networks were particularly suited for the classification of political tweets. The dataset was randomly split into a train and test set. The features from chapters 3 and 4 were implemented as embedding layers into separate models. In order to evaluate the potential of the developed features, we introduced a comparison model based on a pretrained word embedding for the Danish language. To compare the embeddings, we optimized the hyper-parameters in the comparison model, using cross-validation techniques on the training dataset. These optimized hyper-parameters where then applied to all models, in order not to disadvantage the comparison model. Subsequently, we compared the different word embeddings constructed, on the test dataset. Additionally, we examined the potential gain of combining a semantic/syntactic embedding with a supervised political embedding. Both CNNs and RNNs, showed that combining the semantic/syntactic embedding with the supervised political embedding, resulted in the best performing model. Our analysis suggests that including guided features, such as political weighting of words, results in learning algorithms being able to capture patterns that would otherwise be hard to detect.

Lastly, we demonstrated that creating an online learning model, with dynamic political embeddings as well as static semantic/syntactic embeddings, achieve meaningful results. The results suggested that the model was both affected by, and gradually adapted to, changes in the political domain. However, it is difficult to draw conclusions on which aspects of the data that affects the outcome of the model, without some means of interpreting the process. Implementing such a feature was one of the reasons we picked neural networks as a machine learning algorithm. However, due to time restrictions we weren't able to develop such a mechanism and we will therefore address this point in section 6.3.

6.2 Discusison

Our early results paints a picture of what the landscape of using text-as-data, in political partisanship feature evaluation, looks like. As we have already stated, our domain expertise is within computer science and data analysis. For that reason we will not draw causal conclusions on our results. Instead we will draw broad strokes of what we have discovered.

We split the discussion into two parts, one dealing with our findings on natural language, linguistic differences, data-driven language models and usage of learning algorithms for classification tasks. Another discussing ethical considerations about classifying political affiliation.

On linguistic differences and natural language. Capturing the complexity of natural language is a difficult task. This can be done in a variety of ways and for different purposes, depending on which aspects of language is intended to capture.

In the following list we present conclusions on the diachronic and context-dependent nature of language.

- Natural language evolves through time. A synchronic snapshot is thus not enough and we need to take the diachronic nature of language into consideration.
- Natural language is subject to it's context. Whenever working with natural language, one should always take into consideration that language is highly specific to it's context and cannot just be generalized to outside it's domain.
- We can model large scale text corpora to understand how language is used and in that model extract salient information about which words relate to each other and how they relate.
- Neural network models are well suited for NLP tasks and learning patterns of natural language from scratch, i.e. not incorporating any prior knowledge encoded about the language.

The complexity of natural language is reflected, when trying to create models that classify text into distinct categories. To optimize the performance of a NLP Model, it is beneficial to consider more than one approach to represent and capture the different aspects of language. Due to the diachronic and context specific nature of language, models need to adapt to context and temporal changes. Because natural language is context dependent, a static word embedding is not enough, we need to consider using information from it's specific domain, by e.g. constructing class specific weighted embeddings or retraining a word embedding on the dataset of interest. Likewise, embeddings need to be dynamically updated to account for shifts in the domain they try to map.

On ethics. As political standpoints and politics are sensitive subjects, a due diligence regarding ethical considerations is mandated. From what we see, there are at least three different ways in which trying to predict political affiliation might be a problem. The three problems are: misclassification (1), abuse (2) and fear (3). We will discuss each problem in sequence. (1) Since the methods and models presented here are not 100% accurate, well intending researchers risk misclassifying documents and people, leading to individuals being labelled unfairly. (2) Ill intending or outright malicious actors can use such systems, to target specific groups of people in a harmful way, if they choose to do so. (3) Having systems in place that can predict political affiliation from personal discourse, might unintentionally inhibit individuals' perceived freedom of speech, if they are aware that their utterances are monitored and analyzed.

Being aware of these issues, we can consider what potential steps there might be, in order to alleviate them. To reduce problem one, we can consider which metrics are the most important in the classification performance. In this regard, precision is the most important metric, due to the fact that mislabelling the political standpoint of a text document can have unintended consequences. Achieving high precision is not enough. We must be able to interpret what drives the results of a model, thus we need to create models where we can qualitatively asses the algorithmic decision making process. To reduce the second problem not much can be done from our side other than identifying the problem and openly discuss it with other researchers. The third problem can be alleviated by carefully applying tools in algorithmic decision making, as to not violate the trust of the general public. This is not a problem we can solve single-handedly, but one which must be targeted by the larger community on a political as well as legal and cultural level.

