Topic detection in microblogs using big data and neural networks

Robert Musters - s1774395

June 2017

Master's thesis
Artificial Intelligence
University of Groningen,
the Netherlands

Supervisors:

Dr. M.A. Wiering
Artificial Intelligence and
Cognitive Engineering,
University of Groningen

Prof. Dr. L.R.B. Schomaker
Artificial Intelligence and
Cognitive Engineering,
University of Groningen

university of groningen

faculty of mathematics and natural sciences

Abstract

In Artificial Intelligence and Natural Language Processing, topic detection is challenging. Topic detection is the classification, differentiation and detection of, for example text, in a single program. A common approach is using Latent Dirichlet Allocation (LDA) to model a topic as a combination of words. A close resemblance to LDA is Non-Negative Matrix factorization. Both algorithms are used in topic modeling. The latter has a close resemblance to a Neural Network named word2vec, but lacks the ability to make an informed decision. The downside of training the previously mentioned algorithms, is that they require a large amount of annotated data to work accurately. Annotating data is a labor intensive task. Kmeans is an algorithm that overcomes the lack of annotated data. The pitfalls of Kmeans are discussed and Kmeans is compared to a neural network.

This thesis researches the use of Active learning techniques to facilitate the annotation task. Active learning is a semi-supervised machine learning technique which interacts with the user to train an algorithm. Active learning can significantly reduce the amount of annotation by asking a user to label data in a smart manner. The data contain five years of textual messages collected from the social media platform Twitter. Architectures to analyse large amounts of data, the field of Big Data, are discussed and used. Topic detection using Kmeans is compared to the word2vec model with an extra neural network layer. We use Active learning to research the efficiency of using it to annotate Twitter data. An objective experiment, using real life users, is done to validate the word2vec model on Twitter data.

The proposed method could increase the accuracy on topic detection. Also, it could aid business and academia to annotate (Twitter) data more efficiently.

Keywords: Neural Network, Tensorflow, Big Data, Kmeans, Twitter, Active Learning.

Acknowledgements

This work was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative.

Contents

Chapter 1

Introduction

Humans understand text in an almost automatic manner. For intelligent algorithms it is still more challenging. Topic detection is one of these challenges and comprises of three sub-challenges, namely: classification, differentiation and detection. A lot of research is done into topic detection using Naive Bayes [1], (Online) Latent Dirichlet Allocation [2, 3, 4], Latent Semantic Analysis and Non-Negative Matrix Factorization [5]. A recent development that contributed to text analysis, is in the field of Neural Networks. A method called word2vec, closely related to Non-Negative Matrix Factorization [6], has been developed. In the recent years, a lot of text data has become available due to a social media platform called Twitter. Twitter is a social media platform where people can be more engaged with topics than ever before. A lot of research has been done on Twitter [7]. The research includes geo-tagging [8, 9] of users which choose to not share their location information and detecting upcoming events [10]. Location information can be added as a feature in e.g. Online Dirichlet Allocation [8]. More traditional machine learning algorithms as K-Nearest Neighbors and Support Vector Machines have been used to detect radicalization on Twitter [11]. Previous research covers most of topic classification, differentiation and detection. However, a qualitative analysis of classifying, differentiating and identifying a topic using Twitter data is absent. Previous research also uses the English language and heavily relies on out-dated datasets, such as Reuters[1]. These datasets lack the domain specific syntax and vocabulary used on Twitter. In this research we give a quantitative analysis of topic detection using Twitter data, as well as incorporating word2vec in topic detection. Data are acquired using Active Learning, a method which labels data in an intelligent manner using a human annotator.

In the next section, we formalize the research questions.

[1]`https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection`

1.1 Research questions

Main question:

- How to best do topic discovery on social media text?

Sub-questions:

- What is a promising method for extracting the underlying structure of a microblog? [12]
- How to cope with missing or noisy data? [13]
- Which current machine learning techniques are used to do topic discovery and what are the limitations?
- How to best find a mapping between social media language and natural language?
- How to improve the state of the art machine learning techniques for topic detection on Twitter data?

In this thesis, topic discovery using Kmeans will be compared to the word2vec model with an extra classification layer. To the best of our knowledge, we believe that this has not been done before.

1.2 Applications

Understanding text and extracting topics is useful in many applications. One such application is serving advertisements to a user based on some text which he/she placed on the internet. A company which tries to add relevant social media information to news articles is Crowdynews[2]. Crowdynews needs to know what both sources are about, which they try to accomplish using machine learning.

Another application is in Google advertisement services. Advertisements need to be interesting to a user, else the advertisements will not be clicked and Google won't earn money. The better the advertisement is suited to the interest of the user, the more it will be clicked and the Click-Through-Rate (CTR) will increase as well as the revenue [14]. A better suited advertisement with a high CTR will result in a high Cost-Per-Impression (CPI). CTR and CPI are strongly correlated. When an intelligent system tracks and understands information which a user has read or has posted on the internet, a profile of a user can be constructed. The profile could contain the topics a user is interested in. The better such a profile is constructed and the better the understanding of the intelligent system, the higher the CTR and CPI.

1.3 Outline

This thesis has the following structure: in chapter 2, we provide the necessary theoretical background. In chapter 3, we describe the experimental setup. In chapter 4, we present and discuss the results. Finally, conclusions are drawn and future work set out in chapters 5 and 6.

[2]`www.crowdynews.com`

Part I

State of the Art and Theoretical Background

Chapter 2

Theoretical Background

In this chapter we give the theoretical background of the methods that will be used and/or discussed.

2.1 Topic detection

Topic detection can be done by determining correlations between words. Highly correlated words can indicate a similarity in topic. The state of the art method for deriving vector representations of highly correlated words, which sort of have the same meaning or are related, is a neural network called word2vec [15].

Another type of neural network is the Long Short-Term Memory (LSTM) [16]. In [17], the authors use a multi-layered LSTM to learn to map input sentences of variable length to a fixed-dimensional vector representation. These vector representations can be used to do topic analysis.

State of the art topic modeling is done using LDA or online-LDA which belong to the family of Bayesian statistical models.

2.2 Latent Dirichlet Allocation and variations

Latent allocation of words [18, 19, 20] is used to find topics (using Latent Dirichlet Allocation, non-Negative Matrix Factorization). Latent allocation is the deconstruction of the matrix of co-occurrences of words into factors. These factors contain clusters of words which are weighted. Online learning is used to update the latent allocated words [21].

Online-LDA (OLDA [3, 4]) represents documents as random mixtures of underlying topics. A topic is described as a probability distribution over a number of words in a predefined dictionary of words. The process of generating a dictionary with LDA is performed in four steps. For each document D, a multinomial distribution ϕ over topics is chosen from a Dirichlet parameter α. A topic is chosen from this distribution. For a specific topic z_{di}, a word distribution ϕ_{di} is selected to produce a specific word by randomly sampling words from the topic distribution. OLDA uses dynamic changes in the data to update the learned topics. Words in the stream of documents are sampled based on the latest distribution.

Location and time information can be incorporated in the OLDA process [8]. The authors of [8] propose a graphical model called location-time constrained topic (LTT) to capture the social data information over content, time, and location. The latitude and longitude are modeled by a Beta distribution conditioned on the topic. Not only do they have a probability distribution of words over a topic, they also have a probability distribution of locations over a topic. In other words, a social message is represented as a vector of probabilities over the space of topics which are dependent on the word in a tweet, the time, and locations of tweets. The dissimilarity between two tweets is calculated by using the Kullback-Leibner divergence. The dissimilarity is transformed to a true distance function.

Hierarchical Latent Dirichlet Allocation (hLDA) [22] is another algorithm used in topic detection. The difference between LDA and hLDA, is that hLDA learns the topic parameters from the corpora.

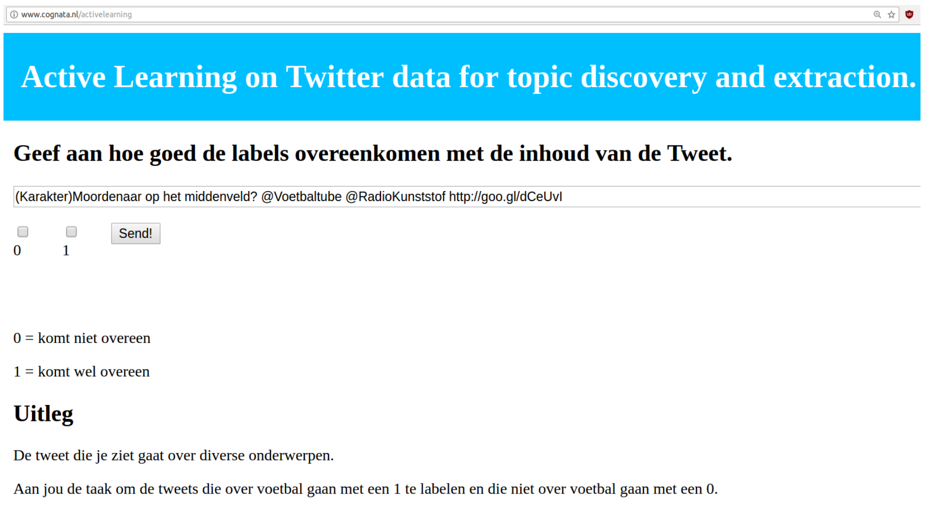
2.3 Naive Bayes

Sriram et al. used Naive Bayes to classify tweets in general topics [7]: News (N), Events (E), Opinions (O), Deals (D), and Private Messages (PM) based on the author information and features within the tweets. Eight features were extracted using greedy search. The features consist of one nominal (author) and seven binary features: presence of shortening of words and slang, time-event phrases, opinion words, emphasis on words, currency and percentage signs, “@username” at the beginning of the tweet, and “@username” within the tweet). This research does not focus on broad topics, rather it focuses on specific topics.

2.4 Data collection

Twitter data contain small text messages which are not labeled by topic. To acquire these labels, we need people to label the data on topic. A service which provides the architecture to do this is Crowdflower[1] or Mechanical Turk [2]. This platform is used by data scientists to enrich data. A template containing the enrichment requirements is sent together with the data and the labeled data are retrieved. Unfortunately, this service has almost no Dutch users and therefore we created our own architecture as can be seen in figure 2.1.

[1]`https://www.crowdflower.com`
[2]https://www.mturk.com


www.cognata.nl/activelearning

Active Learning on Twitter data for topic discovery and extraction.

Geef aan hoe goed de labels overeenkomen met de inhoud van de Tweet.

(Karakter)Moordenaar op het middenveld? @Voetbaltube @RadioKunststof http://goo.gl/dCeUvl

0 = komt niet overeen

1 = komt wel overeen

Uitleg

De tweet die je ziet gaat over diverse onderwerpen.

Aan jou de taak om de tweets die over voetbal gaan met een 1 te labelen en die niet over voetbal gaan met een 0.

Figure 2.1: The annotation system where tweets can be labeled according to a topic.

2.5 Clustering

In this section we explain the different variants of clustering methods and their up- and downsides. We start with explaining two distance measures, as it is of great influence in how clusters are formed.

