BLEKINGE TEKNISKA HÖGSKOLA

BTH

DEGREE PROJECT FOR MASTER OF SCIENCE IN ENGINEERING
DEGREE OF MASTER OF SCIENCE IN ENGINEERING: COMPUTER SECURITY

Email Classification with Machine Learning and Word Embeddings for Improved Customer Support

Jim Ahlstrand | Oliver Rosander

Blekinge Institute of Technology, Karlskrona, Sweden 2017

Supervisors: Martin Boldt & Anton Borg, Department of Computing, BTH

Abstract

Context. Classifying emails into distinct labels can have a great impact on customer support. By using machine learning to label emails the system can set up queues containing emails of a specific category. This enables support personnel to handle request quicker and more easily by selecting a queue that match their expertise.
Objectives. This study aims to improve the manually defined rule based algorithm, currently implemented at a large telecom company, by using machine learning. The proposed model should have higher F_1-score and classification rate. Integrating or migrating from a manually defined rule based model to a machine learning model should also reduce the administrative and maintenance work. It should also make the model more flexible.
Methods. By using the frameworks, TensorFlow, Scikit-learn and Gensim, the authors conduct five experiments to test the performance of several common machine learning algorithms, text-representations, word embeddings and how they work together.
Results. In this article a web based interface were implemented which can classify emails into 33 different labels with 0.91 F_1-score using a Long Short Term Memory network.
Conclusions. The authors conclude that Long Short Term Memory networks outperform other non-sequential models such as Support Vector Machines and ADABoost when predicting labels for emails.

Keywords: Email Classification, Machine Learning, Long Short Term Memory, Natural Language Processing

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ADA Adaptive Boosting. 3, 12, 13, 27, 31–36, 38

ADAM Adaptive Moment Estimation. 26

ANN Artificial Neural Network. 12, 13, 27, 31–36, 38, 41, 47

API Application Programming Interface. v, vi, 44–46

AvgWV Average Word Vector. 5, 10, 27, 31–33, 35, 36, 41, 47

BoW Bag of Words. v, 3, 5, 7, 8, 19, 20, 31–33, 35, 36, 47

BoWBi Bag of Words Bi-gram. 8, 19, 31–33, 35, 36, 47

CBoW Continuous Bag-of-Words. 3, 9, 19, 25, 28, 29

CTC Connectionist Temporal Classification. 3

DT Decision Tree. 3, 12, 27, 31–36, 38

FN False Negative. 21

FP False Positive. 21

GloVe Global Vectors. 3, 9, 10, 19, 25, 28, 29, 31, 41, 46

HTML Hypertext Markup Language. 17, 20, 44

HTTP Hypertext Transfer Protocol. 44

IDF Inverse Document Frequency. 8

ITS Issue Tracking System. x

JSON JavaScript Object Notation. 44

LSTM Long Short Term Memory. iv, v, 1–5, 11, 12, 14, 23–27, 29–31, 35, 38–48

MVC Model-View-Controller. ix

NB Naive Bayes. 12, 27, 31–38

NLP Natural Language Processing. 1–4, 6, 7, 19, 23, 25, 26, 29, 41, 43

OCR Optical Character Recognition. 6

ReLU Rectified Linear Unit. 13

RNN Recurrent Neural Network. 1, 3, 13, 14

SGD Stochastic Gradient Descent. 24, 26, 40

SVM Support Vector Machines. 3, 12, 27, 31–38, 41, 44, 47

TF Term Frequency. 8

TN True Negative. 21

TP True Positive. 21

Glossary

classification rate The frequency of emails that are classified with a valid label. 4

corpus A large set of structured text documents, normally one document per row. v, 3, 4, 9, 18, 23, 25, 26, 30, 31, 43

document Any source which contains text segments related to a shared topic, e.g. an email, blog post, Wikipedia page etc. 7, 8, 10, 11, 21

Flask A Python web framework for building Model-View-Controller (MVC) applications. 44

Gensim Gensim is an open source software library focusing on vector space and topic modelling. 16

one-hot-vector A vector with all elements set to zero except for one. 7

queue labels Aggregated version of the 33 email labels that consist of 8 different classes. 17

Scikit-learn Scikit-learn is an open source software library for simple and efficient data mining and data analysis. 16, 27

semantic The meaning or signification of a word or sentence. 1, 7–11, 25, 41, 42

sequential model A machine learning model that takes variable length time series input. 5

Språkbanken Språkbanken (the Swedish Language Bank) is a nationally and internationally acknowledged research unit at the Department of Swedish, University of Gothenburg. Språkbanken focuses on language technology, in particular methodologies for handling the Swedish language, and the development of linguistic resources and tools for Swedish. 25, 30, 46

support ticket A support request made from a customer to the company results in a ticket which is queued and processed by the company support team. A Issue Tracking System (ITS) is commonly used in order to manage support tickets. 1

syntactic A phrase that follows a correct syntax is syntactic, i.e grammatically correct. 10, 11, 25, 41

TensorFlow TensorFlow is an open source software library for numerical computation using data flow graphs. 16, 45

Chapter 1

Introduction

Communications is part of everyday business and it is vital for operations to run smoothly. Good communication is not limited to inside the business but also extends to the customer. Customer service in telecommunication businesses today is mainly based on email and chat correspondence. Each email from a customer is referred to as a support ticket. For small or medium sized companies it might be sufficient to have a single email inbox for which the whole support team collaborate on customer support tickets. However this approach is not scalable, as the company grow, the support team also grows. Consider a scenario with a large support team divided into smaller specialised teams that each handles different errands. In order to optimise the performance and minimise the time the support ticket spends in the system it is necessary to sort incoming tickets and assign them the correct support team. This task is both time consuming and labour intensive, however automating it is not a trivial task because of the complex natural languages that has to be understood by the program. Any program that is doing any processing of natural languages, i.e. a language used by humans to communicate, is performing Natural Language Processing (NLP) [1].

The human languages can be extremely complex where a single word can have several semantics depending on its context. Teaching a computer to understand the human language has shown to be a difficult task, but during recent years we have seen a steady development of software that utilise technologies such as machine learning for speech recognition, language translation, summarising and information extraction.

Automating email labelling and sorting requires a model that can differentiate between different types of errands and support requests. Such models must be able to do this even if the email contains spelling mistakes, previous conversations, irrelevant information, different formatting or simply rubbish. The LSTM model is an extended version of the Recurrent Neural Network (RNN) network which is a sequential model often used in text classification [2]. Word embedding models aim to model the words of a language in a vector space and placing words with similar semantic meaning close to each other [3, 4]. This helps the classifier to understand the meaning of the text and therefore improving its ability to predict the correct class [4]. In this report we propose a system that uses NLP together

with a LSTM classifier to tag emails based on the contents of the email. The tagged emails are then sent to the correct email queue where they are processed by the specialised support personnel.

This report is structured as follows. First the theoretical framework is presented in chapter 2, from which the experiments, discussions and conclusions are based on. Chapter 3 describes the method and how each experiment was conducted. The results are presented in chapter 4, followed by the discussion and conclusion which is presented in chapter 5 and 7 respectively.

1.1 Related work

In the Email Statistics Report, 2016-2020[1], a report from The Radicati Group, Inc, it is concluded that the email usage continue to grow worldwide. During 2016 there were 2.6 billion active email users, and in 2020 they expect there to be 3.0 billion email users. The expected number of business and consumer email sent each day will increase with an annual rate of 4.6%, from 215.3 billion to 257.5 billion emails.

Managing the increased number of emails is important for a company, and managing them well is even more important. Bougie, Pieters and Zeelenberg evaluate how the feeling of anger and dissatisfaction affect the customers reactions to service failure across the industry [5]. The intuitive notion that anger or unfulfillment can make the customer change provider is confirmed. An effective and accurate email classification is therefore a useful tool for the overall quality of the customer support.

The severity of the dissatisfaction is also an important factor. If the customer experience a minor dissatisfaction they are not prone to complain. If they experience moderate levels of dissatisfaction then it is possible for the company to win back the customer and turn the dissatisfaction to a positive experience. If they experience a major dissatisfaction they are more prone to complaining even though actions are taken from the companies side [6].

Coussement and Van den Poel propose an automatic email-classification system that is intended to separate complaints from non-complaints. They present a boosting classifier which labels emails as either complaints or non-complaints. The authors also argues that the use of linguistic features can improve the classification performance [7]. Selecting a corpus to train word vectors that are used by sequential models is not a trivial task.

The use of domain-specific language are shown by Coden et al. to improve the NLP model used for part-of-speech tagging from 87% accuracy to 92%. Even though this is not the same task as training word embeddings, it can give an indication that including domain-specific language in the corpus can improve the

[1] `https://www.radicati.com/wp/wp-content/uploads/2016/01/Email_Statistics_Report_2016-2020_Executive_Summary.pdf`

model [8]. The word embeddings are supposed to model the language, but finding a large enough corpus that represent the domain in which they are used is difficult.

Word vectors trained on huge corpora, such as Google News which is trained on about 100 billion words, are available to the public but they are only trained on English. Fallgren, Segeblad and Kuhlmann have evaluated the three most used word2vec models, BoW, skipgram and Global Vectors (GloVe), on a Swedish corpus. They evaluate their word vectors on the Swedish Association Lexicon. They show that Continuous Bag-of-Words (CBoW) perform best with a dimension of 300 and 40 iterations [9].

Nowak et al. show that LSTM and bi-directional LSTM perform significantly better when detecting spam and classifying Amazon book reviews compared to the non-sequential approach with Adaptive Boosting (ADA) and BoW [10].

Yan et al. describe a method of multi label document classification by using word2vec together with LSTM and Connectionist Temporal Classification (CTC). Their model is evaluated on different datasets including emails and produce promising results compared to other versions of both sequential deep learning models such as RNN and non-sequential algorithms such as Support Vector Machines (SVM). Their research tries to solve the problems with multi-label classification by first representing the document with a LSTM network, then training another LSTM network to represent the ranked label stream. Finally they apply CTC to predict multiple labels [11].

Gabrilovich and Markovitch compare SVM with the C4.5, a Decision Tree (DT) algorithm on text categorisation. The C4.5 algorithm outperform SVM by a large margin on datasets with many redundant features. They show that the SVM can achieve better results than the C4.5 algorithm by removing the redundant features using aggressive feature selection [12].

Measuring the energy consumption used by an algorithm is hard. However measuring the execution time is trivial but useful. Bayer and Nebel conclude that the fastest algorithm is not always the one that consumes the least amount of energy, however in most cases it is true that shorter execution time consumes less energy [13].

1.2 Research gap

There exists research on several of the topics required to successfully classify emails [11], i.e models that interpret natural language[3, 14, 15, 16] and classifiers that utilise the relations of words in a time series [2]. When researching related work, the majority of the recent work has been studied on the English language and little research has been conducted on the Swedish language. Little research has also been conducted in terms of email classification and how to best utilise the NLP and machine learning models within said context. Baron argues that emails belong to a distinct group of documents by stating that they are informal, enables

a levelled playing field in terms of social hierarchy, encourages personal enclosure and can become emotional [17]. These distinctions may have to be accounted for when creating the machine learning model.