One important point we want to emphasize, is: *automated tools should not be treated as a substitution for common sense and thoughtful research.*

6.3 Future Research

Since computational linguistics is such a huge field, we have only scratched the surface of what we can explore with this dataset. As such, we will outline a path for future research in that area. We do so in the following.

It would be worth exploring utilizing using parts-of-speech tagging and named-entity recognition to give the text more structure before it is used as input to a machine learning algorithm. Topic modelling methods such as Latent Dirichlet Allocation are also worth exploring, to find whether the hashtag topics correspond to roughly what a typical topic modelling algorithm would find.

Furthermore, we propose some other interesting research areas that are worth exploring. In the following list, we present the possibilities that we find worth exploring for creating better and more interpretable machine learning models.

- Common-sense relationships of words and sentences that are usually unstated can be incorporated by using knowledge bases to extend the information about natural language use. Such examples are seen in [70, 71, 72]. A technique known as retrofitting can be used to that extent [73]

- Applying other neural network architectures, since we only tested RNNs and CNNs. e.g. the compositional attention network architecture to perform multi-step reasoning [74] should generalise better, or generative adversarial networks [75]. Another

interesting architecture is the denoising auto encoder, where we could use the features extracted in chapter 3 to guide the network in reconstructing the important parts.

- Model text on a sentence-level and a character-level to evaluate what granularity is better [43, 44, 60].
- Explore the attention mechanism of neural networks to gain insights into how the networks weigh each feature upon input. Since the machine learning algorithms are considered black-box prediction algorithms, gaining insights into what drives the output is as important as getting good performance results.

An interesting avenue not explored in this thesis is *network science* which briefly mentioned can be used to analyze the social network of the political actors in the dataset. A large number of studies already show the amount of political information there is in the *retweets*, *user-mentions*, *URLs* and also on the *follower-following* graph of the political actors. We can use this information for a variety of analyses and a few is listed here.

- By assuming the followers of a political actor hold the same political opinions, we could employ semi-supervised learning methods to use unlabelled data points to construct better models that also generalize to more diverse text corpora.
- Incorporating information of interactions by the political actors among their peers can drive feature weighting, by either increasing or lowering the weight of a retweet for certain groups that exhibit behaviour that differentiate them from the rest.

Bibliography

[1] Andreas Jungherr. Twitter in politics: A comprehensive literature review. *SSRN Electronic Journal*, 2014.

[2] Todd Graham, Marcel Broersma, Karin Hazelhoff, and Guido van 't Haar. Between broadcasting political messages and interacting with voters. *Information, Communication & Society*, 16(5):692–716, 2013.

[3] Nigel Jackson and Darren Lilleker. Microblogging, constituency service and impression management: Uk mps and the use of twitter. *The Journal of Legislative Studies*, 17(1):86–105, 2011.

[4] Michael D Conover, Bruno Gonçalves, Jacob Ratkiewicz, Alessandro Flammini, and Filippo Menczer. Predicting the political alignment of twitter users. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 192–199. IEEE, 2011.

[5] Marco Pennacchiotti and Ana-Maria Popescu. A machine learning approach to twitter user classification. *Icwsm*, 11(1):281–288, 2011.

[6] Felix Ming Fai Wong, Chee Wei Tan, Soumya Sen, and Mung Chiang. Quantifying political leaning from tweets and retweets. *ICWSM*, 13:640–649, 2013.

[7] Antoine Boutet, Hyoungshick Kim, Eiko Yoneki, et al. What's in your tweets? i know who you supported in the uk 2010 general election. *ICWSM*, 12:411–414, 2012.

[8] Daniel Xiaodan Zhou, Paul Resnick, and Qiaozhu Mei. Classifying the political leaning of news articles and users from user votes. In *ICWSM*, 2011.

[9] Prashanth Vijayaraghavan, Ivan Sysoev, Soroush Vosoughi, and Deb Roy. Deepstance at semeval-2016 task 6: detecting stance in tweets using character and word-level cnns. *arXiv preprint arXiv:1606.05694*, 2016.

[10] H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, et al. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS one*, 8(9):e73791, 2013.

[11] Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. Fightin' words: Lexical feature selection and evaluation for identifying the content of political conflict. 2008.

[12] Isabel Moreno-Sánchez, Francesc Font-Clos, and Álvaro Corral. Large-scale analysis of zipf's law in english texts. *PloS one*, 11(1):e0147073, 2016.

[13] Christoffer Green-Pedersen. The growing importance of issue competition: The changing nature of party competition in western europe. *Political studies*, 55(3):607–628, 2007.

[14] John R Petrocik. Issue ownership in presidential elections, with a 1980 case study. *American journal of political science*, pages 825–850, 1996.