2.5.1 Distance measure

To compare an instance to another, some measure of similarity is needed. K-Nearest Neighbors (KNN) is a method often used as a baseline. KNN is a method which compares an instance to all other instances for which the Euclidean distance (EUD) measure is often used. KNN finds the K closest instances to instance A, collects the labels of these instances and classifies instance A as the class with the most frequent label from the retrieved instances. The distance is used as a means of quantifying the similarity between instances. The smaller the distance, the more similar the two instances. A downside to this method is, that all the data need to be stored. Another downside is that each time an instance needs to be classified, the distance to all the other instances has to be calculated. KNN is a memory and computationally expensive method. KNN is used, and explained well in [11] to find radicalization on Twitter, using the Term-Frequency approach.

Several distance measures have been proposed in the literature. Some have very similar behaviors in queries, while others may result in different answers. In [23] the common distance measure: Euclidean distance (EUD) and the distance measure related to word2vec: Cosine Similarity (CS) are compared. The CS distance is defined as:

$$d(v_i, v_j) = \frac{v_i \cdot v_j}{||v_i|| \cdot ||v_j||}, \tag{2.1}$$

where v is a feature vector and i is the feature instance compared to all other feature instances j.

The CS is a measure to compare two vectors to each other. A value of one means they are highly similar, a value of minus one means they are highly dissimilar and a value of zero means that the vectors are not similar nor dissimilar. The CS measures the similarity using the vector its direction and not its magnitude. The CS can be used to compare vector representations of words to each other. Not using the magnitude can be seen as a method to normalize sentence length. In other words, the amount of words in a sentence does not influence the similarity.

Qian et al. concluded that in high-dimensional data spaces, the query results for EUD and CS are similar [23]. The results were based on normalized as well as non-normalized data, uniformly distributed data, clustered data, and real image data. For clustered data the results are even more similar. Qian et al. made the assumption that data points are uniformly distributed within the space and there is no dependence between dimensions, which in our Twitter dataset is probably not the case. A positive effect of using the CS is that it abstracts out the magnitude of the term vector, because it takes out the influence of document length. Only the relative frequencies between words in the document, and across documents, are important, and not how large the document is which is especially important in micro-blogs such as tweets.

Qian et al. conclude that when both distance measures are normalized, the query results will be exactly the same. For this reason, we use the Euclidean distance throughout this research.

2.5.2 Kmeans

The Kmeans algorithm is a popular clustering method used in a wide variety of research fields. Kmeans vectorizes the data into Voronoi tesselations or in other words: clusters. A data point belongs to a cluster according to the hard assignment scheme. The hard assignment formula is shown in equation 2.2. It determines to which cluster i, the data point j, is a member of.

$$b_i^t = \begin{cases} 1 & \text{if } \|x_t - c_i\| \; = \; min_j \, \|x_t - c_j\| \\ \\ 0 & \text{otherwise} \end{cases} , \tag{2.2}$$

where x_t is the t-th data point and c_i is the i-th centroid.
Given the membership values b, the total reconstruction error is defined as:

$$E(\{c_i\}_{i=1}^{k}) = \sum_{t=1}^{n} \sum_{i=1}^{k} [b_i^t \, \|x_t - c_i\|]^2 \tag{2.3}$$

,
where n is the amount of data points and k the amount of clusters.
The error function is used to determine the homogeneity of the clusters, if the total sum of squared distances between each point and their closest cluster goes down, the homogeneity goes up.

The centroids are then computed as:

$$c_i = \frac{\sum_{t=1}^{n} b_i^t x_t}{\sum_{t=1}^{n} b_i^t} \tag{2.4}$$

which can be explained as each centroid c_i is the mean of the data belonging to the cluster i. The above process is repeated until the memberships of data points do not change anymore. Kmeans has several desirable, and undesirable, properties depending on its usecase:

1. The cluster amount k has to be predefined.
2. It always finds k clusters, even if the data are random.
3. Initial cluster centers, centroids, are not intuitively chosen.
4. Data should be normally distributed.
5. Data should have equal variance.
6. Not dependent on data order.
7. Susceptible to local optima.

Figure 2.2 explains this in more detail.

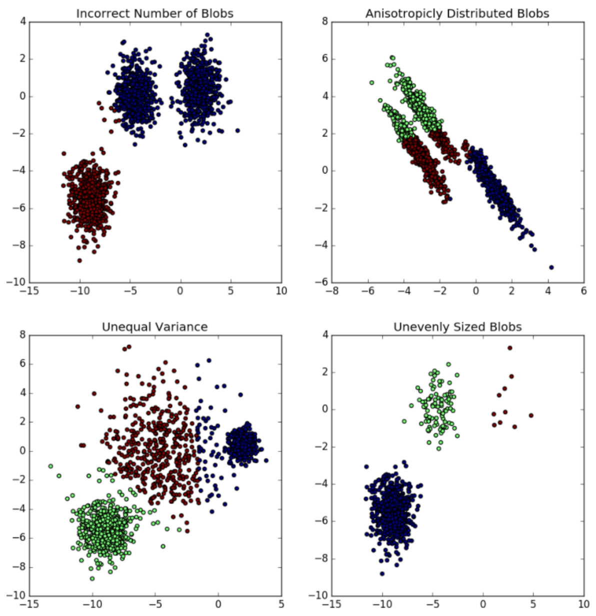


Figure 2.2: In the top left corner, the effect of an incorrect amount of clusters K is shown. K is chosen as 2, while there are actually 3 clusters. In the top right corner, the effect of not normally distributed data is shown. The true clusters are not represented well by the Kmeans clustering. In the bottom left corner the effect of unequal variance is shown. Visually inspecting this image clearly shows the clustering to be wrong. The bottom right corner, shows the effect of unevenly sized blobs, which Kmeans can cluster correctly. Source: `http://scikit-learn.org/stable/_images/sphx_glr_plot_kmeans_assumptions_001.png`

Empty clusters can be formed when a centroid is not initialised near an input vector from the data, or other centroids attract data points more. A clustering method solving this problem is bi-secting Kmeans.

2.5.3 Bi-secting Kmeans

Bi-secting Kmeans [24] is a clustering method which is a variant of Kmeans where clusters are formed in a top down approach. The data are iteratively split, where each split should increase the cluster homogeneity. Bisecting Kmeans also builds a hierarchy in the data. Each branch in the hierarchy represents a sub-cluster. These sub-clusters can be used as initialization for another clustering method.The performance of Kmeans, as well of its variant Bisecting Kmeans, decreases when data are not normally distributed or when clusters do not have equal variances. The separation between clusters could be compromised, because it is not reflecting the true separation. When data are more naturally distributed, meaning not always normally distributed or not having equal variances, the clustering method Optics is preferred.

2.5.4 Optics

Optics is a clustering method which uses the density of the data to cluster [25]. We wanted to mention a density based approach for completeness and because we used in it testing, but we will not elaborate further on this method due to the conclusion drawn below. Optics clusters data by taking a radius around an initial data point and calculating how many data points are within a preset range. If there are more than a minimal amount of data points within a range, the data points are added to a cluster, else a new cluster is formed. The downside of Optics is that it does not automatically return clustered data. However it does visualize probable clusters in high dimensions in a 2D graph which Ankerst et al. named the reachability plot (see figure 2.3 [25]).

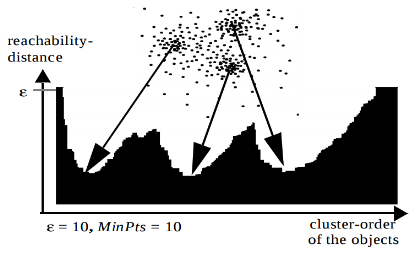


Figure 2.3: Intuitive image showing how Optics visualizes clusters in a 2D plane. The steep abyss followed by valleys, indicate clusters. Image taken from [25]

We conclude that Optics is not a good method for clustering data, because it uses a threshold in the reachability plot to determine clusters boundaries. In large datasets with many clusters, empirically determining cluster boundaries is too labor intensive and subjective. It does however give a good insight into the structure of the data.

2.6 Neural networks

Rong explains the concept of a neural network well [26]. Not only an insight into neural networks is given in general, but also into the neural network word2vec in specific.

2.6.1 Artificial neuron

An artificial neuron is a biologically inspired computational unit. The unit receives activation signals from other units, sums the signals and makes a decision according to a decision function as shown in figure 2.4.

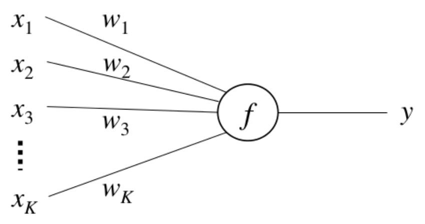


Figure 2.4: The artificial neuron.

The unit (figure 2.4) works in the following way: $x_1, ..., x_K$ are input values, $w_1, ...w_K$ are weights, f is the activation function and y is the output. K is the amount of units.
The output is calculated according to:

$$y = f(u) \tag{2.5}$$

where u is a scalar number, which is the total activation from all the other units to which the current unit is connected. The total input, u, is calculated by multiplying each activation signal from a connected unit with the strength of this connection, called a weight w. Equation 2.6 expresses this mathematically.

$$u = \sum_{i=1}^{K} w_i x_i \tag{2.6}$$

Using vector notation, we can write:

$$u = w^T x \tag{2.7}$$

A bias term can be added to u to shift the activation function to the left or the right, aiding learning.
An activation function is the function which gives the output value given the summed inputs. The Perceptron uses a unit step function, defined as:

$$f(u) = \begin{cases} 1, & \text{if } u > 0 \\ 0, & \text{otherwise} \end{cases}$$

A different activation function can be chosen to change the dynamics of the neural network and increase the Vapnik-Chernovenkis (VC) dimension. The VC dimension is a measure for the capacity of a classifier.

Another activation function is the linear activation function which maps input linearly to the output.

$$f(x) = x \tag{2.8}$$

Most multi-layer perceptrons use the logistic activation function (also called the sigmoid function):

$$f(u) = \frac{1}{1 + e^{-u}} \tag{2.9}$$

The logistic activation function has four properties:

1. The output is between 0 and 1.
2. The logistic activation function is smooth.
3. The logistic activation function is easily differentiable in the update equation.
4. The function introduces non-linearity into the model.

To train perceptrons, an update equation is needed:

$$w_{t+1} = w_t + \eta(y_t - y)x_t, \tag{2.10}$$

where η is the learning rate ($\eta > 0$), y is the output, y_t the target output and x_t the input.
To update the weights correctly, by comparing the output and the target output, an error function needs to be defined:

$$E = \frac{1}{2}(y_t - y)^2 \tag{2.11}$$

The training of the neural network stops when the error is below a threshold. Minimizing the error function is the objective of a neural network. Combining the update equation (equation 2.12) with the derivative of the error function (equation 2.11), results in stochastic gradient descent:

$$w_{t+1} = w_t + \eta(y_t - y)y(1 - y)x \tag{2.12}$$

A Rectified Linear Unit (ReLU) [27] can be used to replace the sigmoidal activation function. The ReLU is defined as:

$$f(x) = \max(0, x) \tag{2.13}$$

ReLUs are faster to compute than sigmoidal activation functions and introduce non-linearity into the model as well.
Layering artificial neurons is called a multi-layer neural network. Word2vec is such a network. In the next chapter we go into further detail about word2vec and multi-layer perceptrons.