1.3 Research questions

1. **To which degree does the NLP model (e.g word2vec) affect the classifiers classification performance?**
   Motivation: *Investigating the pre-processing algorithms effect on the classification performance is important when drawing conclusion on which model to use and for future work.*

2. **How well can LSTM, compared to non-sequential machine learning models, classify emails?**
   Motivation: *Investigating how well the classification model can take advantage of the data is important, answering the question will answer if sequential models can take advantage of the data better than non-sequential models.*

3. **Does an aggregation of the labels increase the performance metric compared to having all labels separated?**
   Motivation: *Measuring the performance between the two label sets is useful information about the expected performance gain that can be achieved by aggregating the labels into a set of more distinct labels.*

4. **To which degree does the corpora affect the LSTM performance?**
   Motivation: *Answering how important it is to build a good corpus is useful when deciding which model is to be used if there exists few good corpora.*

5. **To which degree does the LSTM network size and depth affect the classifier performance?**
   Motivation: *Knowing how to tune a LSTM is not easy. Answering this question will help researchers and machine learning experts tune their network better if trained in a similar environment.*

1.4 Objectives

The objective of this work is to increase the performance of the automated email classification and sorting process at a large telecommunication company that support this thesis. In order to improve the current model our proposed model should have greater or equivalent F_1-score and a better classification rate. The possibility to integrate the model in the current production environment should also be delivered.

The telecommunication company will supply all the emails and the infrastructure required to evaluate the classifiers. The proposed framework will be evaluated by domain experts in a test environment where it is compared against the current rule-based classifier. If the suggested model is performing better than the current classifier and if it adds business value to said company it will be integrated into the production chain.

The research questions will enable researchers to continue to make progress on Swedish email classification. The framework, although owned by the telecommunication company, will serve as a basis for further experiments on multi label emails and semantic analysis.

1.5 Delimitations

The emails could be structured in chains and contain several emails that are sent back and forth between the customer and the support personnel. The models could theoretically use this to decide the context and how the conversations change subject. The emails our model use do not contain these chains, they are separated and classified individually. The model will be restricted to and there will be no classification or training performed on any document types other than emails.

The non-sequential models are used as a baseline for comparison with the LSTM network. The parameters of these models are not optimised and do not use preprocessing techniques such as lemming, stemming or stop word removal. The BoW or Average Word Vector (AvgWV) text representation models used for the non-sequential models are also not optimised. The BoW hyperparameters filter the numbers of words within a relative range, i.e sub- and supersampling. This is done to make the experiments possible with all non-sequential models.

The corpus is built to contain a recommended amount of words, i.e enough to train general word vectors. Specialised word vectors can be trained with less words but for general vectors the more words in the corpus the better [15]. The content of the corpus is therefore not reviewed or optimised because of its relative large size of just over one billion words.

Due to the vast amount of computational power required to run the algorithms, extensive hyperparameter testing is not practical. Instead a set of common hyperparameters were used during the experiments and no further optimisation of hyperparameters was performed.

Chapter 2

Theoretical framework

The theoretical framework covers most of the theory behind the models and algorithms used in this paper. The models and algorithms are explained as well as the underlying theory that defines them.

2.1 Natural language processing

Whenever humans communicate we are transferring more than just a set of instructions. We have the ability to communicate experiences, knowledge, feelings, information and more. We can do this by using a medium of mutually known signs, tokens and rules. This medium can be text, speech, body language or, more commonly, a combination of all three. This is normally referred to as natural languages, i.e. any language used by humans to interact with other humans [1].

A computer that takes any form of natural language and processes it in any way is using NLP. For example, when the bank scan a check, or the post office scan the address of an email, they are using Optical Character Recognition (OCR) which is a branch of NLP. Another example is voice commands, which can be used to search the internet or create notes without even touching your phone [1].

With the use of natural languages we can communicate effectively across many domains and situations. However because natural languages are mostly ambiguous it creates a difficult barrier for computers. Take the phrase "The trophy did not fit in the bag, it was too big" for example, what does the "it" refer to, the bag or the trophy? This may seem like a trivial question for a human because we know that big things does not fit into smaller things. A word, e.g. "it" in this case, can have several different meanings depending on the context. If we change the phrase into "The trophy did not fit in the bag, it was too *small*" the "it" now refers to the bag instead.

2.2 Text representation

Looking at machine learning classification with a broad perspective shows a common factor between all classifiers. The general principle revolves around

projecting data in a n-dimensional space and then separating that space into subspaces which are labelled with a specific class. The first step, projecting the data, varies greatly depending on the different types of data. However a general requirement is that the projection have to be of fixed output length, i.e. if you want to project a document you have to make sure that the result is of the same dimension regardless of the document length.

In order for a text document to be projected into a n-dimensional space we need to consider the fact that documents contain texts of variable length. The texts themselves also consists of words of variable length. In order to manage the words it is common to build a dictionary of fixed length. The words can then be represented as a one-hot-vector. Depending on the NLP model these vectors are managed differently. There are three common categories of NLP models when it comes to text processing, *count based*, *prediction based* and *sequential*. Count based methods are based on the word frequencies with the assumption that common words in a document have significant meaning to the class. Prediction based methods models the probabilistic relations between words, e.g. in the context of "The cat drank ..." the words milk and water are more likely than car and shoe. Sequential models are based on the assumption that a sequence, or stream, of words are significant to the documents semantic meaning. Sequential models are often combined with prediction based models to better capture the linear relations together with the sequential order of the words.

2.2.1 Preprocessing

In the preprocessing step the documents are transformed from the raw document to a structured document that is intended to contain as much information as possible without discrepancies that can affect the prediction result. A common method to increase the information density of a document is to remove the words that are very common and rarely has any significance, often referred to as stop words. These are word such as "the", "are", "of", which are insignificant in a larger context. In BoW these are a list of predetermined words, but word2vec take a probabilistic approach, called subsampling, which avoid overfitting on the most frequent words. For each instance of a word in word2vec a probability of removal is decided by equation 2.1 where t, usually set to 10^{-5}, is the threshold and $f(w_i)$ is the word frequency [14].

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{2.1}$$

In BoW these words are discarded from becoming features and in word2vec they are sometimes skipped, i.e not included in the context window used for training.

In a corpus of millions of words there will be some outliers, e.g. random sequences of numbers, noise, misspellings etc. As these words are very uncommon

and often does not appear more than a couple of times it is common to enforce a minimum count before adding words to the dictionary.

2.2.2 Bag of words

A commonly used method to model the meaning of a document is BoW which outputs a fixed-length vector based on the number of occurrences of terms. A frequently used term would indicate that the document has to do more with that term and should therefore be valued higher than the rest of the terms within the document. This is achieved by calculating the occurrences of each term in the document i.e. a Term Frequency (TF). The TF models the document in a vector space based on the occurrence of each term within the document. Downsides of this simple model is that it does not contain information about the semantic of each term and it does not contain information about the context of the terms either. Another fact to note is that all terms have the same weights and are therefore seen as equally important when modelling the document, even though there is a possibility it is not the case [16].

	the	power	was	missing	factor	sadly	we	are	wisdom
d_A	2	1	1	1	1	0	0	0	0
d_B	1	0	0	1	0	1	1	1	1

Table 2.1: Word frequency counts example

In BoW the documents d_A = "the power was the missing factor" and d_B = "sadly we are missing the wisdom" would be modelled by the features and the final vector representation seen in table 2.1.

To capture the context of words in a BoW model it is common to combine the terms in a document in a model called bag of n-grams. These n-grams are combinations of tokens found in the documents. A Bag of Words Bi-gram (BoWBi) model based on the document d_A mentioned previously would have the following features:

the power, power was, was the, the missing, missing factor

Inverse Document Frequency (IDF) weighting scheme is introduced to solve the problem of equally weighted terms. The document frequency df_t, is defined by the number of documents that contain a term t. If a term t has a low frequency and appears in a document d then we would like to give the term a higher weight, i.e increase the importance of the term t in the document. The IDF weight is therefore defined as shown in equation 2.2 where N is the total number of documents [16].

$$idf_t = log\frac{N}{df_t} \tag{2.2}$$

2.2.3 Word2Vec

The Word2Vec model is based on the assumption that words with similar semantics appear in the same context. This can be modelled by placing a word in a high dimensional vector space and then moving words closer based on their probabilities to appear in the same context. There are mainly three different methods to calculate these vectors, CBoW [3], skipgram [14], and GloVe [15]. A relatively large corpus is required for these models to converge and achieve good results with word vectors, normally around one billion words or more.

CBoW

The CBoW method is based on the principle of predicting a centre word given a specific context. The context is in this case the n-history and n-future words from the centre word, where n is determined by the window size. The structure of CBoW is somewhat familiar to auto-encoders, the model is based on a neural network structure with a projection layer that encodes the probabilities of a word given the context. The goal is to maximise the log probabilities which makes CBoW a predictive model. The projection layer and its weights is what later becomes the word vectors. However in order to feed the network with words you first have to encode the words into one-hot-vectors which is defined by a dictionary. This dictionary can be over a million words while the projection layer typically range from anywhere between 50 and 1000 nodes [3, 18].

Skipgram

The skipgram model is similar to the CBoW model but instead of predicting the centre word given the context, skipgram predicts the context given the centre word. This allows the skipgram model to generate a lot more training data which makes it more suitable for small datasets, however it is also several magnitudes slower than CBoW [14].

Skipgram n-gram

The skipgram n-gram model is based on skipgram but instead of using a dictionary with complete words it uses variable lengths n-grams. Other models rely on the dictionary to build and query vectors, however if the word is not in the dictionary the model is unable to create a vector. The skipgram n-gram model can construct word vectors for any words based on the n-grams that construct the word. The model has slightly lower overall accuracy but with the benefit of not being limited to the dictionary.

GloVe

The GloVe model does not use neural networks to model the word probabilities but instead relies on word co-occurrence matrices. These matrices are built from the global co-occurrence counts between two words. GloVe then performs dimensionality reduction on said matrix in order to produce the word vectors. Let X be the co-occurrence matrix where X_{ij} is the number of times word j occurs in the context of word i. Let $X_i = \sum_k X_{ik}$ be the number of times any word appears in the context of i. The probability that word j appears in the context of i can now be calculated as following

$$P_{ij} = P(j|i) = \frac{X_{ij}}{X_i} \tag{2.3}$$

This makes GloVe a hybrid method as it models probabilities based on frequencies [15].