[15] John Sides. The origins of campaign agendas. *British Journal of Political Science*, 36(3):407–436, 2006.

[16] Wikipedia. Højreorienteret — wikipedia, the free encyclopdia, 2017. [Online; collected 15-May-2018].

[17] Wikipedia. Venstreorienteret — wikipedia,the free encyclopdia, 2018. [Online; collected 15-May-2018].

[18] Lasse Lange. Her er det nye politiske kompas, 2016.

[19] Akiva M Liberman. How much more likely? the implications of odds ratios for probabilities. *American Journal of Evaluation*, 26(2):253–266, 2005.

[20] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[21] Michael Laver, Kenneth Benoit, and John Garry. Extracting policy positions from political texts using words as data. *American Political Science Review*, 97(2):311–331, 2003.

[22] Joseph Berkson. Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, 39(227):357–365, 1944.

[23] Martin Vinæs Larsen, Asmus Leth Olsen, Jens Olav Dahlgaard, and Frederik Georg Hjorth. Meningsmålinger under valgkampen. In *Oprør Fra Udkanten*. Djøf/Jurist-og Økonomforbundet, 2017.

[24] Sergios Theodoridis and Konstantinos Koutroumbas. Chapter 2 - classifiers based on bayes decision theory. In Sergios Theodoridis and Konstantinos Koutroumbas, editors, *Pattern Recognition (Fourth Edition)*, pages 13 – 89. Academic Press, Boston, fourth edition edition, 2009.

[25] Dan Jurafsky and James H Martin. Semantics with dense vectors. In *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*, pages 286–299. Prentice Hall, Pearson Education International, 2009.

[26] Det Danske Sprog og Litteraturselskab. Baggrund og tilblivelse, 2018.

[27] Warren Weaver. Translation. In William N. Locke and A. Donald Boothe, editors, *Machine Translation of Languages*, pages 15–23. MIT Press, Cambridge, MA, 1949/1955. Reprinted from a memorandum written by Weaver in 1949.

[28] Wikipedia. Semeval, 2018.

[29] Dan Jurafsky and James H Martin. Computing with word senses. In *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*, pages 300–325. Prentice Hall, Pearson Education International, 2009.

[30] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[31] J. R. Firth. A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis. Special volume of the Philological Society*, 18(11):11, 1957.

[32] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[33] Hinrich Schütze. Word space. In *Advances in neural information processing systems*, pages 895–902, 1993.

[34] Kevin Lund and Curt Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208, Jun 1996.

[35] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323:533, oct 1986.

[36] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[37] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc., 2014.

[38] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

[39] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[40] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[41] Andrew Trask, Phil Michalak, and John Liu. sense2vec - A fast and accurate method for word sense disambiguation in neural word embeddings. *CoRR*, abs/1511.06388, 2015.

[42] Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W. Cohen. Tweet2vec: Character-based distributed representations for social media. *CoRR*, abs/1605.03481, 2016.

[43] Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb Roy. Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 1041–1044, New York, NY, USA, 2016. ACM.

[44] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[45] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[46] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[47] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[48] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April 2017.

[49] Nicholas Dingwall and Christopher Potts. Mittens: An extension of GloVe for learning domain-specialized representations. In *Human Language Technologies: The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Stroudsburg, PA, March 2018. Association for Computational Linguistics.

[50] David A Jurgens, Peter D Turney, Saif M Mohammad, and Keith J Holyoak. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 356–364. Association for Computational Linguistics, 2012.

[51] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

[52] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[53] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.

[54] Wikimedia Foundation. Wikimedia dumps, 2018.

[55] Anna Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56, 2008.

[56] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[57] Yequan Wang, Minlie Huang, Li Zhao, et al. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, 2016.

[58] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[59] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[60] Xiang Zhang and Yann LeCun. Text understanding from scratch. *CoRR*, abs/1502.01710, 2015.

[61] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[62] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *CoRR*, abs/1404.2188, 2014.

[63] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[64] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[65] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[66] Ye Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820, 2015.

[67] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[68] Nils Reimers and Iryna Gurevych. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*, 2017.

[69] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[70] Robert Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*, pages 4444–4451, 2017.

[71] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. Learning distributed representations of texts and entities from knowledge base. *CoRR*, abs/1705.02494, 2017.

[72] Robert Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686, 2012.

[73] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. *CoRR*, abs/1411.4166, 2014.

[74] Drew A Hudson and Christopher D Manning. Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067*, 2018.