2.6.2 Word2vec

Word2vec[3], developed by Mikolov et al., is an algorithm belonging to the class of neural networks [15]. The algorithm provides state-of-the-art word embeddings. Word embeddings are vector representations of words, where words with similar meanings are close to another in vector space. Words which are in similar contexts have similar meanings according to the distributional hypothesis [29].

Two models can be chosen using word2vec: continuous bag-of-words (CBOW) or continuous skip-gram. The continuous bag-of-words model predicts the current word from a window of surrounding context words. The continuous skip-gram model uses the current word to predict the surrounding window of context words.

Generally, a word vector is presented at the input and a target word vector is presented at the output. The vector consists out of zeros or ones. These values represent the skip-grams as described in section 2.6.4. Depending on the model chosen, these vectors consist out of one or more none zero values. In the CBOW model the input consists of more than one non zero element and the output is a vector with a single non-zero value. In the skip-gram model it is vice versa. A vector with only a single non-zero and positive integer value with all the other values being zero is called a one-hot encoded vector.

During training, the input and output pairs are used to set the weights of the network correctly. The weights are representations of the connections between the input to the hidden layer and the hidden layer to the output layer. The activation function of the word2vec model is the linear activation function (equation refeq:linear). The weights are updated using backpropagation.

Backpropagation is a method which updates weights according to a difference between the output and the target output, commonly known as the error. During testing, a word vector is presented at the input and the neural network will be most active in some hidden units, resulting in an output vector where some vector elements are larger than the other. Using the softmax function, the output is converted to real values in the range of $[0, 1]$ which sum up to one. The output with the largest softmax value corresponds to the predicted target word (or context words). The vectors of the activations of the hidden layer can be used with the cosine similarity measure to find words with approximately the same meaning or for clustering words into classes.

As mentioned before, word2vec can be used to predict a word given its context or predict the context given a word depending on the model used. The algorithm used in the first case is discussed in section 2.6.3. The latter, named Skip-gram, is discussed in section 2.6.4. In this thesis we use the skip-gram model, albeit being slower than CBOW, it does perform better for infrequent words according to the author's note[4].

2.6.3 Continuous bag of words

The Continuous Bag Of Words (CBOW)[26] is used to calculate word embeddings which place similarly meaning words close in vector space.

The CBOW model (see figure 2.5) is an efficient method for learning distributed vector representations that reflect a large number of precise syntactic

[3]`https://code.google.com/p/word2vec/` retrieved January 2016
[4]`https://code.google.com/archive/p/word2vec/`

and semantic word relationships [30]. The downside of this model is that the order of context words, which could be important, does not influence prediction.

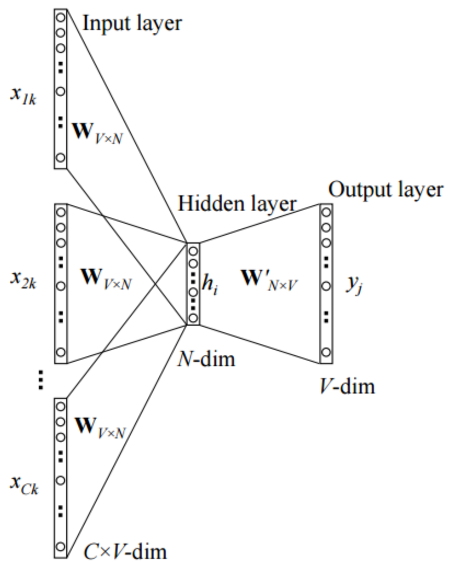


Figure 2.5: The continuous bag of words model takes words as input represented as a one hot encoded vector. x are the context words surrounding the target word. V is the vocabulary. C is the size of the context. N is the number of hidden units. y is vector representing a word which is a one hot encoded vector during training.

At the input layer, word vectors representing sentences are presented. The hidden layer then learns the word-embeddings by ways of adapting weights W and weights W'. Adapting these weights is done using backpropagation. Backpropagation determines the error between the prediction and the target and updates the weights to reduce the error. After training the model, word vectors can be obtained by presenting context words as the input. The vector of the activations of the hidden layer hold some topical meaning about the word. Several activation vectors, obtained from inputting different words from a tweet, can be averaged to be used in classifying if a tweet belongs to a topic.

2.6.4 Skip-gram model

Goldberg and Levy explain the skip-gram model (figure 2.6) well [15]. The model is trained using a corpus of words w and their contexts c. The skip gram model uses words and their context pairs to train the neural network. To limit the pairs, pairs that appear frequently together in some context, but not in others, are replaced by a unique identifier.

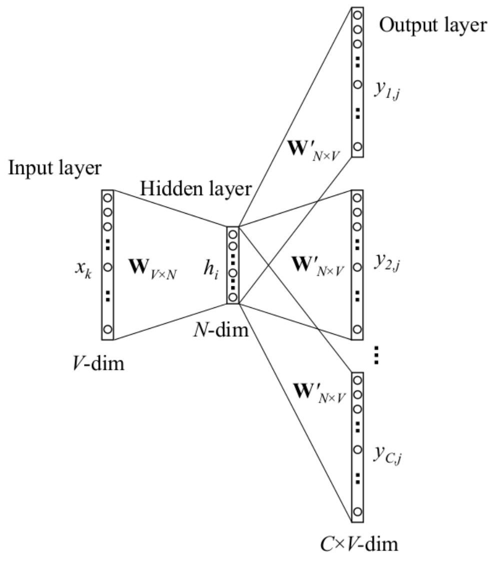


Figure 2.6: The Skip-gram model creates word embeddings through predictions of the context which are based on another word in the sentence. Therefore the skip-gram model is the reverse of the CBOW model.

The decision to replace infrequent words by a unique identifier, is made empirically.
Given a sequence of training words, $w_1, w_2, ..., w_T$, the skip-gram model aims to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} log(P(w_{t+j}|w_t)) \tag{2.14}$$

where c is the size of the training context and T the amount of words in the sentence. The parameter c in this research is small given the character limitation of tweets.
The Bayesian component: $P(w_{t+j}|w_t)$ in equation 2.14 is defined as:

$$P(w_i|w_j) = \frac{e^{u_{w_i}^T v_{w_j}}}{\sum_{l=1}^{V} e^{u_l^T v_{w_j}}} \tag{2.15}$$

where u_w and v_w are the vector representations of a word and its context respectively. V is the number of words in the vocabulary. Equation 2.15 is known as the softmax function. The probability of predicting word w_i given word w_j is determined by the softmax function. As can be seen from the formulas above, calculating $log(p(w_i|W_j))$ is proportional to V, making it computationally expensive. A method to reduce V is needed.

2.6.5 Hierarchical softmax

In [30], the hierarchical softmax algorithm is described. The hierarchical softmax is a more efficient algorithm than the regular softmax function. The first step in the hierarchical softmax algorithm is to build a Huffman tree based on word frequencies. A Huffman tree is a binary tree meaning each node has two branches and each word is a leaf of the tree. For each word, represented by a node, a decision is made to go left or right. The probability for the word in the leaf is the multiplication of the probabilities along the path.

Each sigmoidal output unit is connected to a node in the Huffman tree. Since we have established a Huffman tree, we only have to calculate the sigmoidal activations of the nodes in the Huffman tree which are on the path to the word. Using a Huffman tree, the computational complexity of calculating V sigmoidal outputs reduces to $log(V)$ (under the assumption that the tree is balanced).
To counter the imbalance between rare and frequent words, a sub-sampling method is used.

2.6.6 Sub-sampling

To correct for high frequency words, and limiting the size of the vocabulary, the following subsampling method is used:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{2.16}$$

where $P(w_i)$ is the probability that a word gets discarded, t is a threshold ($t = 10^{-5}$) and $f(w_i)$ is the word its corpus frequency. For example, the word "the" is present in the corpus a thousand times. There are a ten million words in the corpus. The probability of removing the word is: $1 - \sqrt{\frac{10^{-5}}{10^3/10^7}} = 0.68$ Equation 2.16, comes down to a low-pass filter, filtering high frequency words. In this thesis, the high frequency words are removed using stopword removal according to section 3.1.1. Furthermore, we do not agree with the terminology, because their sub-sampling method comes down to a weighted adjustment of the frequency of words. The word2vec implementation used in this thesis has a *min_count* parameter. This parameter is used to filter words which are less frequent than *min_count*. The parameter is set to 0, because stopword filtering is done before the word2vec model is built.

Word2vec using the previous mentioned parameter settings, does approximately the same as Non-Negative Matrix Factorization (NMF) [6, 21, 5], but has been shown to perform better [6].

2.7 (Inter)active learning

The state of the art active learning scenarios [31, 32, 33] are:

1. Membership query synthesis: randomly select examples,
2. Stream-based selective sampling: decision based on information criteria,
3. Pool-based active learning: sampling a pool of examples from a large dataset based on information criteria.

Pool-based selective sampling will be used in this research because a pool of Twitter data is used.

Selecting which examples to present to an oracle (person who labels the examples for the active learning algorithm) is done using uncertainty measures as least confident, margin or entropy. In this proposal we use the confidence to limit the research scope.

In this project, we wanted to use the entropy uncertainty measure. However, the data was too noisy to use entropy. Therefore we used the confidence uncertainty measure. The confidence uncertainty is reflected by the softmax output of the neural network. A low value means the network has more confidence that the data point does not belong to a certain class and a high value means it is certain. Unfortunately, the performance of the neural network was also low. The disagreement measure will be reflected by the test-subjects which label the data using the Cohen's kappa coefficient [34]. The Cohen's Kappa coefficient measures the inter-rater agreement between test subjects. The disagreement measure will not be used to update the model.

Unfortunately, due to the poor performance of the proposed models, active learning could not be achieved. Therefore the definition of active learning is relaxed to interactive learning. Interactive learning is the use of the input of test subjects to guide the training of a model.

2.8 Scalable data processing

For calculating the models we use two scalable architectures, namely: the Peregrine cluster[5] and the SARA Hadoop cluster[6]. The first is optimized for running programs in parallel and the latter is optimized to run programs distributed. The SARA Hadoop cluster consists of 170 data/compute nodes. The nodes have a total of 1370 CPU-cores for parallel processing and has a distributed file system with a capacity of 2.3 PB. The advantage of running programs in a distributed manner is that the program is not limited to a single node containing a certain amount of CPUs. The distributed program can approximately use all the cores in the cluster.

The software used here, which is suitable to train the model on a distributed system, is called Apache Spark[7]. The library used is called MLlib and more specifically we use the word2vec model[8]. The Peregrine cluster was not used in the end, due to unreliable service as well as not having enough RAM memory.