Average word vector

AvgWV is a document representation in which a document is represented by a vector constructed from the average of the word vectors of each word in the document. The word vectors are averaged to create a vector of the same dimension as the word vectors. Equation 2.4 describes how the AvgWV is calculated. n is the numbers of words in the document and w_i is the corresponding word vector of a word. The method of aggregating the word vectors is well known and is a simple way to incorporate the semantic meaning of the words [19].

$$\frac{1}{n}\sum_{i=0}^{n} w_i \tag{2.4}$$

2.2.4 NLP evaluation

The relations between words in the vector space reveal some interesting connections. Consider the words “big” and “bigger”. These two words have a distance between them in the vector space denoted A. Now consider the words “fun” and “funnnier”, which have another distance between them denoted B. The word *big* relates to *bigger* the same way as *fun* relates to *funnier*, and it turns out that this relation is encoded in the vectors. With well trained word vectors, distance A will be almost the same as B. It is also possible to ask the question “Which word relates to fun, in the same way that big relates to bigger?” and predict that word using simple vector operations.

$$V_{big} - V_{bigger} + V_{fun} \approx V_{funnier} \tag{2.5}$$

These analogies can be formulated as either syntactic or semantic questions. Syntactic analysis focus on assessing the correct meaning of a sentence while

a semantic analysis focuses on assessing grammatically correct sentences. An example of a syntactic question could be "run is to running as walk is to ...?", and a semantic question could be "Stockholm is to Sweden as Berlin is to ..?". By predicting the missing word it is possible to calculate the accuracy of the word vectors and how well they model the semantic and syntactic structure of the words [15, 3].

2.3 Classification

Single-label text categorisation (classification) is defined as the task of assigning a category to a document given a predefined set of categories [20]. The objective is to approximate the document representation such that it coincides with the actual category of the document. If a document can consist of several categories we need to adapt our algorithm to output multiple categories, which is called multilabel classification. The task is then to assign an appropriate amount of labels that correspond with the actual labels of the document [20].

A fundamental goal of classification is to categorise documents that have the same context in the same set, and documents that do not have the same context in separate sets. This can be done with different approaches that involve machine learning algorithms. Machine learning algorithms learn to generalise categorises from previously seen documents which is later used to predict the category of previously unseen documents. Peter Flach explain in his book, "Machine learning: the art and science of algorithms that make sense of data" three different groups of machine learning algorithms namely, geometrical, probabilistic and logic based models [21]. The different groups of classifier achieve the same goal, but their methods are different. These classifiers are hereafter referred to as non-sequential classifiers since they do not handle the words in the emails in a sequence. A sequential classifier, such as LSTM, handle each word in the email sequential, which allows it to capture relations between words better and possibly utilise the content of the email better than a non-sequential classifier.

Geometric based classifiers split the geometric space in different parts where each subspace belongs to a class [21]. The boundaries are optimised to reduce the number of wrongly classified training instances. Probabilistic classifiers models probabilistic relationships between the features and the classes [21]. They calculate the probability of $P(Y|X)$ where X is known and Y is the class. Rule based model build a tree based on rules where each node has several child nodes based on the rules constructed [21]. The tree is segmented by iterations where each child is partitioning the instance space. A leaf is then assigned a class, value, probability or whatever is preferred.

2.3.1 Machine learning classifiers

The machine learning models are selected based on their group, diversity and acceptance in the machine learning community. SVM, Naive Bayes (NB) and DT are from three different groups of classifiers. ADA is used test a boosting classifier and Artificial Neural Network (ANN) is used to compare a non-sequential neural network against a sequential neural network, LSTM.

Support vector machine

SVM are based on the assumption that the input data can be linearly separable in a geometric space [22]. This is often not the case when working with real word data. To solve this problem SVM map the input to a high dimension feature space, i.e hyperplane, where a linear decision boundary is constructed in such a manner that the boundary maximises the margin between two classes [22]. SVM is introduced as a binary classifier intended to separate two classes when obtaining the optimal hyperplane and decision boundary.

Decision tree

A DT classifier is modelled as a tree where rules are learned from the data in a if-else form. Each rule is a node in the tree and each leaf is a class that will be assigned to the instance that fulfil all the above nodes conditions. For each leaf a decision chain can be created that often is easy to interpret. The interpretability is one of the strengths of the DT since it increases the understanding of why the classifier made a decision, which can be difficult to achieve with other classifiers. The telecommunication company is today using a manually created decision tree, in which the rules are based on different combinations of words.

Naive bayes

NB is a probabilistic classifier which is build upon the famous Bayes' theorem $P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$, where A is the class and B is the feature vector [23, 24, 25]. The probabilities of $P(B|A)$, $P(A)$ and $P(B)$ are estimated from previously known instances, i.e training data [23, 25]. The classification errors are minimised by selecting the class that maximises the probability $P(A|B)$ for every instance [25].

The NB classifier is considered to perform optimal when the features are independent of each other, and close to optimal when the features are slightly dependant [23]. Real world data does often not meet this criteria but researchers have shown that NB perform better or similar to C4.5, a decision tree algorithm in some settings [23]. The researchers argue that NB performs well even when there is a clear dependency between the features, making it applicable in a wide range of tasks.

AdaBoost

ADA is built upon the premise that multiple weak learners that perform somewhat good can be combined using boosting to achieve better result [26]. This algorithm perform two important steps when training and combining the weak classifiers, first it decided which training instances each weak classifier should be trained on, and then it decides the weight in the vote each classifier should have.

Each weak classifiers is given a subset of the training data that where each instance in the training data is given a probability that is decided by the previous weak classifiers performance on that instance. If the previous weak classifiers have failed to classify the instance correct it will have a higher probability to be included in the following training data set.

The weight used in the voting is decided by each classifiers ability to correctly classify instances. A weak classifier that perform well is given more influence than a classifier that perform bad.

2.3.2 Deep learning classifiers

Artificial neural network

The artificial neural network is based on several layers of perceptrons, also known as neurons, connected to each other. A perceptron is a linear binary classifier consisting of weights and a bias[21]. Connecting several perceptrons in layers allows accurate estimations of complex functions in multi dimensional space. Equation 2.6 describes the output of a single perceptron where W is the weights, X is the input vector, b is the bias and a is the activation function[1].

$$a(W \times X + b) \tag{2.6}$$

The weights and biases in ANN has to be tweaked in order to produce the expected outcome. This is done when training the network which normally is done with backpropagation. The backpropagation algorithm is based on calculating the gradients given a loss function and then edit the weights accordingly given a optimisation function. Normally ANN is designed with a input layer matching the size of the input data, a number of hidden layers and finally a output layer matching the size of the output data.

Recurrent neural net

RNNs are based on ANN, however it not only considers the current input but also the previous input. It does this by connecting the hidden layer to itself. A recurrent network contains a state which is updated after each time step, this

[1]Normally Softmax or Rectified Linear Unit (ReLU) is used as activation functions but several others exists.

allows recurrent networks to model arbitrary lengths of sequential or streamed data e.g. video, voice and text. The network starts with a zero state which then is updated based on the weights, biases and the fixed length input after each time step. Equation 2.7 describes the hidden layer h at time t from the RNN network. Equation 2.8 describes the output layer of the RNN network [27].

$$h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{2.7}$$

$$y_t = W_{hy}h_t + b_y \tag{2.8}$$

Training the RNN is normally done by estimating the next probable output in the sequence and then alter the weights accordingly. However consider a stream of data for which a prediction is done at each time step, each prediction will be based on the current input and all previous inputs. This makes it very hard to accurately train the network as the gradients will gradually vanish or explode the longer the sequences are [2].

Long short term memory

The LSTM network was developed in order to avoid the gradient problems introduced in RNN [2, 10, 28, 29]. LSTM introduces a *forget gate* and an *input gate* which both acts as filters. The forget gate determines what to disregard or forget from the current cell state. The input gate determines what to add from the input to the current cell state. The input gate together with a tanh layer is what produces the new cell state after the forget gate has been applied. In this way the LSTM network models a more accurate state after each time step as the new gates gives it a "focus span" for which redundant information eventually gets filtered out. This also reduces the effects of exploding and vanishing gradients. There are several variants of the LSTM network with peepholes and other features which further expands the networks capabilities [10].

Cross entropy loss

The cross entropy loss is a logarithmic measurement of how wrong the model is predicting the output compared to the ground truth. Being logarithmic, it will punish estimations that are far from the ground truth. As the predictions become better they are receiving a substantial decrease in loss. The cross entropy loss is used in training of the LSTM to reduce the discrepancy between the predicted value and the ground truth. Minimising the cross entropy loss will lead to predictions that are closer to the ground truth.

$$H_x(y) = \sum_{i,j} p(i,j) log_b p_i(j) \tag{2.9}$$

$$L(w) = \frac{1}{N} \sum_{n=1}^{N} H_{x_n}(y_n) \tag{2.10}$$

The Shannon entropy function, seen in figure 2.9, is the base upon which the cross entropy loss is defined [30]. Equation 2.10 defines the cross entropy loss.

Gradient descent optimiser

The gradient descent algorithm is an optimiser which minimises a cost function C with the objective to reduce the training error E_{train}. The cost function is defined as the discrepancy between the output $O(Z, W)$, where Z is the input and W is the weights used, and the desired output D. Normally a mean square error or cross entropy loss is used as a measure of discrepancy [31]. Mean square error is shown in equation 2.11 while cross entropy loss is described in section 2.3.2.

$$C = \frac{1}{2}(D - O(Z, W))^2 \tag{2.11}$$

Overfitting

A desired trait in machine learning models is its ability to generalise over many datasets. Generalisation in machine learning means that the model has low error on examples it has not seen before [32]. There are two measures which normally is used to indicate how well the model fits the data, bias and variance. The bias is a measure of how much the model differ over all possible datasets, from the desired output. The variance is a measure of how much the model differ between datasets.

When the model first starts training the bias will be high as it is far from the desired output, however the variance will be low as the data has had little influence over the model. Late in the training the bias will be low as the model has learned the underlying function. However if trained too long the model will start to learn the noise from the data, this is refereed to as overfitting. In the case of overfitting the model will have low bias as it fits the data well and high variance as the model follows the data too well and don't generalise over datasets [31]. The F_1-score measures the harmonic mean between the bias and the variance, usually it is preferred to have a good balance between the bias and the variance.

There exists methods to avoid overfitting. Early stopping is one of them and involves stopping the training of the model due to some stopping criterion. The criteria can be human interaction, low change in loss or any other ad-hoc definition [32]. Another method is dropout which only trains a random set of neurons when updating the weights. The idea is that when only a subset of the neuron are updated at the same time, they each learn to recognise different patterns and therefore reduce the overall overfitting of the network [33].

Chapter 3

Method

As the authors had data available and a practical environment to work and test the models in, conducting experiments were deemed suitable as a method to answer the research questions. This chapter describes the data and its design and properties. It also covers how data collection, pre-processing, word representation and experiment evaluation were performed. Section 3.8 explains the aim of each experiment and why it is performed.

3.1 Hardware and software

To run the software a consumer grade desktop computer were used. The computer has 64GB DDR4 non-ECC memory, Nvidia GTX 1080Ti graphics card and an Intel Core i7-7820X 3.6GHz processor. For the development environment Jupyter Notebook were used with a Python3 kernel. TensorFlow [34] v1.3, Scikit-learn [35] v0.19 and Gensim [36] v2.3 were used for defining the classifiers and word vectors. Where applicable the algorithms were accelerated with the Nvidia GPU using CUDA v8.0 and CuDNN v6.0.