[75] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Appendix A

Data storage

Database storage To store the data we used a NoSQL database hosted on Amazon Web Services (AWS). The reason that a NoSQL database was chosen is due to the simplicity of the data structure. There are two Tables in the database, one containing tweets and one containing twitter profiles. The two are linked together with a foreign key called the *author_id*. The twitter profiles contain all the information available through the twitter API, as described on their website [1]. The twitter profiles additionally contains information about their political party affiliation and their occupation (parliament member, local candidate, parliament replacement). The tweet entries contains all the tweet data and metadata available through the API as described in the tweet-object documentation[2]. When we store the data locally to analyse and process, the information from both tables is merged on a tweet level, such that every tweet object get as an attached profile name, party and occupation. The final set of attributes can be seen in table A.1, which is an example of an entry in the dataset.

[1] developer.twitter.com/en/docs/tweets/data-dictionary/overview/user-object.html
[2] developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json#tweetobject

Attribute	Example
author_id	1227383683
coordinates	None
created_at	2017-11-07 22:24:43
entities	{'urls': [{'expanded_url': 'https://twitter.co...
favorite_count	0
favorited	False
id	928025449812451329
in_reply_to_screen_name	None
in_reply_to_status_id	NaN
in_reply_to_status_id_str	None
in_reply_to_user_id	NaN
in_reply_to_user_id_str	None
is_retweet	False
lang	en
name	Anja Camilla Jensen
party	A. Socialdemokratiet
place	{'country_code': 'DK', 'country': 'Denmark', '...
possibly_sensitive	False
quoted_status	{'in_reply_to_status_id_str': None, 'in_reply_...
quoted_status_id	9.28011e+17
quoted_status_id_str	928010790115962880
retweet_count	0
retweeted	False
retweeted_status	NaN
source	¡a href="http://twitter.com/download/iphone" r...
text	F'ing scary!!!! https://t.co/ZjNE3IpEGx
truncated	False
withheld_copyright	NaN
withheld_in_countries	NaN
withheld_scope	NaN
acc_parties	Socialdemokratiet

Table A.1: The full set of attributes in the dataset, these values are a random example from the dataset

Appendix B

Tf-idf

The term-frequency inverse-document-frequency (tf-idf) method is a simple way of reducing the impact of words that occur in many corpora, without directly adding emphasis to rarely occurring words. The tf-idf transformation variant used in this example is the normalized version,

$$tfidf_{kw}^{(i)} = f_{kw}^{(i)} \cdot \frac{1}{idf} \tag{B.1}$$

where, *idf* is the number of corpora the word appears in. However, utilizing this method on a small collection of texts, each containing many similar words, leads to negligible difference in the resulting term, against the result of the word proportion. This is the case with the parties aggregated into wings, where the amount of corpora is only 2 and most words occur in both corpora. The results of using tf-idf on the dataset yielded little to no difference compared to using simple proportions. The results of using the tf-idf algorithm on our dataset can be seen in figure B.1.