[5]http://www.rug.nl/society-business/centre-for-information-technology/research/services/hpc/facilities/peregrine-hpc-cluster

[6]https://userinfo.surfsara.nl/systems/hadoop/description

[7]https://spark.apache.org/docs/latest/index.html

[8]https://spark.apache.org/docs/latest/mllib-feature-extraction.html#word2vec

Part II

Experiments and Results

Chapter 3

Experiments

In this chapter we explain the experiments done in this research. We will first shortly give an overview. We first gather labels from test subjects. These labels refer to two topics: soccer and jihadism. The second topic is not used further in this research and is chosen at random. The second topic is important because it is a whole different topic than soccer and thus suffices as a negative class. We use the labels in two methods: Neural networks and Kmeans. First a word2vec model is trained on the unlabeled data. An extra layer is added to the word2vec model for classification purposes. This neural network is quantitatively compared to Kmeans.

We will now give a more detailed explanation of our experiments.

3.1 Pre-processing

To reduce the dimensionality of the data, we remove some words which are of little meaning.

3.1.1 Stop word removal and stemming

All the dutch stop words are removed and replaced by the "<STOPWORD>" token and URLs are replaced by the "<URL>" token. To further reduce the dimensionality, words are stemmed [35] using a well known Dutch stemmer[1].

Traditionally in text pre-processing the steps described above are enough. In Twitter messages, other terms have to be replaced as well. These terms include mentions (terms characterized by the @ sign) and hashtags (terms characterized by the # sign). Mentions are names of Twitter users which the user wants to include in the conversation [12].

We use the methods described above to pre-process the data. The pre-processing might not be optimal, but it sets out a baseline to which other researchers can compare with.

[1]`http://www.nltk.org/`

3.1.2 Features

Several features can be used to represent text. The most well-known is the Term Frequency, which as its name suggests, counts the frequency of the words in a text. A more robust method is the term frequency–inverse document frequency, which normalizes the term frequency by the amount of occurrences in a document. We use the output of the word2vec model to represent the terms [28]. Each term of a tweet is presented to the word2vec model and the output for each term is summed and averaged [30].

The downside of averaging terms is that it loses the word order in the same way as the standard bag-of-words [7] models do. Another approach, using a parse tree to combine word vectors, has been shown to work for only sentences because it relies on parsing. The latter option is not suitable for tweets, because syntactical structure is often missing [36].

3.2 Dataset

The data consist of 352 million tweets saved on the Hadoop Distributed File System (HDFS). HDFS allows us to distribute processing on chunks of data. On a normal desktop system, programs such as pre-processing (section 3.1), are run in series and data can not be shared amongst processes. HDFS allows to share chunks of data amongst multiple processes running the same program, rendering an enormous performance boost.

From all the data, we take a sample which consists of tweets from the first month of January 2015. The January sample contains $25,630,785$ data points. The word2vec model is trained using the January sample. From the January sample we take $13,628$ tweets about soccer and $1,532$ tweets about jihadism. The tweets are selected by checking if the word ”voetbal” or ”jihad” is in the tweet. Soccer and jihadism are the two classes we use in the experiments.

3.2.1 Missing data

Missing data are a challenge for algorithms such as Kmeans. Calculating the distance of a data-point with some feature elements missing is difficult. The Euclidean distance measure, which is used in the Kmeans algorithm, can not handle missing data.

Missing data occur in the word2vec model. The word2vec model is constructed using a limited vocabulary. The words in this vocabulary are used to construct higher level representations using e.g. the skip-gram model. In section 3.1.2, we explained how the vectors of each token can be combined into a single vector for each tweet. When a word is not in the vocabulary, the weight vector can not be extracted and the word is disregarded. When no words in a tweet are in the vocabulary, the whole tweet will be disregarded.

The skip-gram model of the word2vec model uses a window which convolves over the tokens in the tweet. We use a window size of five. When the window size is too large, some tweets will be disregarded due to the fact that it does not have enough words. When the window size is too small, context could not be learned effectively. When a tweet does not have enough words, it will be disregarded.

3.2.2 Kmeans

Kmeans is a distance-based clustering algorithm. We use Kmeans to find clusters of topics by using the weight vectors gathered from the word2vec model. All the $25,630,785$ tweets from the January sample mentioned in section 3.2 are used. The Kmeans model requires a preset parameter k which sets the amount of clusters which we think are in the data.

The gap statistic [37] is often used to find the right amount of clusters with the highest accuracy on the train and testset. However, we do not use the gap statistic here, because small clusters could be missed. These small clusters could contain specific topics which are not abundantly present in the dataset.

3.3 (Inter)active learning

Active learning is a method used to reduce the amount of expert labelling. Examples which maximize the certainty of membership to a class, or being a new class by itself, are presented. Active learning is a semi-supervised method which interacts with a user in an efficient manner. Active learning will be done using Neural networks and Kmeans [33, 38, 39, 40]. Unfortunately, preliminary results showed a poor performance of the models. The classes were not well separable. Optimizing class boundaries using e.g. entropy between datapoints was pointless. Therefore we relaxed the properties to interactive learning. In the neural networks case, tweets which were semantically similar were harvested and evaluated. In the Kmeans case, annotated tweets from the test subjects were used to identify the cluster corresponding to the class label. The Kmeans clusters are evaluated. The cluster corresponding to the class label is evaluated in depth. In both cases, the manual labeling of the test subjects could result in a multitude of new data which have the same label.

An active learning system will be used which asks the user to classify example text about which it is uncertain. In other words, examples about which the algorithm is uncertain will be presented to the user to label. This labeling will limit the version space. Presenting unlabeled data will be done as efficiently as possible. In other words: the system will manage resources efficiently with an as close to optimal performance. The resources consist of money which has to be spent on manual labor and computing time. Performance means being a both fast and accurate classification algorithm.

The final system will be a website (see figure 2.1) made using Flask[2] which asks the user for input using active learning to find the mapping between topics and tweets.

3.4 Binary search

Participants were asked to label tweets in such a manner that the least amount of annotations were needed to segment the data. We established this by using the probabilities of the output layer of the neural network. The first node of the output layer represents the soccer topic. When probabilities are close to one,

[2]`http://flask.pocoo.org/`

the tweet is determined to be about soccer. A threshold is needed to make this distinction. Binary search [41] is a method which iteratively adapts its value to search for an optimal threshold. The pseudocode for this algorithm can be found in Appendix B.

3.5 Evaluation measure

The performance of the system for detecting correct topics will be evaluated using the accuracy as well as the F1-score. The F1-score for the system detecting a topic with active learning, will be compared to the system without active learning.

$$F1 = 2 * \frac{(precision * recall)}{(precision + recall)} \tag{3.1}$$

The F1 score consists of two parts: precision and recall as can be seen in figure 3.1.

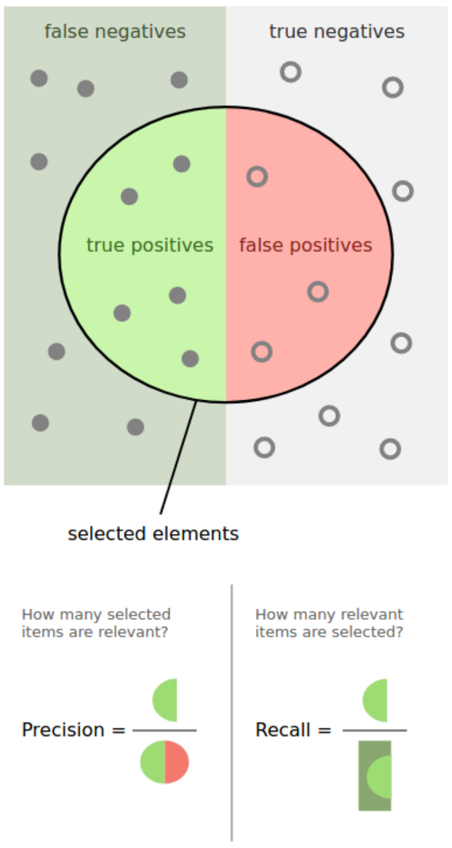


Figure 3.1: Image explaining the concepts of precision and recall visually. Source: `https://en.wikipedia.org/wiki/Precision_and_recall`. Retrieved: May 2017.

The downside of this evaluation measure is that it does not include the true negatives in its computation. Not including the true negatives introduces a bias which becomes larger when the data do not contain positive samples. When a classification system is designed to always give output, the amount of true negatives decreases, which is a problem. The upside of this evaluation measure is that it takes into account the false positives. The accuracy evaluation measure does not take into account the false positives, it includes it as being false, which introduces a bias. If a system needs to be precise, the accuracy can not measure the preciseness, the F1 score can. In other words, accuracy is not soccer specific. Accuracy also takes into account the true negatives which have nothing to do with soccer. The influence of the difference between these two evaluation measures is researched.

We now set out the experiments and continue with the results of the experiments.

Chapter 4

Results

In this section we discuss the results of the experiments described in the previous chapter. First, we explain the results of the word2vec neural network in section 4.1. The word2vec model was trained on Twitter data and similar words were placed in similar areas of the model space. For the main subject, namely soccer, we show which words are most similar.

Secondly, we discuss the results for initializing the second layer of the neural network for segmentation and classification purposes in section 4.2. A second layer is added to the neural network model together with two output nodes, for segmentation and classification purposes. The main subject, namely soccer, is attributed to the first output node and an opposite subject, jihad, is attributed to the second output node. The second layer is initially trained using seed datasets. These seed datasets consist out of tweets which contain the subject as a word in their string. After the initialization using seeds, 10 participants were asked to label the top 20 examples with the highest softmax output for soccer which did not contain the subject in their string.

We test if the participants are in agreement with each other in subsection 4.3. In this manner, we can qualitatively determine how well the model placed similar tweets together.

During exploratory data analysis, we found that the amount of words in a tweet could probably affect the quality of the model. Data sets were formed which split the original data into data with a certain amount of tokens (see section 4.5). These datasets were also labeled by test subjects. Summarizing, not only the top 20 tweets which were determined by the model to be most likely about soccer were labeled, data split by the amount of tokens were labeled individually as well.

Thirdly, we perform binary search on the output of the neural network to determine if the probabilities reflect the soccer subject. Binary search uses the confidence of the neural network to harvest positive labels, which is a form of active learning that will be explained in more detail in section 4.4.

Finally, we show that Kmeans does not work well in segmenting the soccer topic from the rest of the data in section 4.6.

4.1 Word2vec

In training a word2vec model, several parameters have to be considered. These parameters are: window size, rare word pruning and the learning rate. Which parameters are optimal is difficult to assess. As the authors of [15] explain:

"The distributional hypothesis states that words in similar contexts have similar meanings. The objective above clearly tries to increase the quantity $v_w \cdot v_c$. for good word-context pairs, and decrease it for bad ones. Intuitively, this means that words that share many contexts will be similar to each other (note also that contexts sharing many words will also be similar to each other). This is, however, very hand-wavy. Can we make this intuition more precise? We'd really like to see something more formal."