3.2 Email dataset

The email dataset used during the experiments consists of 105,195 emails from a real life telecommunication support environment. The emails contains support errands about invoices, technical issues, number management, admin rights etc. The emails are classified with one or more labels. There is in total 33 different labels with varying frequency as shown in image 3.1. The label "DoNotUnderstand" is an artefact from the rule based system where the email did not match any rule, this label is filtered out and is not used during training or testing. The dataset contains 31,700 emails with the label "DoNotUnderstand". This results in a classification rate of 69,9% with the currently implemented manual rule based model. The figure also show a major class imbalance, however no effort were made to balance this since those are the relative frequencies that will be found in the operative environment.

The email labels can be aggregated into queue labels which is an abstraction of the 33 labels into 8 queue labels. The merger is performed by fusing emails from the same email queue, which is a construction used by the telecommunication company, into a single queue label. The labels that are fused together are often closely related to each other, which effectively will reduce the amount of conflicts between the email labels and their contents. If an email contain two or more labels it is disregarded since it might introduce conflicting data which is unwanted when training the classifier. Without "DoNoUnderstand" and the multilabel emails there are a total of 58,934 emails in the dataset.

Each email contains a subject and body which is valuable information for the classifier. The emails may also contain Hypertext Markup Language (HTML) tags and meta data which are artefacts from the infrastructure. The length of each email varies, however the average is 62 characters. Figure 3.2 shows the length distribution where emails under 100 characters is the most common.

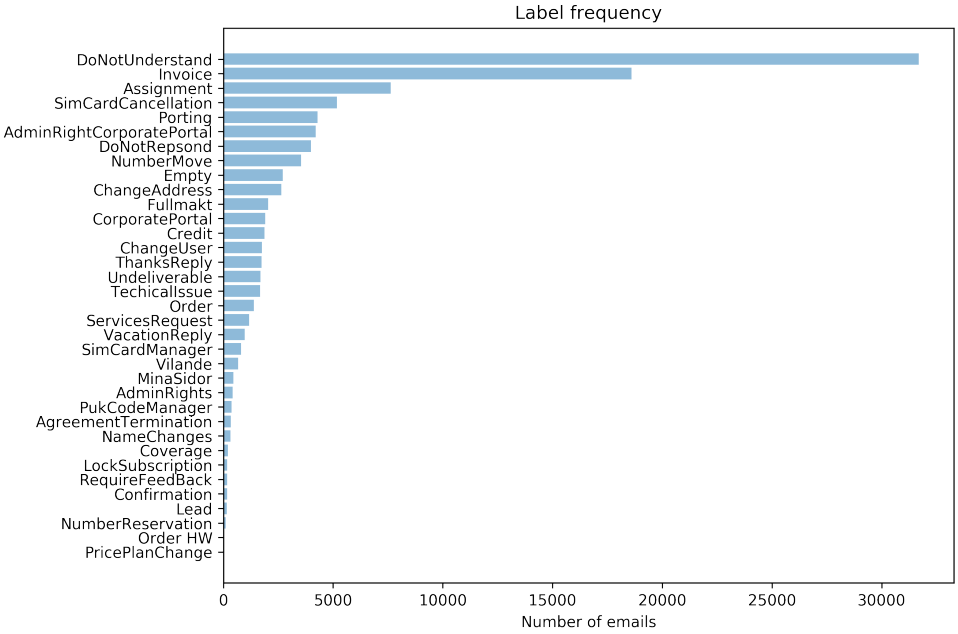

Figure 3.1: Label frequencies.

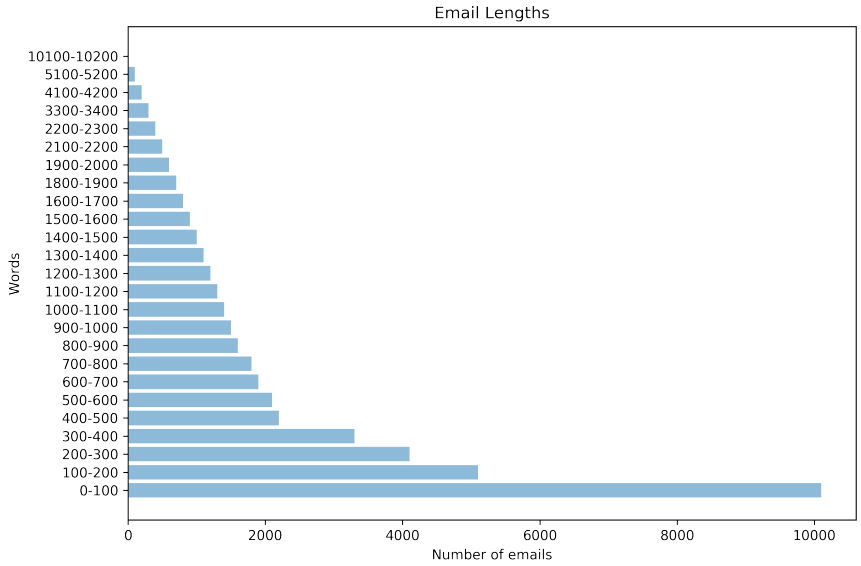

Figure 3.2: Length of each email rounded to nearest 100 characters.

3.3 Data collection for word corpus

When collecting data for word vectors there are several points to consider. Firstly the data needs to be extensive, i.e the more the better as a general rule. To accomplish this the Swedish Wikipedia [37] were used which can be downloaded online[1], the 2000 to 2015 collection of web crawling from Swedish forums and blogs made available by Språkbanken [38] and lastly the emails themselves for increased domain knowledge.

The Wikipedia dataset contains about 380 million words and can be accesses online. It is formatted in HTML and XML which were converted to plain-text JSON before processing it further. The corpus from Språkbanken is bundled with scripts that converts the pages to plain-text. The Språkbanken corpus contains roughly 600 million words. The emails are formatted in HTML which also were converted into plain-text. Only the subject and the body were kept from the email headers. Finally the datasets were merged into one corpus with special characters removed. The end product is a plain-text file with one page per file with a stream of words separated by a single white space.

Secondly the data needs to be representative, i.e the words used in the prediction needs to exist in the corpus as well. The reason for this is simple, when the word vectors are created they are made according to a dictionary. The dictionary is based on the words in the corpus, if the word is not in the corpus there will not be a vector to represent the word which leads to the word being ignored later in the training and prediction stage. For this reason it is a good idea to base the

[1] `https://dumps.wikimedia.org/svwiki/latest/`

corpus on the targeted domain, in our case it is the support emails, and then fill the corpus with data from other sources to make it more extensive.

3.4 Word representation

The models are trained on the largest corpus based on Wikipedia, Språkbanken and emails. This is due to skipgram and GloVe are shown to perform better on a larger corpus and that domain-specific language can improve a NLP model [39, 8]. As a comparison GloVe will be trained on a smaller corpus based solely on the emails.

Skipgram and CBoW word vectors are implemented using the gensim Python package[2]. The GloVe word vectors are generated using the source code[3] published by the GloVe authors [15]. Skipgram-ng word vectors are generated by the framework released by Facebook on Github[4].

All word vector models are trained with the hyperparameters shown in table 3.1. These are the settings that achieved the best results. A systematic tuning of hyperparameters may lead to better performance.

Parameter	Value
Vector size	600
Window size	10
Minimum word occurrences	5
Iterations	10

Table 3.1: Word vector hyperparameters

BoW and BoWBi is implemented using scikit-learn. To reduce the number of features and improve the quality some filtering feature is done by hyperparameters as shown in table 3.2, the rest of the settings were default. The hyperparameters increased the performance compared to the default values, but they were not decided through a systematic hyperparameter tuning which might increase the performance.

BoW consist of 2374 features and BoWBi consist of 7533 when trained on the emails using the hyperparameters in 3.2.

[2]https://pypi.python.org/pypi/gensim
[3]https://nlp.stanford.edu/projects/glove/
[4]https://github.com/facebookresearch/fastText

Parameter	Value
Minimum document frequency	0.001
Maximum document frequency	0.01

Table 3.2: BoW hyperparameters

3.5 Data preprocessing

An email goes through several pre-processing steps before classification which removes meaningless data and increases the overall quality of the information found in the email. Firstly HTML tags and meta data are removed since it does not contribute to the understanding of the email. Then the email subject and body is extracted as described in algorithm 1. Only the latest body is extracted from the email and no previous parts of the conversation is considered. Next the emails are cleaned using algorithm 2 which effectively reduces the email into a single line string.

Algorithm 1: Extract subject and body from email

input : Email e in text format with meta data
output : Email subject and the first body

convert e to lower case;
subject $\leftarrow$ first line of e;
body $\leftarrow$ all lines after the first that does not start with 'from:';

Algorithm 2: Cleanup the email body, used in preprocessing before training classifiers

input : Unprocessed email body, e
output : Email body without undesired characters and extra spaces

undesired $\leftarrow$ array of characters to remove;
convert e to lowercase;

for *char* $\in$ *undesired* **do**
  replace *char* with whitespace in e;
end

remove extra newlines from e;
remove extra tabs, punctuations and commas from e;
replace numbers with the token 'nummer' in e;
remove extra whitespaces from e;

3.6 Evaluation metrics

For classification problems it is common to use a coincidence matrix to determine the performance [40]. The coincidence matrix for a two-class classification is build from four terms, True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). The terms are derived from the predicted class versus the true class as illustrated by table 3.4. Table 3.3 shows how said positives and negatives are defined and used in this paper.

Metric	Definition
TP	Label is present, label is predicted
TN	Label is not present, label is not predicted
FP	Label is not present, label is predicted
FN	Label is present, label is not predicted

The metrics are defined per label

Table 3.3: Positives and negatives definition

There exists several metrics that utilise the coincidence matrix. However there are pitfalls that must be considered when using the metrics. Accuracy is defined as the true predictions divided by the total, shown in equation 3.1. In a multiclass problem, in our case it is email labelling with 33 classes, the average probability that a document belongs to a single class is $\frac{1}{33} \approx 0,0303$, i.e. 3.03%. A dumb algorithm that rejects all documents to belong to any class would have a error rate of 3% and an accuracy of 97% [41]. To gain better insight we also measure the Jaccard index seen in equation 3.2. The Jaccard index disregard the TN and only focus on the TP which makes the results easier to interpret. Equation 3.3, precision, measure how many TP there is among the predicted labels and equation 3.4, recall, measure how many labels that are correctly selected amongst all labels. A classifier that chooses all labels as predicted would have a low precision since it would have many FP but the recall would be high because there would not be any FN. The F_1-score is the harmonic mean between precision and recall [21], a good score is only achieved if there is a balance between the precision and recall. The F_1-score make an implicit assumption that the TN are unimportant in the operative context, which they are in this context.