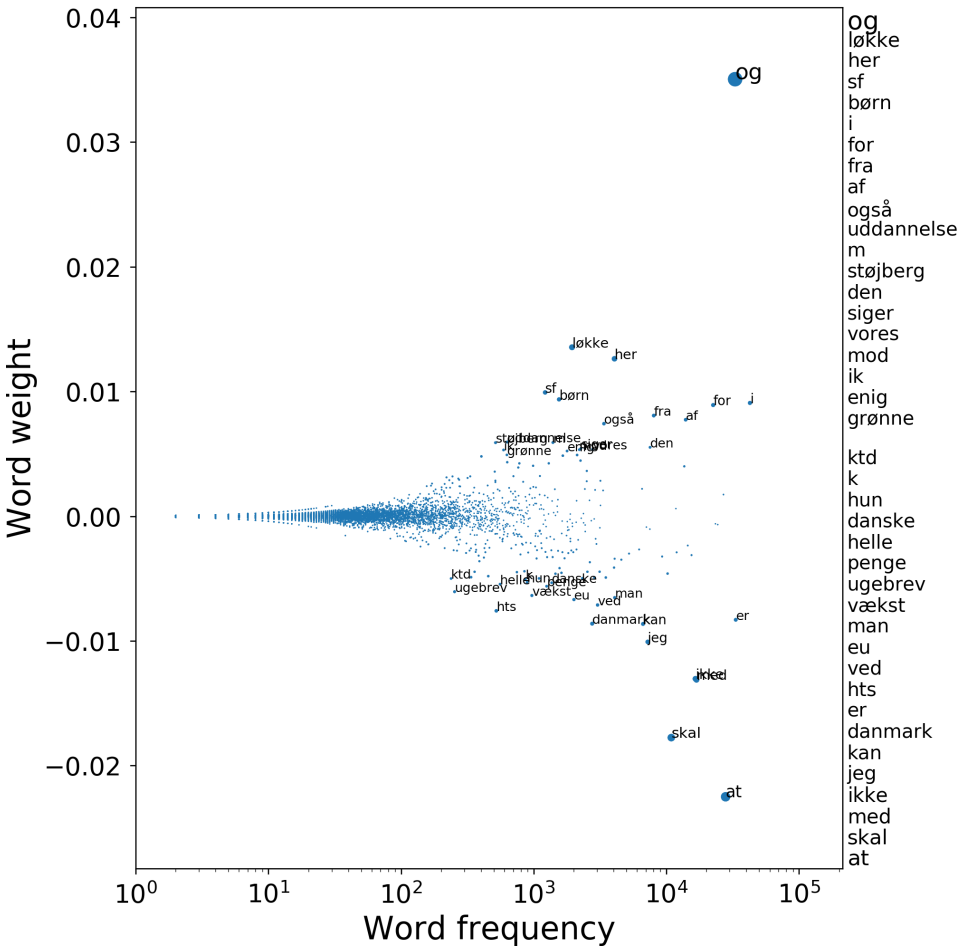


Figure B.1: Results of calculating tf-idf values for each word between the left-wing and right-wing corpora. Subsequently the transformed word counts have been subtracted, such that high positive values indicates left-wing and high negative values indicates right wing. Because tf-idf does not take into account the relative word frequency between corpora, the results resemble those presented when calculating proportions.

Appendix C

Wordscores

The *wordscores* are calculated as,

$$W_{kw}^{(L-R)} = \frac{f_{kw}^{(L)} - f_{kw}^{(R)}}{f_{kw}^{(L)} + f_{kw}^{(R)}} \tag{C.1}$$

The result of applying *wordscores* on the data set can be seen in figure C.1

What can be observed from figure C.1 is that the extremes are dominated by very low frequency words that only occur in one corpus. If we filter away these words as in figure C.2 we can see that even though the extremes show many words that could be considered indicative of political affiliation, they are still low frequency words.

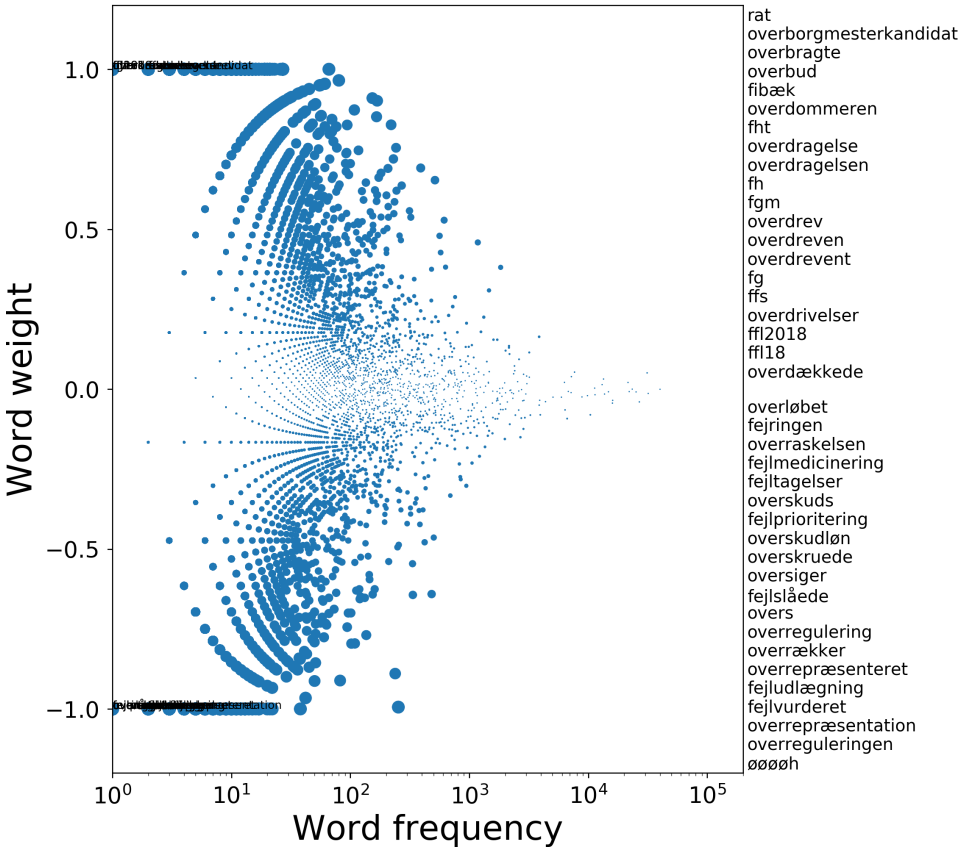


Figure C.1: Wordscores Calculated for the left-wing and right-wing corpora. The extremes are dominated by words that only occur in one corpora.