However, we can make some well thought out assumptions. The first assumption we made is to set the window size to 5. A value of 5 for the window size is large enough to use the surrounding words as context, but small enough to not use too much of the context. As [15] explains, it is difficult to assess the quality of a parameter setting, but it does work.
A linear decay function is used. The learning rate is reduced from 0.025 to 0 in n steps, where n is depending on the total amount of words divided by $100,000$. We chose the vector size to be 70. The reasoning behind this setting is that it compresses data well and leaves enough room to hold discriminative information. The word2vec model was trained on a computing cluster running Hadoop [42]. In section 3.1.1 we mentioned replacing mentions by <MENTION> tokens to reduce dimensionality. We found that if we take these tokens into consideration, we are left with attractor states which reduce the sensitivity of the model. We chose to remove these tokens as they hold little information.
The model is tested by determining if words which should have the same semantic meaning are also similar as defined by the model. The similarity is determined by the cosine similarity and the weights of each word in the model. We presented the model with the word: "voetbal" and collected the 20 most similar words based on their cosine similarity. The top 5 words are presented below and the complete list can be found in appendix A.

1. voetbal
2. voetbalnieuws
3. followback
4. international
5. voetbalnt

The first item should be the same as the item which we wanted to compare with since the similarity of something with itself is one. The second item: "voetbalnieuws" translates to: "news about soccer", which is indeed about soccer. The third item is not about soccer, but is a word used by Twitter users to get a follow from another user. A follow is where a user subscribes to another user to get updates about what he puts on Twitter. The fourth item is commonly used together with "voetbal" or "international"competitions. "vi" and "vinl"

refer to the Dutch soccer tv program. The other items are left to the reader to check. We have established that the model is working properly.
We now have established a subjective measure of how well the model functions. Now we want to determine an objective measure of how well the model not only functions on words, but also on Twitter messages as a whole.

4.2 Neural network and seeds

A neural network is trained on two classes; a class containing the string "voetbal" and a class containing the string "jihad". Other datasets are formed which split the previous mentioned dataset on amount of tokens, e.g. tweets about "voetbal" which contain four tokens are collected in a dataset containing four tokens. These datasets we call the "ntokens" datasets. We found that tweets contain between one and eighteen tokens (after filtering). These groups are used as a seed to initialize the neural network and trying to form an initial concept about the subject. 80% of the data is used to train the network and 20% to test the network. The network converges to a state with low loss as can be seen in figure 4.1.

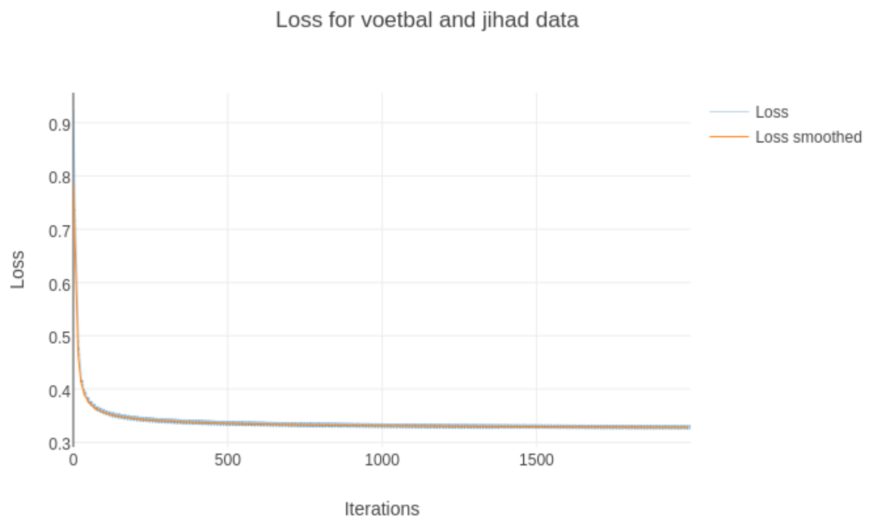


Figure 4.1: The neural network converges as can be seen by the diminishing loss which seems to stabilize.

After training, tweets not containing the string "voetbal" are fed into the neural network. Test subjects were asked to label the top 20 tweets with the highest probabilities. The tweets are labeled as being about soccer (1) or not about soccer (0). The test subjects were selected by asking if they were experts on the subject. From figure 4.2, it can be concluded that the amount of tokens indeed is of influence on the accuracy.

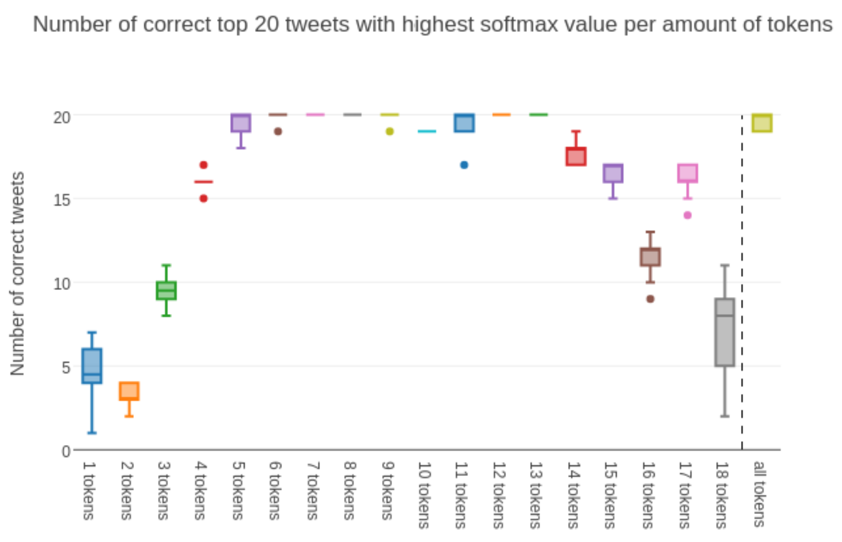


Figure 4.2: Box plots of the ntokens datasets.

We can conclude that the seed method is a good method to initialize a neural network when there are more than four tokens and less than fourteen tokens.

4.3 Inter-rater agreement

To validate if the test subjects were in agreement with each other, we calculated the Cohen's Kappa coefficient [34] for each pair of subjects (see figure 4.3). The Cohen's Kappa coefficient is a qualitative measure to determine the inter-rater agreement.

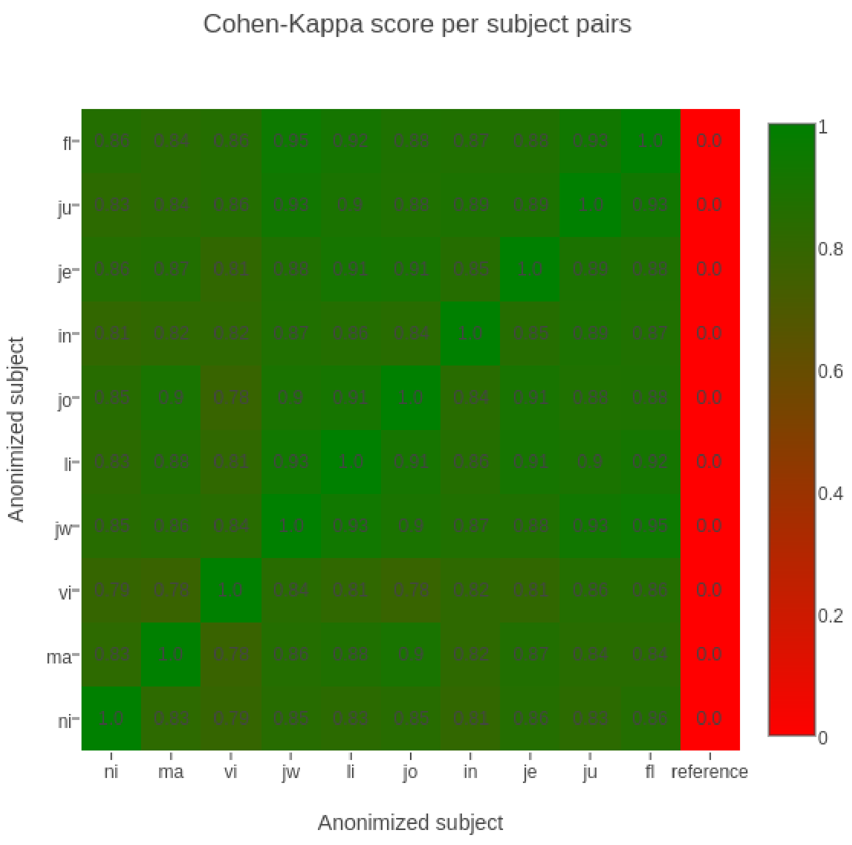


Figure 4.3: The Cohen's Kappa coefficient for each pair of test subjects. The test subjects are anonymized.

The authors of [34] state that a Cohen's Kappa coefficient between $[0.89, ..., 0.99]$ indicates almost perfect agreement amongst the raters. Figure 4.3 shows almost all the pairs of subjects have a coefficient laying in the domain of being in almost perfect agreement.

We conclude that the test subjects are in agreement with each other. The combination of the affirmative answer to the question: "Are you a soccer expert?" and the high inter-rate agreement, gives us enough evidence to conclude that the test subjects are all experts on the subject.

4.4 Binary search, (inter)active learning and neural networks

Unseen tweets not containing the string "voetbal" are fed into two neural networks. We expect that the word2vec model places tweets which have about the same meaning close together. Using the premise that tweets containing the

string ”voetbal” are about soccer, we can test if tweets which do not contain the string ”voetbal”, but are about soccer, are close together. The two neural networks are the all tokens neural network and the highest performing ntokens neural network. The all tokens neural network contains all the tweets not separated by amount of tokens in a tweet. The highest performing ntoken neural network can be determined by looking at figure 4.2. Unfortunately, due to being too inconvenient to ask the test subjects to come back after all results are gathered, we have chosen to select ntoken equals ten to be the most optimal performing neural network. The highest performing ntokens neural network is the network trained with data that contains tweets with exactly ten tokens.
Using a Binary Search algorithm a threshold is determined to expand the soccer dataset. Using the Binary Search method a threshold is determined by presenting each tweet to the test subject.
Each test subject is asked to label hundred randomly sampled tweets which have a probability above the threshold. In figure 4.4 and 4.5, the results for the 10 tokens and all tokens neural network are shown.

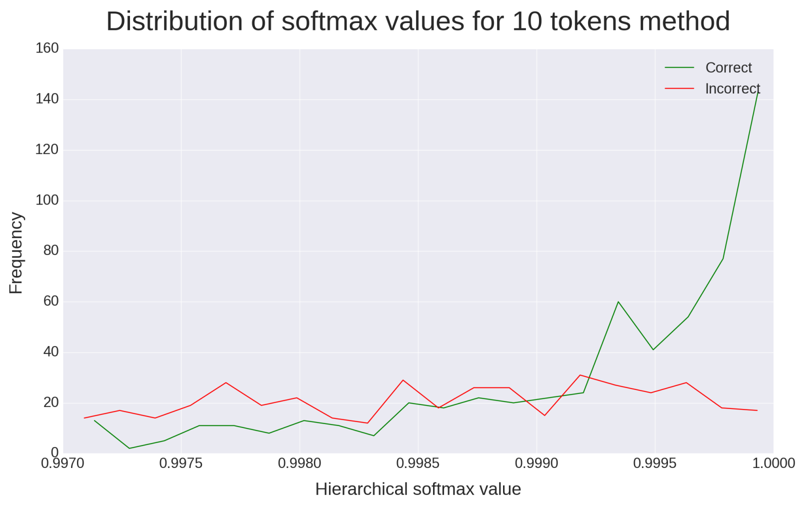


Figure 4.4: Thresholds are determined using the Binsearch algorithm on the 10 tokens neural network. Test subjects validated the results. N correct is 582, N incorrect is 418, binsize is 20.