		True class	
		Positive	Negative
Predicted Class	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Table 3.4: Coincidence matrix

Olson and Delen defines the following metrics for evaluating predictive models [40] as described in equation 3.1, 3.2, 3.3, 3.4, 3.5. These measurements are used to give insights in the classifiers performance on previously unseen emails.

[21]

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

$$JaccardIndex = \frac{TP}{TP + FP + FN} \tag{3.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.4}$$

$$F_1 - score = \frac{2TP}{2TP + FP + FN} \tag{3.5}$$

3.7 Evaluation procedures

The evaluation methods are used draw correct conclusions from the results that are generated. The methods are applied to the metrics described above if it is possible.

3.7.1 Ranking

Ranking is used to obtain a rapid understanding and significance of the result [42]. The rank is assigned in order of magnitude in which a higher rank is a better. Ranking can be performed if there are two or more results that can be compared against each other.

3.7.2 Friedman test

The Friedman test is a statistical significance test that measures a number of algorithms over different measurements and compare them against each other [21]. The test is non-parametric, based on ranking, and does therefore disregard the distribution of the measurements. The average rank of the algorithms measurements is calculated and then the sum of squared differences is compared against a critical value found in a table for the Friedman test. If the sum of squared differences is greater than the critical value it is possible to reject the null hypothesis. The null hypothesis is what the test seeks to answer, i.e if there a significant difference in the performance of the algorithms. Instead of comparing the critical value it is possible to calculate the probability that the null hypothesis is true. If the p-value is below a certain significance level it is possible to reject the null hypothesis. Significance levels that are common and will be used in the experiments are 0.01 and 0.05, which correspond to a probability of 1% and 5%.

3.7.3 Nemenyi test

The Friedman test does only measure if there is a significant difference in the performance of the algorithms that are compared, it does not do any pairwise measurement [21]. The Nemenyi test is a post-hoc test that perform pairwise comparisons based on the average rank of each algorithm to decide which algorithms that perform significantly better than others. In the Neyemni test it is also possible to calculate the p-value, which will be compared against the same significance levels as with the Friedman test. The null hypothesis that the two algorithms perform equal can be rejected with a certainty decided by the significance level if the p-value is less than the significance level.

3.8 Experiment design

Evaluating all the combinations of corpora, NLP models and classifiers is not possible within the relative short time frame available. The experiments are therefore designed such that the best performing models are selected to be used as the go-to model when evaluating the corpus, the NLP model and the classifiers.

Two branches of experiments will be conducted with focus on sequential models and non-sequential models. The experiments on the sequential models depend on three major variables, the dataset, the word vectors and the LSTM hyperparameters. The experiments on the non-sequential models depend on two variables, the document representation and the classifier. Some of the document representation in the non-sequential experiments build on the results of the experiments on the sequential models.

Because it would be too time consuming the sequential model is not validated

with 10-times 10-fold cross validation. Instead we use a test set that consist of a random 10% sample of the data and train the model on the remaining 90%. The sets are randomly chosen from a unifrom distribution without any class balancing. These experiments measure the accuracy, precision, recall, F_1-score and the Jaccard index. The sequential models are not tested using statistical tests because there is too little data to work properly.

The non-sequential models are tested with 10-times 10-fold cross validation. These models will be measured by the F_1-score and the Jaccard index since they measure the performance from different perspectives which both are valuable. Friedman test and the Nemenyi post-hoc test will be performed on both of the measurements to show if there is any significant difference in the results.

3.8.1 Non-sequential classifier experiments

The non-sequential models are tested with 10-times 10-fold cross validation. Friedman test is used to test if there is a significant difference in the performance. If the Friedman test show a significant difference a Nemenyi test is performed to show which algorithms that perform different. The classifiers will be trained on a subset of 10000 emails chosen randomly because of the drastic increase in training time when increasing the number of emails.

3.8.2 Sequential classifier experiments

The experiments on the sequential models will evaluate which combination of corpus, text representation and LSTM hyperparameters that work best. These experiments aim to answer the research questions 1, 4 and 5. The results from these experiments together with the results from the non-sequential experiments will be used to answer research question 2 and 3.

The LSTM network was built using TensorFlow Python module which contains a predefined LSTM cell class. The cells used were "tf.contrib.rnn.LSTMCell" with a orthogonal initializer. For multiclass training the softmax cross entropy loss were used together with a Stochastic Gradient Descent (SGD) optimiser. The hyperparameters used for the LSTM network are described in table 3.5. Limiting the emails to 100 words was a trade-of between batch size and run-time, since each batch consist of a matrix which had to fit in the graphic cards memory of 11GB. The majority of emails were under 100 characters, as seen in fig 3.2. Increasing the word limit above 100 characters did not seem to increase the performance of the classifier but rather increasing the training time significantly. Orthogonal initialisation is a way of reducing the problem with exploding or vanishing gradients which hinder long term dependencies in a neural network [43]. The rest of the settings were set to values which achieved the best results, a systematic hyperparameter tuning may lead to a better result, but further tuning were considered too time consuming by the authors.

Parameter	Value
Word limit (sequence length)	100
Hidden layers	128
Depth layers	2
Batch size	128
Learning rate	0.1
Maximum epochs	200
Dropout	0.5
Forget bias	1.0
Use peepholes	False
Early stopping	True

Table 3.5: LSTM network hyperparameters

3.8.3 Experiment 1, NLP semantic & syntactic analysis

The objective of this experiment is to decide which Word2Vec model that perform best on the corpus which is based on Språkbanken, Wikipedia and emails, by using an analogy dataset. GloVe will also be trained on a smaller corpus based only on the emails for comparison. The analogy test will show which NLP algorithm that can model the Swedish language best.

1920 analogy questions were used to evaluate CBoW, skipgram, skipgram n-gram and GloVe. The dataset include semantic and syntactic questions about capitals-countries, nationalities, opposites, genus, tenses, plural nouns and superlatives. These models are then ranked in order of how well they perform against each other. All word vector models are trained with the same hyperparameters which are listed in table 3.1.

3.8.4 Experiment 2, NLP evaluated in classification task

This experiment will show which of the NLP models that perform best when tested with a LSTM network on labelled emails with 33 classes. Experiment 1 does not test the NLP models in a classification task which is what this experiment will do. The aim of this experiment is to strengthen the knowledge of the NLP models performance upon which a decision is made about which NLP model that will be used in the following experiments.

The NLP models are trained on Språkbanken, Wikipedia, and emails and it is evaluated on a LSTM network using the hyperparameters in table 3.5 except for the hidden layers where 256 were used. The NLP models are trained with the hyperparameters shown in table 3.1.

In this experiment F_1-score, Jaccard index, precision and recall were used as evaluation metrics. The result from this experiment will highlight which NLP

model that perform best given a LSTM network.

3.8.5 Experiment 3, LSTM network size

Selecting the correct network dimension for the LSTM network is not a trivial task. Often some insights can be gained from the loss curves generated by the test and validation data set. The optimisation of the hyperparameters is also hampered by the training time of the LSTM network.

This experiment will determine which network dimensions that perform best within a defined set of sizes. The network sizes will be tested with the NLP model that performed best in experiment 2. The network size that will be used for further evaluations is determined by the size that achieves the best Jaccard index. During this experiment the LSTM network as described in section 3.8.2 will be used, except for the optimiser where Adaptive Moment Estimation (ADAM) were used instead of SGD. ADAM were used as the optimiser because of initial thoughts that ADAM would perform better than SGD as shown by researchers [44]. Later tests, not included in this report, disproved the theory on the email data and SGD was chosen as the optimiser to use. Because of the time frame there was not enough time to rerun the experiments with SGD.

Cells	Layers
128	1
512	1
1024	1
128	2
512	2
1024	2

Table 3.6: Tested LSTM network sizes

3.8.6 Experiment 4, NLP corpus & LSTM classifier

This experiment will show which combination of corpus and classifier that perform best. Two different corpora will be trained with the best performing NLP model from experiment 2. The network size is decided by the result of experiment 3. In this experiment we also perform one reboot once the network triggers early stopping or the maximum epochs. These classifiers will be tested on both the 33 email labels and the aggregated 8 queue labels.

3.8.7 Experiment 5, non-sequential models performance

This experiment will be used as a baseline for comparison against the LSTM network. This experiment will also answer research question 2 and 3 with the use of the experiment results from experiment 4.

The models ADA, ANN, DT, NB, and SVM are trained using Scikit-learn implementations with default settings[5]. For the ANN we use scikit-learns MLP-Classifier with 500 max iterations and an adaptive learning rate, the rest of the settings are kept at default values. The SVM model is based on the LinearSVC classifier. DT is based on an optimised version of CART. The ADA classifier is using the scikit-learns DT classifier as its weak classifiers. Finally NB is based on the Gaussian distribution.

3.8.8 Experiment 6, Training time

ADA, ANN, DT, NB, SVM and LSTM will be trained on 10000 emails until they converge. The non-sequential classifiers are trained using the AvgWV features. The wall time, the time from start to finish, will be measured and compared. The models will be trained on a computer with the hardware found in section 3.1.

[5]The default values are described in the documentation `http://scikit-learn.org/stable/modules/classes.html`

Chapter 4

Result and analysis

In this chapter we present the results of the experiments described in chapter 3. The performance metrics is presented together with analysis and statistical tests to verify significance where applicable. The performance of the word vectors, non-sequential algorithms and the sequential model with both labels and queues are presented.

4.1 Experiment 1, NLP semantic & syntactic analysis

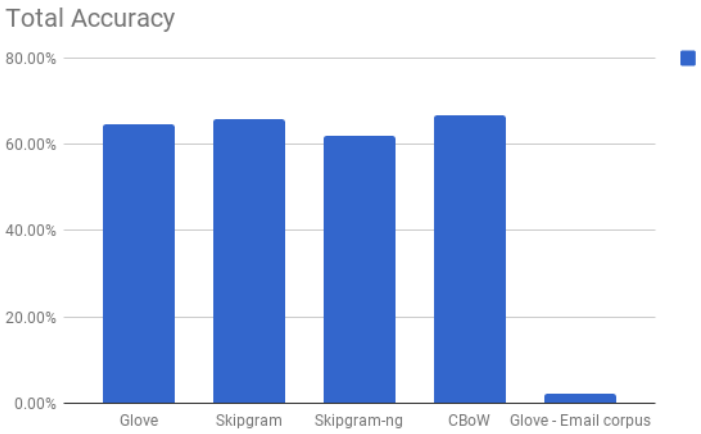

Figure 4.1: Word vector total semantic and syntactic accuracy.

Figure 4.1 and 4.2 show the total and per category accuracy of the semantic and syntactic questions. The different models performed similar however CBoW achieved the highest total accuracy of 66.7%. Skipgram-ng achieved the lowest total accuracy but with the added benefit of being able to construct vectors for words not in the original dictionary. GloVe trained on the smaller corpus solely based on the emails achieve a total accuracy of 2.1%, which is 64.6% points less than the best model.