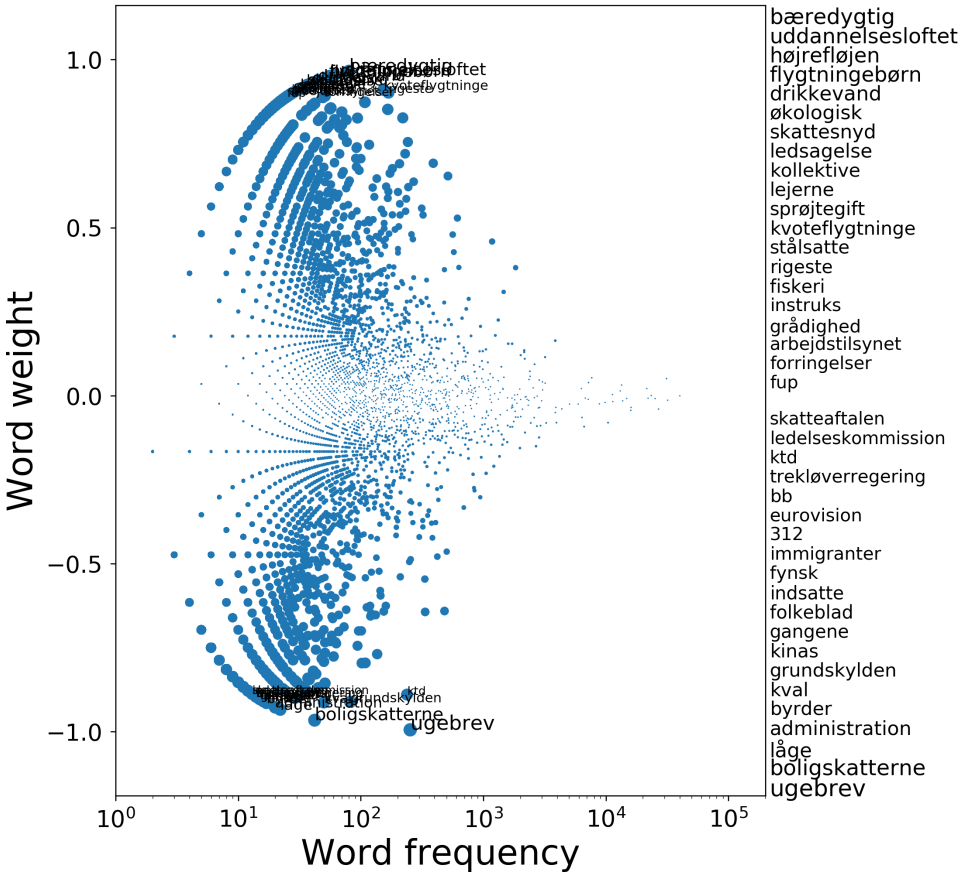


Figure C.2: Wordscores without words that only occur in one corpus. The results show that wordscores do capture word that could be specified as being partisan. However, it is too biased towards low frequency words, neglecting all high frequency words.

Appendix D

User-Mentions, Hashtags & URLs

We have in the previous analyses been focusing on the *words* contained in the tweets, because those are the most generalizable and interpretable results. However, as we display in this appendix, using the model on the *user-mentions*, *hashtags* and *URLs* can give insights into who, what and how Danish politicians communicate on twitter. From a machine learning perspective, this content is also interesting, because these feature sets also display a high degree of polarization between parties from opposing wings on high frequency features.

User-mentions. In figure D.1, we present the models results, when applying it only to the user-mentions used in the tweets. The results show politicians tend to reference politicians or parties from their own wing.

Hashtags. Figure D.2, shows the results when analyzing only the hashtags contained in tweets. Hashtags can be powerful tools to describe partisanship, because it can be used to subscribe a tweet to a topic or to emphasize a point. Both user-mentions and hashtags show a high degree of polarization between the wings in high frequency user-mentions/hashtags. This could indicate that these features could be very good predictors, when trying to classify the political affiliation of tweets.

URLs. Lastly, the model has been applied on the URLs shared in the tweets. The results presented in figure D.3, show that many of the URLs shared by politicians which dominates the extremes, are news media websites. This indicates that politicians from each wing, tend to share news content from specific sources instead of sharing news from multiple sources. Furthermore, the figure shows that the right wing politicians seem more prone to using link shorteners,[1] which can be used to measure and analyze user clicks.