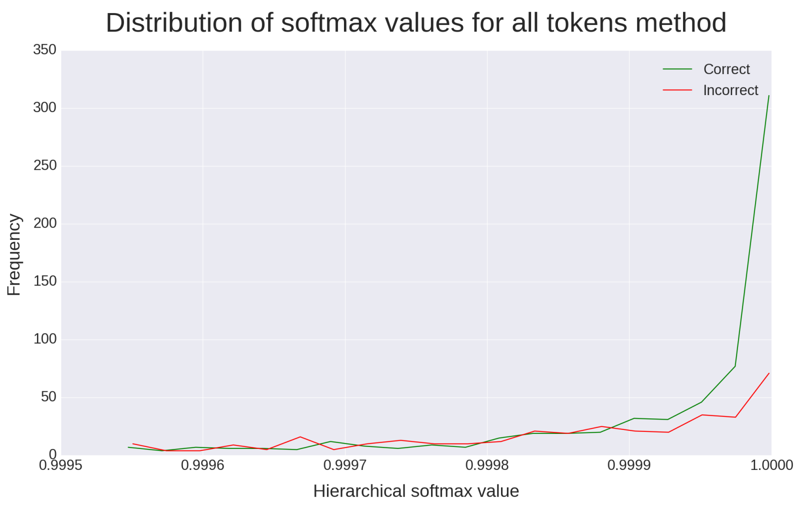


Figure 4.5: Thresholds are determined using the Binsearch algorithm on the all tokens neural network. Test subjects validated the results. N correct is 647, N incorrect is 353, binsize is 20.

From this figure, it can be seen that when increasing the threshold, the amount of correctly classified versus incorrectly classified tweets increases. Unfortunately, a threshold which does not result in false positives is not possible. The thresholds are determined using the Binsearch algorithm. Each test subject labeled tweets which resulted in an individual threshold as determined by the Binsearch algorithm. The softmax values are also high for the incorrect tweets which means the accuracy will not be high.

4.5 Influence of the amount of tokens

Test subjects where asked to label 100 tweets which were randomly sampled above the threshold of the Binary Search method.

4.5.1 Accuracy

The accuracy of the all tokens neural network and the 10 tokens neural network are compared. Ten test subjects were asked to label hundred randomly sampled tweets which were above the threshold mentioned in section 4.4. In the first section we compare the average accuracy from subjects and in the second section we compare the accuracy in relation to a threshold.

Average accuracy

A Welch t-test for unequal variance is used to determine if there is a difference between the accuracies on the all tokens neural network and the 10 tokens neural network. The null hypothesis is formulated as follows:

The all tokens and 10 tokens neural network have the same identical average values.

The Welch t-test outputs a p-value of 0.03, giving enough evidence to reject the null hypothesis. The mean of the all tokens accuracy is 64.7% and the 10 tokens accuracy is 58.2%
We conclude that the all tokens accuracy is higher than the 10 tokens accuracy. The all tokens dataset performs better than the best ntoken dataset.

Accuracy and threshold

We now have determined the performance of the two methods using the accuracy measurement. Due to the large amount of data, we should check if the performance changes if we calculate accuracy using a threshold.

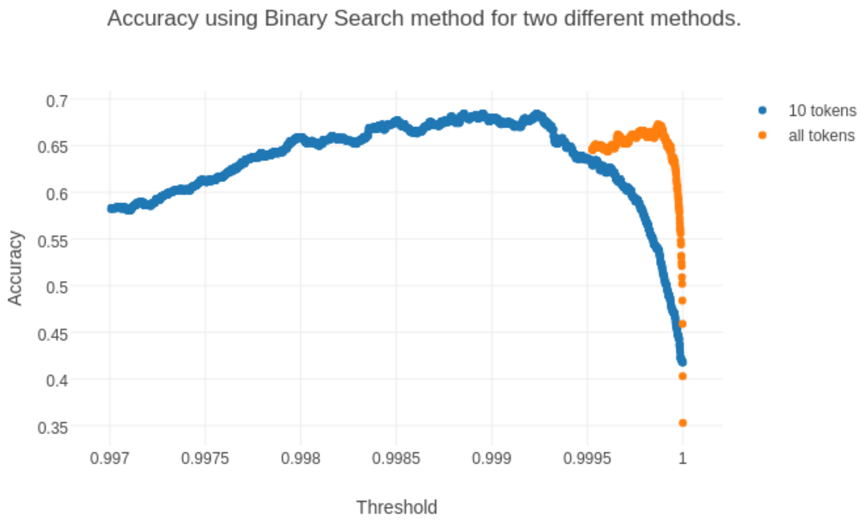


Figure 4.6: The accuracy decreases when the threshold increases. The amount of annotated data N for the 10 tokens model is 1000 and N for the all tokens model is 1000. The accuracy values for the all tokens model are not present below the threshold value of 0.9995 because the test subjects, using the binary search algorithm, did not find those threshold values.

In figure 4.6, we see an unexpected result. The 10 token neural network has a higher accuracy than the all tokens neural network. The result can be explained by the different probability distributions from the neural networks which influences the accuracy. To compare the two methods in a fair manner, we calculate the precision, recall and F1-score.

Precision, Recall and F1-score

In figure 4.7, we see the precision and recall for the two methods.

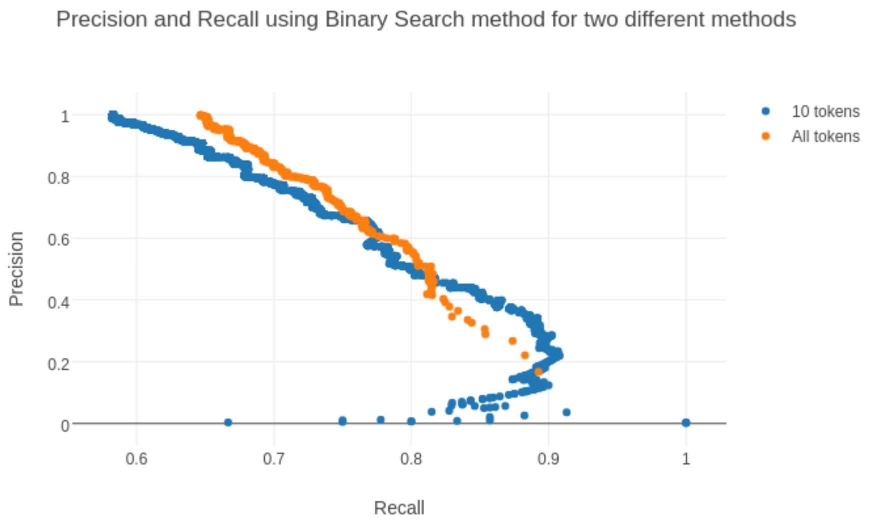


Figure 4.7: The precision and recall. The blue dots in the lower regions of precision, which have a high spread in recall are peculiar. These dots are however correct and reflect issues with working with real world data and test subjects. The amount of annotated data N for the 10 tokens model is 1000 and N for the all tokens model is 1000.

Traditionally, the highest precision and recall combination is chosen as the best model. In figure 4.7, it can be easily seen that the all tokens model is the best model with a precision 0.79 and a recall of 0.74.

In figure 4.8, we show the F1-score for the two methods for a clearer result.

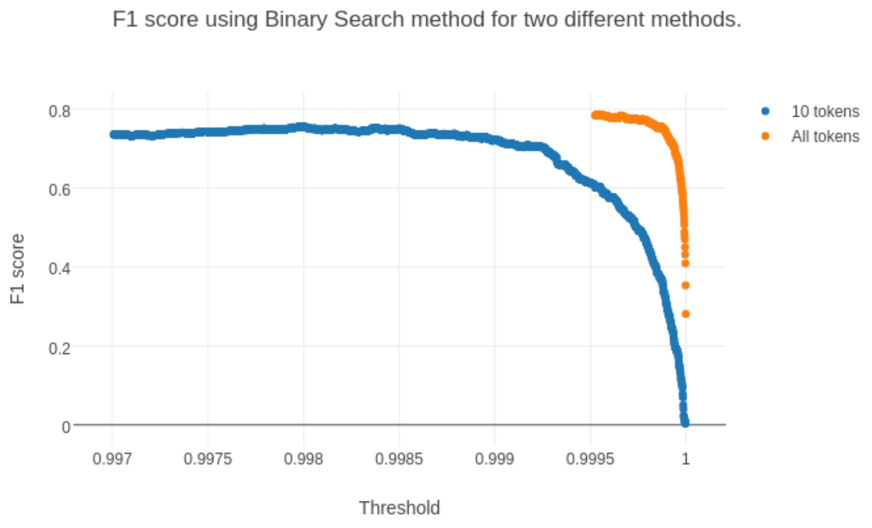


Figure 4.8: The F1-score of the 10 tokens model never has a higher maximal value than the all tokens model. N 10 tokens is 1000, N all tokens is 1000.

Interestingly, we conclude that the choice of evaluation measure influences the outcome of which method is better.

4.6 Kmeans

The Kmeans clustering algorithm is often used to segment data which do not have a label. A downside of this type of algorithm is that it finds clusters even if the data consist out of noise. Another downside is that the real-world cluster should be uniformly distributed and isotropically (uniformity in all orientations) shaped.

Having gathered the labels for the soccer class, we want to test the hypothesis that the soccer class is uniformly distributed and that the soccer tweets do not lie in a single soccer cluster. To test if the Kmeans assumption of uniformly distributed data holds, we use the Kolmogorov-Smirnov test. The Kolmogorov-Smirnov test[1] tests for uniformity by comparing a known distribution to the uniform distribution. The null hypothesis is defined as the two distributions being similar. The alternative distribution, using a two-sided test, is that the distributions are dissimilar.

The Kolmogorov-Smirnov test resulted in a p-value of 0.00, which gives evidence to the hypothesis that the distribution is not uniformly distributed.

The second hypothesis is formulated as follows:

The soccer tweets do not lie in a single cluster.

[1]`https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.kstest.html`

A Kmeans algorithm is tested on all the soccer data which were labeled by the test subjects. In figure 4.9, we show the decreasing Within Set Sum of Squared Error (WSSSE). The WSSSE is computed by calculating the squared distance from all the data in a cluster to the cluster centroid. The squared distances are calculated for each cluster and summed. The WSSSE is a measure for data dispersion in clusters. A low WSSSE means the dispersion is lower. A low WSSSE value is preferred to higher WSSSE values taken into account that k is not too large. A large k will reduce the WSSSE to zero when k is as large as the amount of datapoints in the data, but the clusters will not be meaningful. A trade-off has to be made.