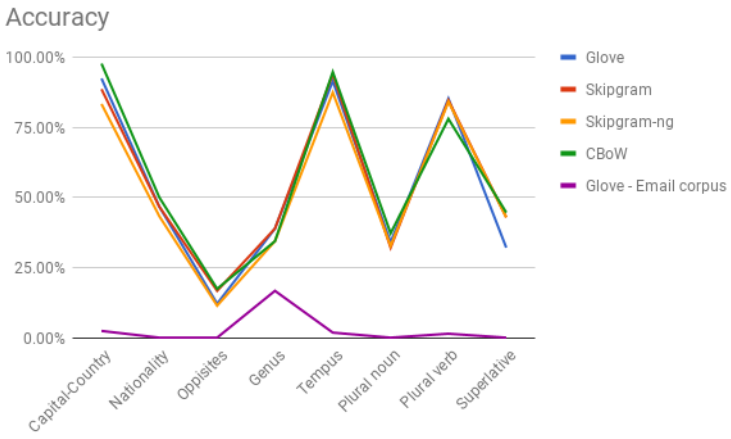

Figure 4.2: Word vector semantic and syntactic accuracy per category.

All models struggled with "opposites" which is a semantic question. However all models also excelled at "capital country" which also is a semantic question.

4.2 Experiment 2, NLP evaluated in classification task

In this section the result of four different NLP word vector algorithms are presented in which they were evaluated on a classification task. Together with the results presented in section 4.1 these results will answer research questions 1 found in section 1.3.

Algorithm	Accuracy	Jaccard	Recall	Precision	F_1-score
Skipgram	0.99	0.82	0.90	0.91	0.90
CBoW	0.99	0.82	0.90	0.91	0.90
GloVe	0.99	0.83	0.91	0.91	0.91
Skipgram n-gram	0.99	0.82	0.90	0.91	0.90

Table 4.1: Performance metrics for each word vector algorithm used in LSTM classification model

The results from table 4.1 show that the word vectors generated by GloVe perform best. The results are very similar and drawing any conclusions is therefore difficult. In the following experiments the word vectors trained by GloVe will be used.

4.3 Experiment 3, LSTM network size

This experiment was supposed to answer research question 5, the experiment were performed to decide the network size and depth, but it did not show any interesting results. The network with 1024 cells and 2 layers performed about 1% better than the smallest network size. The network size that is used in the following experiments is therefore decided by the trade-off between the training speed and the performance. The training time were several factors longer for the bigger network than the smallest, which was unfeasible for this projects time frame. Further experiments will be based on 128 cells in two layers based on its performance and run time.

4.4 Experiment 4, NLP corpus & LSTM classifier

4.4.1 LSTM classification with 8 queue labels

Corpus	Accuracy	Jaccard	Recall	Precision	F_1-score
Språkbanken, wiki, emails	0.98	0.87	0.93	0.93	0.93
emails	0.97	0.81	0.90	0.90	0.90

Table 4.2: Comparing the same LSTM network trained on different corpora and 8 queue labels

The results from table 4.2 show that the full corpus of Språkbanken, Wikipedia and emails perform better then the corpus only based on the emails. The Jaccard index is 6% points better and F_1-score 3% points better when LSTM is trained on the big corpus. However it is interesting that LSTM can achieve good performance with word vectors based on a small corpus even though it scored terrible in the semantic and syntactic analysis as seen in figure 4.1 and 4.2.

4.4.2 LSTM classification with 33 email labels

Corpus	Accuracy	Jaccard	Recall	Precision	F_1-score
Språkbanken, wiki, emails	0.99	0.83	0.91	0.91	0.91
emails	0.99	0.77	0.87	0.88	0.87

Table 4.3: Comparing the LSTM network trained on different corpora and 33 email labels

Table 4.3 show the results when LSTM is trained on two different GloVe word vectors on different corpora. Training LSTM on the big corpus increases the Jaccard index by 6% points and F_1-score with 4% points. The relative performance is about the same as the results from section 4.4.1 trained on queues. The decrease in F_1-score may suggest that the corpus based on the emails may struggle when the number of classes grows.

The results from this experiment and the experiment in section 4.4.1 answer research question 4 regarding the corpus effect.

4.5 Experiment 5, non-sequential models performance

4.5.1 Non-sequential classification with 8 queue labels

Algorithm	BoW	BoWBi	AvgWV
ADA	0.577	0.737	0.588
ANN	0.577	0.766	0.866
DT	0.579	0.733	0.594
NB	0.407	0.622	0.426
SVM	**0.624**	**0.784**	**0.872**
Ranking	1	**2.6**	2.4

Friedman test: p-value=0.022, statistic=7.600

Table 4.4: Jaccard index on queue labels with non-sequential algorithms and ranking of pre-processing performance

Algorithm	BoW	BoWBi	AvgWV
ADA	0.562	0.730	0.480
ANN	0.535	0.702	0.818
DT	0.570	0.730	0.485
NB	0.345	0.498	0.420
SVM	**0.599**	**0.759**	**0.831**

Friedman test: p-value=0.165, statistic=3.600

Table 4.5: F_1-score on queue labels with Non-sequential algorithms

Table 4.4 and table 4.5 show the different pre-processing algorithms performance. From table 4.4 the average rank suggest that BoWBi perform best when compared

to BoW and AvgWV. Even though BoWBi seem to perform better on average there are two outliers in which AvgWV perform about 10 percentage points better which also is the best result obtained.

Friedman test confirm that there is a significant difference in the performance when measuring the Jaccard index at an significance level of 0.05, $X^2(2) = 7.600$, p-value = 0.022. But the test does not confirm a significant difference at significance level 0.05 for the F_1-score measurements, $X^2(2) = 3.600$, p-value = 0.166.

Algorithm	BoW	BoWBi	AvgWV
BoW		0.031	0.069
BoWBi	*		0.946
AvgWV			

**: Significant at $p < 0.05$*
***: Significant at $p < 0.01$*

Table 4.6: Nemenyi post-hoc test on Jaccard index based on table 4.4

A Nemenyi post-hoc test evaluates the difference between the pre-processing algorithms on Jaccard index. The results from table 4.6 show a significant difference between BoW and BoWBi. Even though AvgWV gain the best result it is not a significant difference because it perform worse on the rest of the classifiers.

Tables 4.4 and 4.5 are transformed, by swapping rows and columns, to investigate if there is a significant difference in the classification algorithms performance. The transformation is done because of the Friedman test, which measure the difference on a column basis. Transforming the tables allows statistical tests on classification performance instead of pre-processing performance.

Algorithm	ADA	ANN	DT	NB	SVM
BoW	0.577	0.577	0.579	0.407	0.624
BoWBi	0.737	0.766	0.732	0.622	0.784
AvgWV	0.588	0.866	0.593	0.426	0.872
Ranking	2.333	3.667	3.000	1.000	**5.000**

Friedman test: p-value=0.031, statistic=10.667

Table 4.7: Jaccard index with regards to non-sequential algorithms on queue labels and ranking of classifier performance

Algorithm	ADA	ANN	DT	NB	SVM
BoW	0.561	0.535	0.570	0.345	0.599
BoWBi	0.730	0.702	0.730	0.498	0.759
AvgWV	0.480	0.818	0.485	0.421	0.831
Ranking	2.667	2.667	3.667	1.000	**5.000**

Friedman test: p-value=0.037, statistic=10.237

Table 4.8: F_1-score with regards to non-sequential algorithms on queue labels and ranking of classifier performance

In table 4.7 and table 4.8 the results are presented where the focus is on the classification algorithm and how they perform against each other. ANN and SVM obtain the best result amongst all when trained on AvgWV. The average rank from table 4.7 and table 4.8 show that SVM perform best in all cases and that NB perform worst in all cases. ADA, ANN and DT seem to perform equal except for the good result obtained by ANN when trained on AvgWV. Friedman test on the Jaccard index results in table 4.7, $X^2(2) = 10.667$, p-value = 0.031, does reject the null hypothesis that all classifiers perform equal at an significance level of 0.05. The F_1-score result show the same pattern, $X^2(2) = 10.237$, p-value = 0.037 which reject the null hypothesis at significance level 0.05.

Algorithm	ADA	ANN	DT	NB	SVM
ADA		0.840	0.986	0.840	0.235
ANN			0.986	0.235	0.840
DT				0.530	0.530
NB					0.017
SVM				*	

**: Significant at $p < 0.05$*
***: Significant at $p < 0.01$*

Table 4.9: Nemenyi post-hoc test on non-sequential algorithms Jaccard index based on table 4.7

Algorithm	ADA	ANN	DT	NB	SVM
ADA		1.000	0.938	0.697	0.369
ANN			0.938	0.697	0.369
DT				0.235	0.840
NB					0.017
SVM				*	

**: Significant at $p < 0.05$*
***: Significant at $p < 0.01$*

Table 4.10: Nemenyi post-hoc test on non-sequential algorithms F_1-score based on table 4.8

Table 4.9 and table 4.10 show that the only significant performance difference is between SVM and NB. From the ranking in tables 4.7 and 4.8 SVM perform best in all cases.

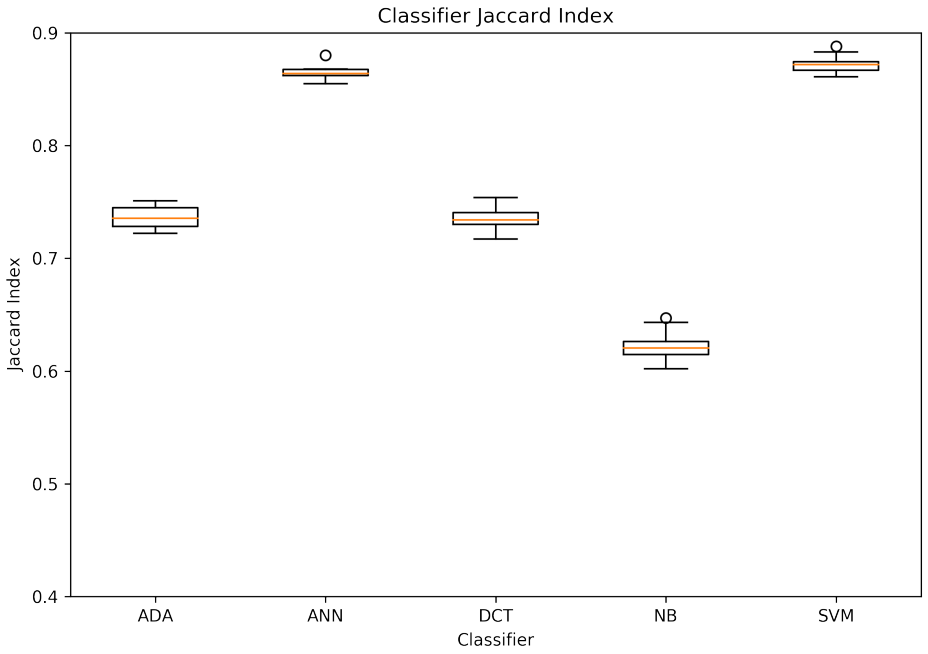

Figure 4.3: The plot show the median in red, first quantities and min/max values with possible outliers shown as circles

The box plot in figure 4.3 show the classification performance over 10 folds using the combination of pre-processing algorithm and classification algorithm that performed best together. The variance is low for all algorithms which is

a good indication that the model does not overfit and can generalise well for previously unseen emails.