[1] Link shortener websites, such as *Bit.ly* and *Ow.ly* provide metrics on clicks that it recieves. These are used in marketing when trying to analyse user selection clicks.

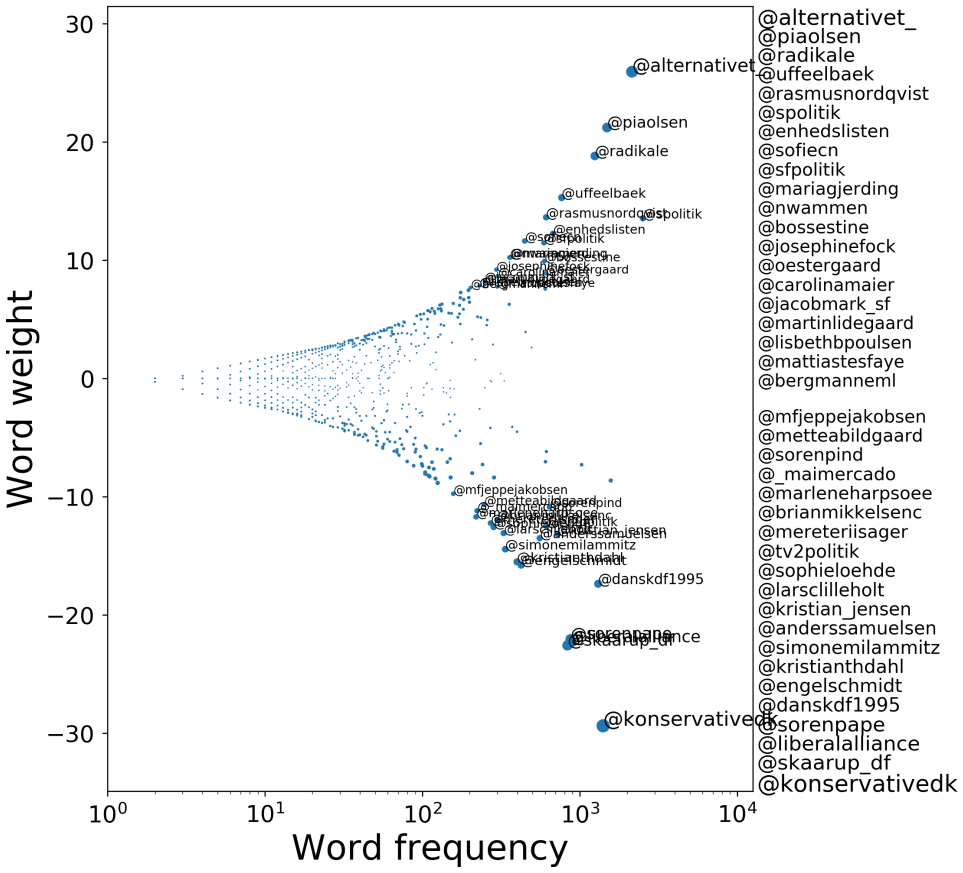


Figure D.1: Model results when applying weights to user-mentions when using an uninformative prior of 1.0e−15. The results show that the left-wing and right-wing tend to reference politicians from their own wing.