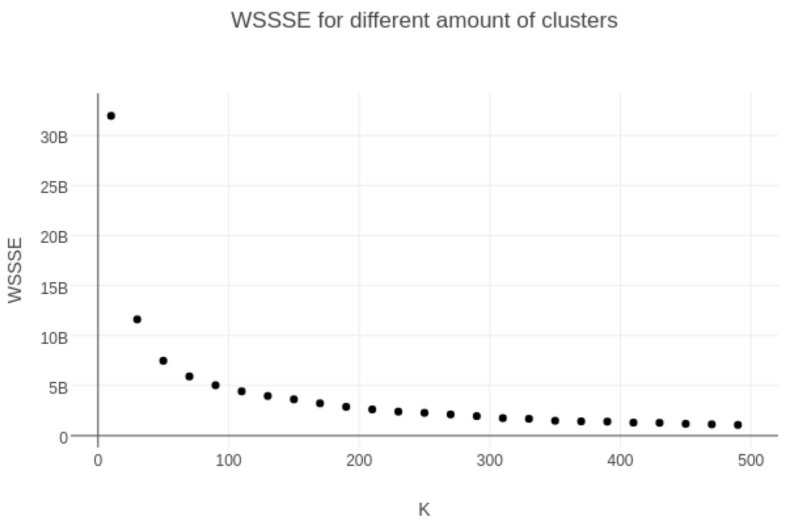


Figure 4.9: The Within Set Sum of Squared Error (WSSSE) is decreasing with an increasing amount of clusters.

In figure 4.10, it can be easily seen that all soccer tweets do not lie in a single cluster. On the contrary, the soccer tweets lie distributed amongst all the clusters.

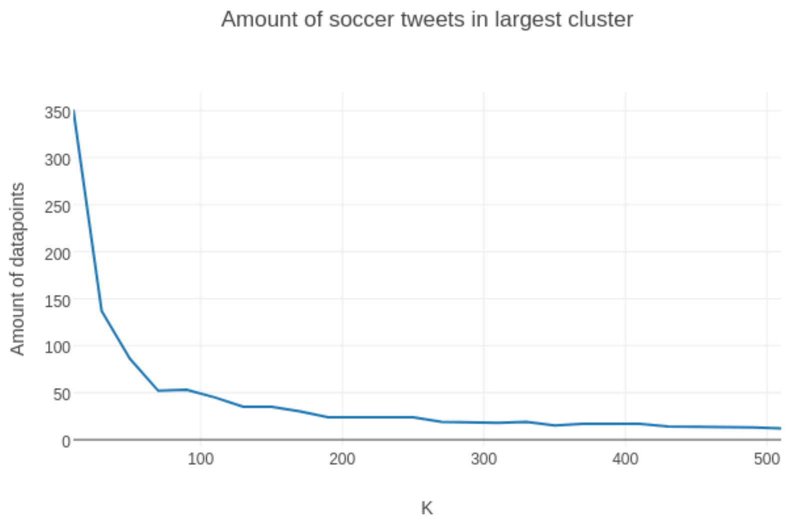


Figure 4.10: The maximum amount of soccer tweets in each cluster decreases with an increasing number of clusters.

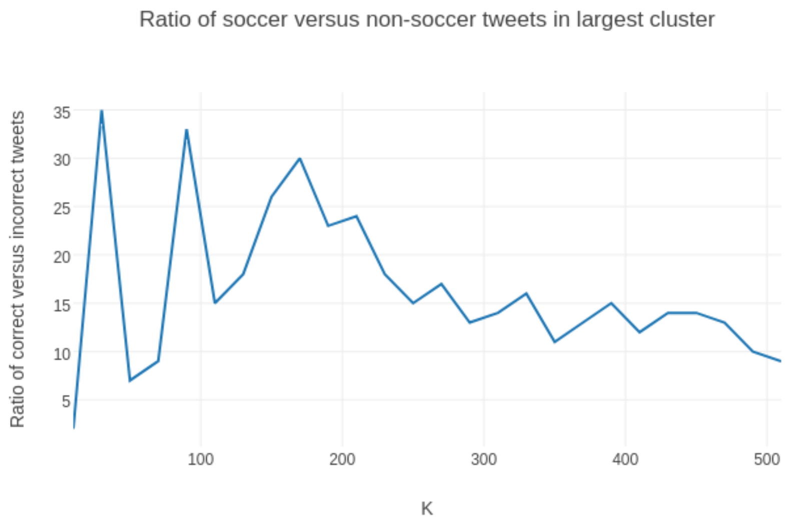


Figure 4.11: The ratio of soccer tweets versus non-soccer tweets for the cluster with highest amount of soccer tweets decreases generally as the amount of clusters increases.

From figure 4.10, we conclude that the soccer tweets do not lie in a single cluster. Soccer tweets could be in more than one cluster, therefore we look at the ratio of soccer tweets versus not soccer tweets. In figure 4.11, we show that the ratio soccer tweets versus non-soccer tweets decreases generally when k increases. When all soccer tweets lie in a single cluster and all non-soccer tweets lie in other clusters, the ratio would be large. In figure 4.11 we see that when k is small, the ratio is large. We would expect that for a sufficiently large k, we would find a cluster configuration which encompasses only the soccer tweets resulting in a large ratio. In fact, when there are no non-soccer tweets, the ratio would be infinite. Unfortunately this effect is not present.

We conclude this section with answering the second hypothesis: the soccer tweets do not lie in a single cluster. From figure 4.10 and 4.11, we can conclude that indeed the soccer tweets do not lie in a single cluster.

Part III

Conclusion and Further Work

Chapter 5

Conclusion

We set out some difficult research questions. Twitter data have not been studied intensively in the context of machine learning and the data itself contains a small amount of information. We did not expect high quality performance, as this research is intended as a baseline as well as giving insight into Twitter data segmentation using word2vec.

5.1 Missing data in Word2vec

In section 4.2, we discussed tweets which do not have enough tokens to be used in the word2vec model training. We have chosen to not use these tweets in the training of the model. In the results section, we showed that word2vec overcomes the problem of tweets which do not have enough tokens. The performance of tweets which have a small amount of tokens is low as we showed in figure 4.2. However if we combine all the tweets, thus not discriminating on token size, we find performance to be high.

We explain this result by tweets with a low amount of tokens not having enough evidence to discriminate between topics. The vectors for each token are averaged into a single vector and this vector does not capture the topic well. When using all the tweets, the added layer to the word2vec model is better able to discriminate.

Interestingly, when the amount of tokens is larger than thirteen, performance drops as well. An explanation could be that averaging the weight vector per token does not work well for these datasets. Perhaps an approach using the median or histogram works better. Another explanation could be that, coincidentally, this dataset has more noise or is less separable than others.

5.2 Experiment

We ran an experiment with ten subjects. The task in this experiment was to label tweets as being about soccer or about jihadism. To assess the knowledge about the topic, we asked the test subjects to grade themselves on how good an expert they thought they are on the topic of soccer. Test subjects with a grade higher than six were selected as experts. We showed that this subjective approach to assess a persons knowledge, results in a high inter-rater agreement.

We conclude that the subjective approach is a good method to select domain experts with an average Cohen's-Kappa score of 0.87.

5.3 Word2vec, neural network and seeds

Data annotation is a labor intensive task. To aid this process, we used the word2vec model to cluster data with a similar meaning. A word2vec model was implemented and trained on the SARA Hadoop super computer[1]. We showed that word2vec indeed finds similarly meaning words for the soccer subject.

Tweets which contained the word "voetbal" were selected as seeds to further segment the data in a second neural network layer. Similar tweets which did not contain the word "voetbal", but were about soccer had a high probability at the output node of the neural network. Unfortunately, performance was not high as a lot of tweets which had a high probability to be about soccer, were not about soccer. For this two-class problem, we were only able to reach an F1 score of 0.79%.

5.4 Binary search method

We hypothesized that the probabilities would increase the more a tweet was about soccer. We tested the hypothesis by asking subjects to label each tweet which lay above the threshold. If the tweet was not about soccer, it was increased and decreased if it was about soccer. When the change in threshold was small enough, or more than twenty annotations were needed, the method was stopped. We found that there was still much uncertainty in the model.

5.5 Kmeans

A Kmeans model was implemented and trained on the SARA Hadoop super computer[1]. We showed that Kmeans is not a good method to segment word2vec vectors. As a proof of concept, we asked subjects to annotate tweets which were about soccer or not about soccer. We trained a Kmeans classifier with K set to 500. Tweets labeled as soccer tweets by the subjects were attributed to each cluster. The membership was visualized and it was evident that tweets about soccer did not lie in a single cluster. On the contrary, soccer tweets were almost evenly spread amongst all clusters.

To quantify the claim that soccer tweets do not lie in a single cluster, we tested the assumption of Kmeans that the soccer data are uniformly distributed. Using a Kolmogorov-Smirnov test, we found strong evidence that this was not the case. The features were not uniformly distributed for the soccer cluster with according to the Kolmogorov-Smirnov test with $p = 0.00$. The soccer cluster is one of many clusters which contains tweets with similar semantic meaning. We understand that other topics could be uniformly distributed, however we showed that at least one is not. We conclude that Kmeans is not a good method for the soccer topic and possibly for other topics as well.

[1]`https://userinfo.surfsara.nl/systems/hadoop/description`

The downside of this approach is that we should test for membership and normality for different K. A commonly used approach for selecting the correct K is the elbow approach. However, it was not the goal to strive for completeness, we wanted to show and explain a possible downside of the Kmeans algorithm.

Chapter 6

Further work

In this chapter we explain what can be done to increase the model performance as well as other further work.

6.1 Dropout

A dropout procedure can be used to improve generalization performance by randomly omitting a fraction of the hidden units in hidden layers [27, 43]. Overfitting is a problem in (deep) neural networks. Dropout is a technique that randomly drops neural units, along with their connections, from the neural network during training. Dropping units prevents them from co-adapting too much.

6.2 Time feature

The distribution of Twitter messages throughout time is not taken into account. When time is added as a feature, discrimination between topics could become more clear. For example, tweets about soccer in the afternoon could indicate amateur soccer, where tweets in the night could indicate professional soccer (on television). Twitter does provide the time of the Tweet.

6.3 Geo-location feature

To aid the segmentation and classification process, other features can be used combined with the features presented in this thesis. Features such as time and geo-location could be of positive influence to the segmentation and classification. Ren et al., as well as [45] researched the use of geolocation features and how to impute geolocation data. Incomplete geolocation data for Twitter are handled by looking at the subject's friends on Twitter who do have geolocation data available. We tried to replicate their results and proposed a density based approach. Unfortunately, we were not able to replicate the results due to the gross of Dutch people not sharing their location information.

6.4 Users

Users can be used to further explore the topic at hand. Particular users could be experts on the topic and use words which are not used often but are domain specific. These words are not salient in the model or abundant in the data, but are of high importance. One of our test-subjects mentioned: ”some words are really domain specific, I think I am the only one who recognizes this to be about soccer ...”, indicating some evidence that this might be true.