4.5.2 Non-sequential classification with 33 email labels

Research question 2 and 3 are answered using the results in this section and the sections 4.4.1, 4.4.2, 4.5.1. The questions compare the sequential LSTM network against the non-sequential classifiers and how the aggregation of the 33 labels into queues affect the classification performance.

Algorithm	BoW	BoWBi	AvgWV
ADA	0.483	0.691	0.469
ANN	0.479	0.689	0.802
DT	0.481	0.677	0.468
NB	0.308	0.488	0.334
SVM	0.524	0.718	0.816

Friedman test: p-value=0.165, statistic=3.600

Table 4.11: Jaccard index on email labels with non-sequential algorithms

Algorithm	BoW	BoWBi	AvgWV
ADA	0.366	0.570	0.225
ANN	0.383	0.550	0.571
DT	0.387	0.559	0.218
NB	0.168	0.254	0.257
SVM	0.423	0.594	0.597

Friedman test: p-value=0.247, statistic=2.800

Table 4.12: F_1-score on email labels with non-sequential algorithms

Table 4.11 and table 4.12 show the results when the pre-processing algorithms are tried on the email labels. The Friedman tests on Jaccard index, $X^2(2) = 3.600$, p-value = 0.165, and F_1-score, $X^2(2) = 2.800$, p-value = 0.247, does not show a significant difference with an significance level of 0.05. SVM and ANN does perform about 10 percentage points better when trained on AvgWV compared to the other pre-processing algorithms and other classification algorithms.

The performance does decrease when compared with the results, based on queues, found in tables 4.7 and 4.5 which is expected due to the increased difficulty of more classes. The result may decreases because some of the classes may be closely related to each other. Closely related labels may be hard to separate which

could explain the drop in performance of email labels compared to the queue labels.

Algorithm	ADA	ANN	DT	NB	SVM
BoW	0.483	0.479	0.481	0.308	0.524
BoWBi	0.691	0.689	0.677	0.488	0.718
AvgWV	0.469	0.802	0.468	0.334	0.816
Ranking	3.667	3.000	2.333	1.000	**5.000**

Friedman test: p-value=0.031, statistic=10.667

Table 4.13: Jaccard index with regards to non-sequential algorithms on email labels and ranking of classifier performance

Algorithm	ADA	ANN	DT	NB	SVM
BoW	0.366	0.383	0.387	0.168	0.423
BoWBi	0.570	0.550	0.559	0.254	0.594
AvgWV	0.225	0.571	0.218	0.257	0.597

Friedman test: p-value=0.126, statistic=7.200

Table 4.14: F_1-score with regards to non-sequential algorithms on email labels

Table 4.13 and table 4.14 is the transformed version of table 4.11 & 4.12. Friedman test with focus on the classification algorithm does show a significant difference on Jaccard index, $X^2(2) = 10.667$, p-value = 0.031, with an significance level of 0.05 but not on the F_1-score, $X^2(2) = 7.200$, p-value = 0.126. From the ranks in table 4.13 we can see that SVM is performing best using all pre-processing algorithms whereas NB is performing worst in all cases.

Algorithm	ADA	ANN	DT	NB	SVM
ADA		0.986	0.840	0.235	0.840
ANN			0.986	0.530	0.530
DT				0.840	0.235
NB					0.017
SVM				*	

**: Significant at $p < 0.05$*
***: Significant at $p < 0.01$*

Table 4.15: Nemenyi post-hoc test on non-sequential algorithms performance with email labels

One significant difference were found between SVM and NB as seen in table 4.15 at an significance level of 0.05. There is however differences between the other algorithms even though they are not considered to be significant enough.

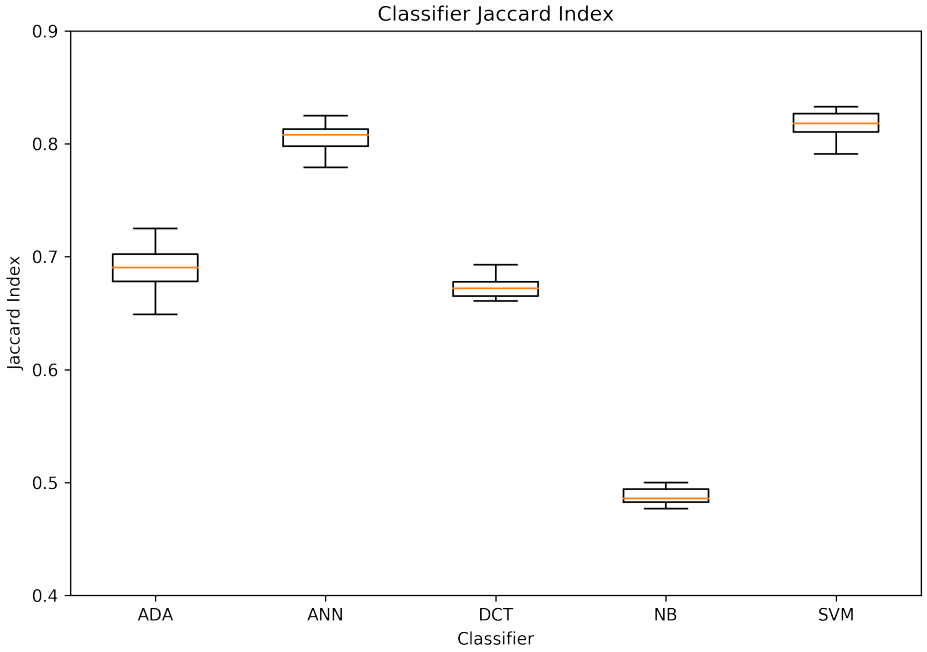

Figure 4.4: The plot show the median in red, first quantities and min/max values with possible outliers shown as circles

Figure 4.4 show visually, though a box plot, how the performance differentiate between the classifiers. The plot is drawn from the text representation that yield the maximum accuracy per classifier. SVM has the highest average accuracy with low variance and low difference between the lowest and highest values.

4.6 Experiment 6, Training time

Algorithm	Time (s)
LSTM	20068.00
SVM	39.13
ANN	9.37
DT	6.63
ADA	6.26
NB	0.03

Table 4.16: Execution time in seconds when trained on 10000 emails

Table 4.16 present the execution time of the algorithms trained on 10000 emails. The wall time is measured from the start to the finish of the training. There is a big difference in the training time where NB is the fastest to train with less than one second. SVM is the slowest of the non-sequential algorithms with a training time of 39 second. LSTM does train in several epochs in which it trains on the same samples several times to adjust its weights, the process is time consuming which is shown by the execution time. The training time of the LSTM network is strongly correlated with how many epochs the network needs before convergence. In this measurement the LSTM network needed 94 epochs to converge.

4.7 LSTM certainty values

Figure 4.5 shows the certainty values for each label by the proposed LSTM model. The data is collected by classifying all instances in the test dataset, which contains 5893 emails unseen during training. When predicting a label each instance also get a certainty value of said label. The average certainty is shown by the yellow line for each label. The circles indicate outliers, i.e. points outside the first and third quartiles.

The ideal box is positioned near 1.0 with low height as this indicate a high average certainty and low variance between instances. In the plot this is shown by the label “numbermove” while the label “servicerequest” shows less promising results and may need further improvements.

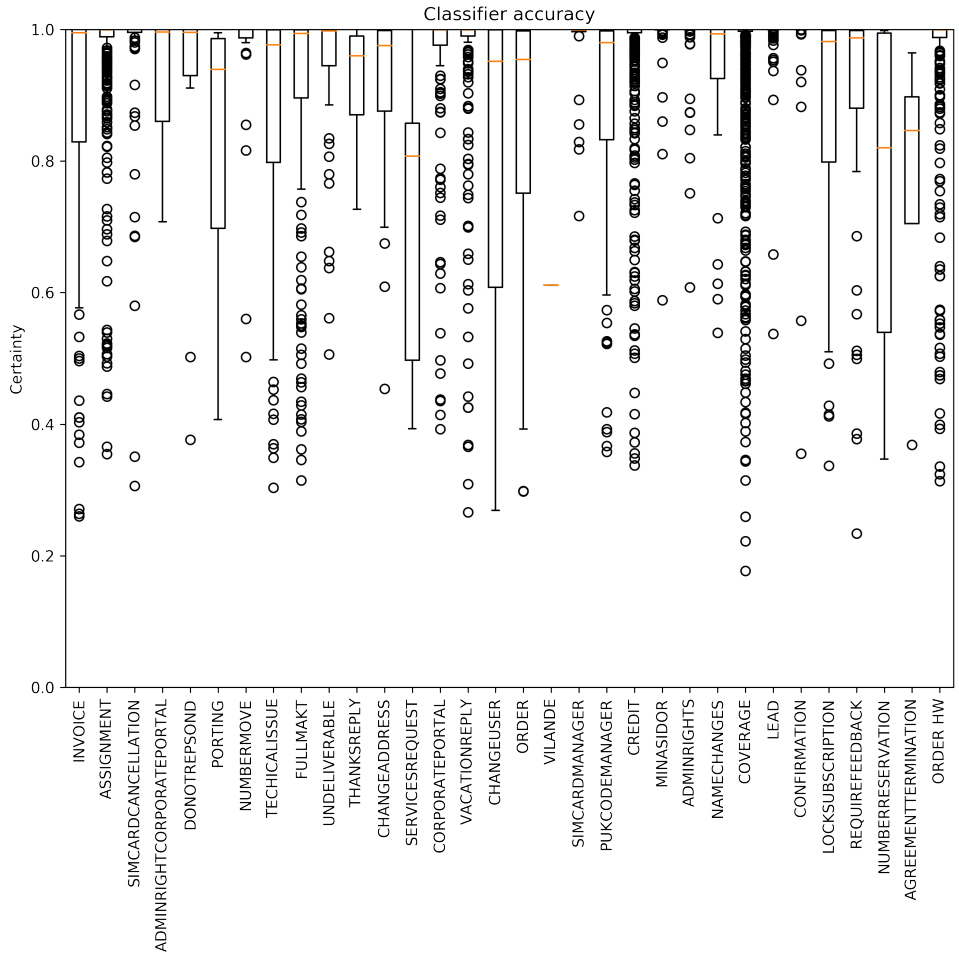

Figure 4.5: Certainty values per label using the proposed LSTM model based on the test dataset, illustrated with a box plot.