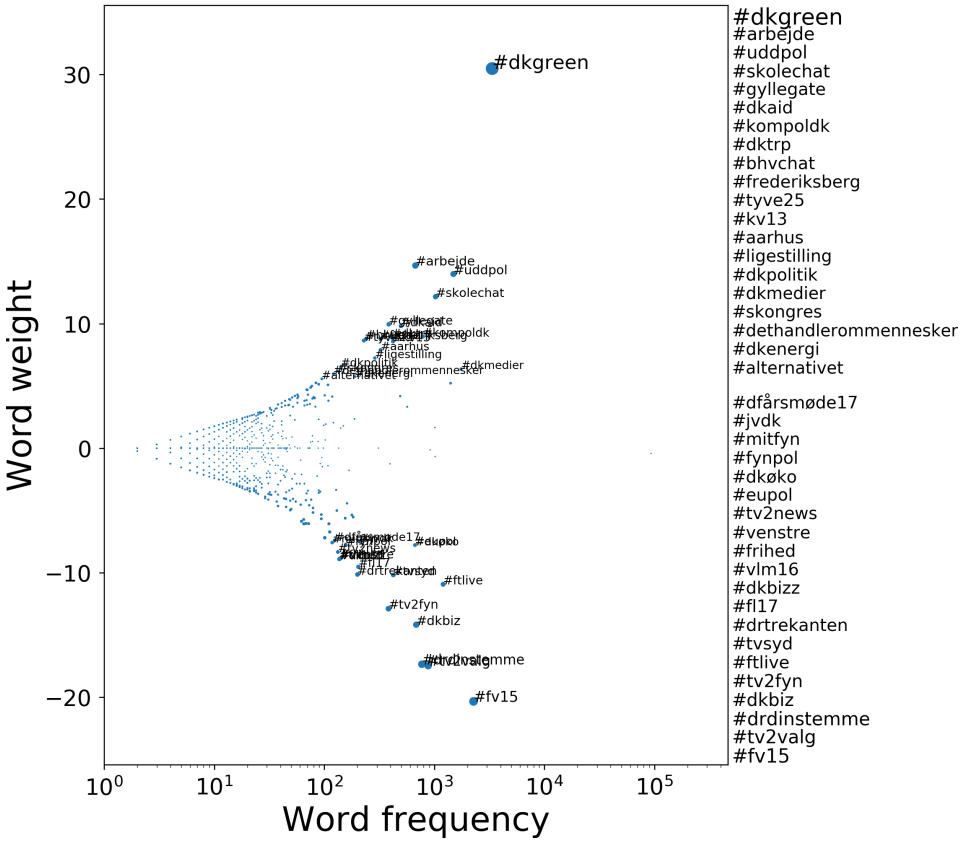


Figure D.2: Results when applying the model to hashtags and using an uninformative prior of 1.0e−15. The results show that hashtags also are highly partisan and certain topics are more referenced than others by both the left-wing and the right-wing.

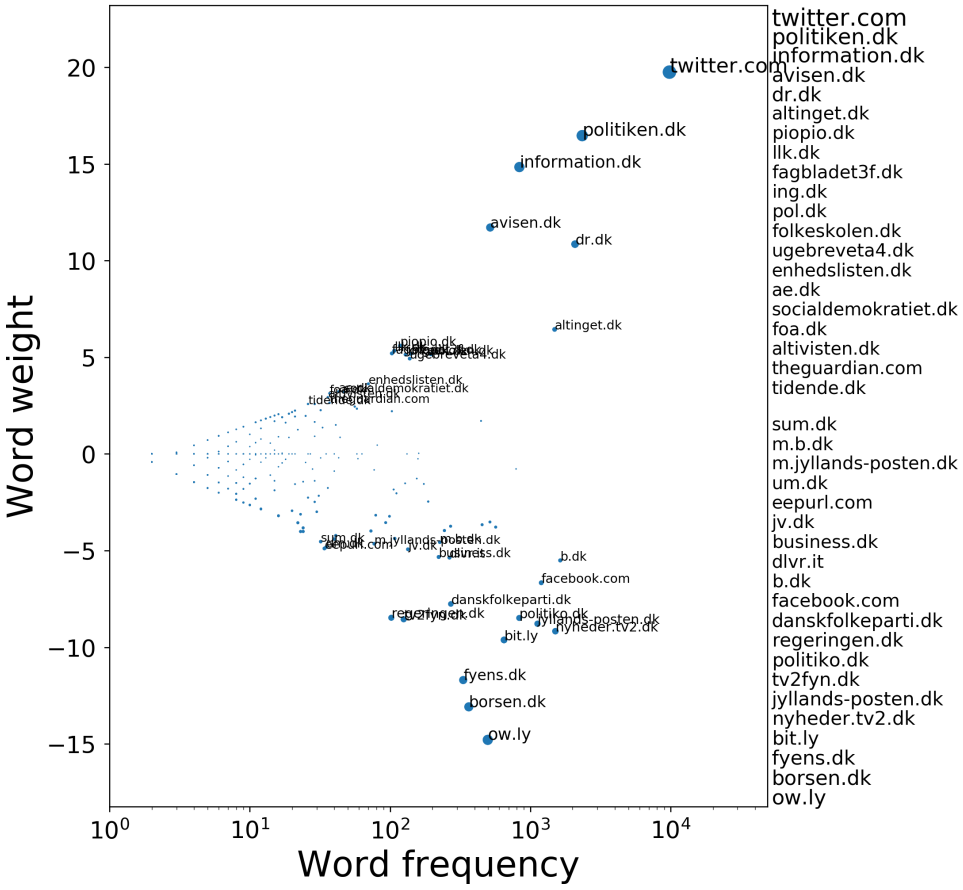


Figure D.3: Model results of URLs when using an uninformative prior of 1.0e−15, the urls have been shortened to their base path and aggregated based on the root website. The extremes are dominated by news media websites, indicating that politicians from each wing tends to reference the same news media. Furthermore, the right-wing tends to use link-shorterners more than the left-wing.