6.5 Missing and noisy data

Data from Twitter are often noisy and incomplete. To be able to cope with spelling errors or slang abbreviations (”inb4” to ”in before”), methods such as the edit-distance and metaphones can be used. Metaphones [35] are word encodings based on their pronunciations. Metaphones [46, 47] could be used to overcome the problem of spelling errors or slang abbreviations.

6.6 Edit distance

The edit distance is a method which compares how different a word is from another word. The edit-distance method uses insertions, deletions and substitutions to determine how many characters are different between words. If the distance is small enough, the word can be replaced by a correct word from a certain vocabulary. This process can possibly improve classification by removing errors.

6.7 Word2vec and Kmeans parameters

The parameter settings used in this research could be researched further. As mentioned before, a different setting for K in Kmeans could result in tweets being in the same cluster. Different activation functions can also be evaluated.

6.8 Experiment website

The website was made in Flask[1]. During implementation, it became clear that this framework is not suitable for multi-user usage. Only a single user can use the system at a moment in time. The website only allows the labeling of a single topic. When multiple topics can be labeled, the model could discriminate topics better due to finding topic borders.

[1] `http://flask.pocoo.org/`

Appendices

Appendix A

Word2vec top 15 similar words to ”voetbal”

1. voetbal
2. voetbalnieuws
3. followback
4. international
5. voetbalnt
6. monchengladbach
7. transfergerucht
8. fenerbahce
9. eredivisie
10. vinl
11. voetbalheadlines
12. vi
13. gal
14. aquile
15. voetbalinformatie

Appendix B

Binary search method pseudocode

```
Data: Tweet with probability
Result: Threshold
Initialize threshold as 1;
Initialize previous threshold as threshold; Initialize low, high, center;
 Initialize Done flag False;
Initialize MaxIter 10;
while Still about soccer do
    Get tweet above threshold;
    Ask if Tweet is about soccer;
    if Tweet about soccer then
        threshold = threshold - 0.01;
    else
        Not about soccer;
    end
end
while Not stable and not Done and not MaxIter do
    Get tweet above threshold;
    Ask if Tweet is about soccer;
    if Tweet about soccer then
        high = center;
        low = high-(high-center)/2.0;
        center = (high - low)/2.0 ;
    end
    if Tweet not about soccer then
        high = center;
        low = low;
        center = center - (center - low)/2.0;
    end
end
```

Algorithm 1: Binary search for optimal threshold.

Bibliography

[1] Fuchun Peng and Dale Schuurmans. “Combining Naive Bayes and n-Gram Language Models for Text Classification”. In: *Advances in Information Retrieval: 25th European Conference on IR Research, ECIR 2003, Pisa, Italy, April 14–16, 2003. Proceedings.* Ed. by Fabrizio Sebastiani. Springer Berlin Heidelberg, 2003, pp. 335–350.

[2] Levent Bolelli, Şeyda Ertekin, and C. Lee Giles. “Topic and Trend Detection in Text Collections Using Latent Dirichlet Allocation”. In: *Advances in Information Retrieval: 31th European Conference on IR Research, ECIR 2009, Toulouse, France, April 6-9, 2009. Proceedings.* Ed. by Mohand Boughanem, Catherine Berrut, Josiane Mothe, and Chantal Soule-Dupuy. Springer Berlin Heidelberg, 2009, pp. 776–780.

[3] Matthew Hoffman, Francis R. Bach, and David M. Blei. “Online learning for latent dirichlet allocation”. In: *advances in neural information processing systems.* 2010, pp. 856–864.

[4] Loulwah AlSumait, Daniel Barbará, and Carlotta Domeniconi. “On-line LDA: Adaptive topic models for mining text streams with applications to topic detection and tracking”. In: *Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on.* IEEE. 2008, pp. 3–12.

[5] Wei Xu, Xin Liu, and Yihong Gong. “Document clustering based on non-negative matrix factorization”. In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval.* ACM. 2003, pp. 267–273.

[6] Omer Levy and Yoav Goldberg. “Neural word embedding as implicit matrix factorization”. In: *Advances in neural information processing systems.* 2014, pp. 2177–2185.

[7] Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. “Short Text Classification in Twitter to Improve Information Filtering”. In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’10. Geneva, Switzerland: ACM, 2010, pp. 841–842.

[8] Xiangmin Zhou and Lei Chen. “Event detection over Twitter social media streams”. In: *The VLDB Journal* 23.3 (2014), pp. 381–400.

[9] S. Chandra, L. Khan, and F. B. Muhaya. “Estimating Twitter User Location Using Social Interactions: A Content Based Approach”. In: *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on.* 2011, pp. 838–843.

[10] Farzindar Atefeh and Wael Khreich. “A Survey of Techniques for Event Detection in Twitter”. In: *Computational Intelligence* 31.1 (2015), pp. 132–164.

[11] Swati Agarwal and Ashish Sureka. “Using KNN and SVM Based One-Class Classifier for Detecting Online Radicalization on Twitter”. In: *Distributed Computing and Internet Technology: 11th International Conference, ICDCIT 2015, Bhubaneswar, India, February 5-8, 2015. Proceedings.* Ed. by Raja Natarajan, Gautam Barua, and Manas Ranjan Patra. Cham: Springer International Publishing, 2015, pp. 431–442.

[12] Jan H. Kietzmann, Kristopher Hermkens, Ian P. McCarthy, and Bruno S. Silvestre. “Social media? Get serious! Understanding the functional building blocks of social media”. In: *Business horizons* 54.3 (2011), pp. 241–251.

[13] Fred Morstatter, Jürgen Pfeffer, Huan Liu, and Kathleen M. Carley. “Is the sample good enough? comparing data from Twitter’s streaming API with Twitter’s firehose”. In: *arXiv* (2013).

[14] Paul W Farris, Neil Bendle, Phillip Pfeifer, and David Reibstein. *Marketing metrics: The definitive guide to measuring marketing performance.* Pearson Education, 2010.

[15] Yoav Goldberg and Omer Levy. “word2vec Explained: deriving Mikolov et al.’s negative-sampling word-embedding method”. In: *arXiv* (2014).

[16] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780.

[17] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems.* 2014, pp. 3104–3112.

[18] Liangjie Hong and Brian D. Davison. “Empirical Study of Topic Modeling in Twitter”. In: *Proceedings of the First Workshop on Social Media Analytics.* SOMA ’10. ACM, 2010, pp. 80–88.

[19] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. “Comparing Twitter and traditional media using topic models”. In: *Advances in Information Retrieval.* Springer, 2011, pp. 338–349.

[20] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. “Latent dirichlet allocation”. In: *the Journal of machine Learning research* 3 (2003), pp. 993–1022.

[21] Ankan Saha and Vikas Sindhwani. “Learning Evolving and Emerging Topics in Social Media: A Dynamic NMF Approach with Temporal Regularization”. In: *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining.* WSDM ’12. ACM, 2012, pp. 693–702.

[22] Michael Bryant and Erik B. Sudderth. “Truly nonparametric online variational inference for hierarchical Dirichlet processes”. In: *Advances in Neural Information Processing Systems.* 2012, pp. 2699–2707.

[23] Gang Qian, Shamik Sural, Yuelong Gu, and Sakti Pramanik. “Similarity between Euclidean and cosine angle distance for nearest neighbor queries”. In: *Proceedings of the 2004 ACM symposium on Applied computing.* ACM. 2004, pp. 1232–1237.

[24] Michael Steinbach, George Karypis, and Vipin Kumar. “A comparison of document clustering techniques”. In: *KDD workshop on text mining.* Vol. 400. 1. Boston. 2000, pp. 525–526.

[25] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. “OPTICS: Ordering Points To Identify the Clustering Structure”. In: *ACM Sigmod record.* Vol. 28. 2. ACM. 1999, pp. 49–60.

[26] Xin Rong. “word2vec Parameter Learning Explained”. In: *arXiv preprint arXiv:1411.2738* (2014).

[27] G. E. Dahl, T. N. Sainath, and G. E. Hinton. “Improving deep neural networks for LVCSR using rectified linear units and dropout”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing.* 2013, pp. 8609–8613.

[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. “Efficient estimation of word representations in vector space”. In: *arXiv* (2013).

[29] Z. Harris. “Distributional structure”. In: *Word 10* 23 (1954), pp. 146–162.

[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems.* 2013, pp. 3111–3119.

[31] Burr Settles. “Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics. 2011, pp. 1467–1478.

[32] Burr Settles and Xiaojin Zhu. “Behavioral factors in interactive training of text classifiers”. In: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Association for Computational Linguistics. 2012, pp. 563–567.

[33] Burr Settles. “Active learning literature survey”. In: *University of Wisconsin, Madison* 52.55-66 (2010), p. 11.

[34] Anthony J. Viera and Joanne M. Garrett. “Understanding interobserver agreement: the kappa statistic”. In: *Fam Med* 37.5 (2005), pp. 360–363.

[35] B.P. Pande and H.S. Dhami. “Application of natural language processing tools in stemming”. In: *International Journal of Computer Applications* 27.6 (2011), pp. 14–19.

[36] Richard Socher, Cliff C. Lin, Chris Manning, and Andrew Y. Ng. “Parsing natural scenes and natural language with recursive neural networks”. In: *Proceedings of the 28th international conference on machine learning (ICML-11).* 2011, pp. 129–136.

[37] Robert Tibshirani, Guenther Walther, and Trevor Hastie. "Estimating the number of clusters in a data set via the gap statistic". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.2 (2001), pp. 411–423.

[38] Simon Tong and Daphne Koller. "Support vector machine active learning with applications to text classification". In: *The Journal of Machine Learning Research* 2 (2002), pp. 45–66.

[39] Leonidas Lefakis. "Transductive Learning for Document Classification and Handwritten Character Recognition". Master's thesis. Utrecht University, 2008.

[40] Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. "ActNeT: Active Learning for Networked Texts in Microblogging." In: *SDM*. Citeseer. 2013, pp. 306–314.

[41] David Cohn, Les Atlas, and Richard Ladner. "Improving generalization with active learning". In: *Machine Learning* 15.2 (1994), pp. 201–221.

[42] Erik Ordentlich, Lee Yang, Andy Feng, Peter Cnudde, Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, and Gavin Owens. "Network-Efficient Distributed Word2vec Training System for Large Vocabularies". In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM. 2016, pp. 1139–1148.

[43] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[44] Pengjie Ren, Zhumin Chen, Jun Ma, Zhiwei Zhang, Luo Si, and Shuaiqiang Wang. "Detecting temporal patterns of user queries". In: *Journal of the Association for Information Science and Technology* (2015).

[45] Ryan Compton, David Jurgens, and David Allen. "Geotagging one hundred million Twitter accounts with total variation minimization". In: *Big Data (Big Data), 2014 IEEE International Conference on*. IEEE. 2014, pp. 393–401.

[46] MA Li-dong. "Metaphone Phonetic Matching Algorithm and its Application". In: *Computer Era* 10 (2010), p. 017.

[47] Lawrence Philips. "Hanging on the metaphone". In: *Computer Language* 7.12 (December) (1990).