Chapter 5

Discussion

During training of the LSTM model a consistent bump in performance were observed if the training were restarted after early stopping had triggered. An increase of approximately two to three percentage points in F_1-score were common. Warm restart of model training and resetting the learning rate has been shown to increase the convergence when using SGD [45]. The authors decided to include the warm restart method as it gave a consistent increase in both training and validation performance. However the reboot was only used once and not continuously during training as suggested by Loshchilov and Hutter. However this effect is not extensively covered by other research and may be observed due to other unknown effects of the training.

The LSTM model achieved high performance when predicting the full set of 33 labels. The number of labels are relatively low compared to other machine learning networks[1], however more classes increase the difficulty of the prediction task. With this reasoning the model should achieve a better performance when predicting queues instead of labels. Due to both the reduced number of labels, eight instead of 33, and the increased divergence between the labels, an increase in performance were expected. As shown in the results there were indeed an increase in performance but not as large as hoped. The downside of aggregating the labels are the reduced flexibility and granularity as the system now has less freedom when sorting emails into queues based on their labels. There could exist labels that are more related to other queues than the queue they are placed in. The queues are not constructed to optimise classification but rather to group labels that the teams with special training can handle efficient.

The dataset used during training of the LSTM model lacked conviction in terms of label accuracy. The labels were set by the rule based model and not fully confirmed by a human expert. This lead to some inconsistencies in the dataset which may have affected the performance of the classifiers. The dataset should be expanded to make sure there is enough examples for each label and all email labels should be verified to make sure that they are correct to avoid noise in the data.

When evaluating the word vectors a set of Swedish, semantic and syntactic

[1] Google's Inception network [46] consists of 1000 classes.

questions were used. These questions were defined by the authors and considered extensive enough. However as the authors are not linguistic experts there may have been both discrepancies and faults in the dataset. Verifying the integrity of the dataset and also expand the set with more questions is important if it should be used to evaluate the word vectors performance. Evaluation the word vectors using QVEC, as proposed by Tsvetkov et al., may be a better evaluation method and lead to a better understanding of the word vectors performance in a classification task [47].

In order to determine which word vector model to use, each model were evaluated using a set of Swedish semantic and syntactic questions. The models performed approximately the same with the exception of GloVe which performed overall about one percentage point better than the other models. However when the word vectors were used in training of the LSTM model they had little or no difference in performance. The cause may be due to the LSTM network being able to learn the same patterns in the dataset even with differences between the word vectors. When choosing the best word vector model for a classifier it is therefore important to evaluate them in a classification task, since the performance of the semantic and syntactic questions did not correlate with the performance of the word vectors in a classification task. The semantic and syntactic analysis show how well the word vectors model the language in general which may not be relevant for domain specific classification. The LSTM network is shown to be able to adapt to word vectors that do not achieve good semantic and syntactic results. It is possible that the word vectors based on the emails does model the domain language which may be what the LSTM network utilise. Incorporating domain language in a corpus is therefor recommended because it may add valuable relations between words that have a different semantic and syntactic meaning in the domain.

As extensive computation is used to solve problems that are unsolvable by humans we have to take into account the efficiency of the algorithm that is used. The energy usage can differ severely between different algorithms depending on several factors. The execution times in section 4.6 show that the LSTM network has about 513 times longer execution than SVM which is the slowest of the non-sequential algorithms. LSTM does execute both on the CPU and the GPU which neither of the non-sequential algorithms do. Improving the LSTM hyperparameters may lead to a reduced execution time. Techniques as warm reboot could also increase the convergence rate [45]. If energy consumption is a concern and the extra performance increase given by LSTM is redundant it is recommended to use ANN with AvgWV.

The different classifiers are well suited for NLP tasks. LSTM does perform better than the other classifiers, but it does require more data. If NLP tasks are to be solved in other domains that do not generate enough data for a LSTM to work properly it would be advisable to train a SVM using AvgWV. LSTM is more adaptable but knowing how to optimise the network does require domain

knowledge and experience with gradient-decent classifiers.

A machine learning based classifier could help a company to reduce work hours that are spent on email support. The classifier could be trained to forward incoming emails to personnel or groups that handle different types of errands. If the company does use a manually created rule-based classifier, which the company supporting this study does, it would be possible to replace it with a machine learning based model which would reduce a substantial amount of work hours spent on tuning the rules. The machine learning model is also more consistent and less prone to failure due to human error. The framework developed and described in chapter 6 is adaptable to support further features such as semantic analysis which would add additional business value. The framework can replace or co-exist with the current the rule-based system in the company without big infrastructure changes. The rule-based system requires tuning in which rules are adapted to support new templates, campaigns or temporary changes in the label structure. A machine learning model does also require tuning but it is much harder to control the outcome since the classification is somewhat of a black box. It is possible to adapt the labels of the training data to achieve these goals, but it might be difficult to gain full control of the classifier. Creating good quality data to train the classifier is therefore crucial for improvement of the framework. A solution would be to integrate data generation in the system in which new labelled data could be produced by the support team. Temporary labels such as campaigns where good labelled data is hard to generate could be handled by a rule-based classifier incorporated in the framework, but keeping the rules minimal and maintainable is crucial.

The classification rate of the current rule based system is approximately 70% as described in section 3.2. One of the objectives for this study were to improve the classification rate. The proposed LSTM model does always produce a class for a given email which can be interpreted as a classification rate of 100%. However if the proposed model does not understand the email it will still assign the email a label but with low certainty. The certainty of a classification can be used to determine if the model understands the email or not, however at what level of certainty this can be determined is not obvious. As shown in image 4.5 the average certainty values of most of the labels are quite high. If 80% certainty is considered as the model "understands" the email, 92% of the emails are above said threshold. However further research has to be done in order to determine if this is a realistic threshold.

Instead of using queues for each category of support errand that the employees grab emails from, the network may assign an email to an employee directly. It might be possible to extend the network with one or more neural networks that specialises in learning which employees that prefer which emails. This can further improve the practical usefulness by reducing the response time further. The network may be able to learn the preferences of each employee directly and assign them emails based on current load, expertise, satisfaction rate etc.

Classifying emails wrong may affect the customer who sends the email. If a company specialise their support personnel they might receive emails that they are not trained to answer. In those cases it is important to have a strict policy that require all personnel to forward the email to another colleague that can handle the errand better. Wrongly classifying emails that contain sensitive information could lead to information disclosure if personnel that do not have the correct security clearance receive the information.

The following list aims to explicitly answer our research questions previously defined in section 1.3.

1. **To which degree does the NLP model (e.g word2vec) affect the classifiers classification performance?**
   Result: *The NLP model used does not significantly affect the classification performance as LSTM seem to compensate for the difference between them.*

2. **How well can LSTM, compared to non-sequential machine learning models, classify emails?**
   Result: *LSTM perform about 1.5 percentage points better on Jaccard index and 31 percentage points better on F_1-score compared on the 33 labels.*

3. **Does an aggregation of the labels increase the performance metric compared to having all labels separated?**
   Result: *Yes, aggregating the labels does increase the performance.*

4. **To which degree does the corpora affect the LSTM performance?**
   Result: *Training word vectors on a corpus that model the language better increase the classification performance compared to training word vectors on a corpus that model the language badly.*

5. **To which degree does the LSTM network size and depth affect the classifier performance?**
   Result: *The network size and depth does not affect the classification performance significantly if a suitable size and depth is used.*

Chapter 6

Implementation of models

In order to test our model in a live environment, a Hypertext Transfer Protocol (HTTP) API was build using Flask. The API takes POST HTTP requests and return the most probable label. The API has the ability to classify an email using either LSTM or SVM. This allows voting between the models if preferred, however as LSTM has shown the highest classification performance this model is recommended and used by default. LSTM does also allow for a certainty value to be returned which SVM does not. The certainty value can be used as an indicator about how certain the model is that the predicted label is the correct label. A lower value might mean that there is content in the email that could be relato another label, meanwhile a high value means that the content of the email most probably is correctly labelled.

The server takes JavaScript Object Notation (JSON) formatted POST data as input. The parameters are "model" and "email", where the model is optional and defaults to "LSTM" while the email is required and contains the contents of the email in text format. All HTML tags must be stripped from the email prior to classification. The API does only allow one email per request but can be extended to also allow email classification in batches. It has also been configured to only allow requests over the HTTPS protocol for increased security. As future work the security can be extended by require a valid API key which also can be used to limit requests per user. Algorithm 3 describes the API process of predicting email

labels.

Algorithm 3: API request handling

input : Email e to be classified and optional preferred model m
output : Class and its certainty if applicable

$wv_d \leftarrow$ load dictionary;
$wv_v \leftarrow$ load vectors;
$lstm_le \leftarrow$ load lstm multilabel binarizer;
$svm_le \leftarrow$ load svm multilabel binarizer;
$svm \leftarrow$ load svm model;
$dropout = 1$; `// Disable dropout for predictions`
$forget_bias = 0$; `// Disable forget bias for predictions`
restore lstm model by loading the latest TensorFlow checkpoint;
$model \leftarrow$ get model from request else LSTM ; `// Default model is LSTM`
$email \leftarrow$ get email from request;
clean the email from unknown characters;
lookup the indices using wv_d of each word in the email;
lookup the vectors using wv_v of each index in the email;
if $model \equiv lstm$ **then**
  predict the class using the $lstm$ model and $lstm_le$;
else if $model \equiv svm$ **then**
  average the email to one single vector;
  predict the class using the svm model and svm_le;
else
  return error, invalid model;
end

An example request could be made as following

```
curl
    -H "Content-Type: application/json" -X POST
    -d '{ "model": "lstm",
          "email": "hej, jag har problem med min faktura" }'
    https://api.fqdn.com/api/classify
```

which will result in a response with the format

```
{
  "class": "Invoice",
  "prob": "0.9932"
}
```

The server classifies approximately 20 emails per second with LSTMs. The LSTM model used by the API was trained using the hyperparameters described in table 6.1. The model is trained on the full dataset of $\approx$ 60000 support emails from a live telecom environment. The hardware and software packages are the same as described in section 3.1.

Parameter	Value
Word vectors	GloVe
Corpus	Full (Språkbanken, Wikipedia, emails)
Word limit (sequence length)	100
Hidden layers	128
Depth layers	2
Batch size	128
Learning rate	0.1
Maximum epochs	200
Dropout	0.5
Forget bias	1.0
Use peepholes	False
Early stopping	True

Table 6.1: LSTM hyperparameters used by the API

Chapter 7

Conclusion

Of the six different classifiers that are evaluated LSTM perform best on both queues and the 33 labels. The LSTM network achieve almost as good results when using the 33 labels as when using the queue labels. Aggregating the labels does therefore increase the performance, but only nominal. Of the non-sequential classifiers, ANN and SVM, achieved best results on both the queue and the 33 labels when trained on AvgWV. The use of AvgWV improved the performance substantially compared to BoW and BoWBi if used with a suitable classifier. The training time of LSTM is several factors longer than that of the non-sequential models, if power consumption and training time is important, select a non-sequential modes such as SVM with AvgWV.

A framework was implemented based on the results of the experiments. The framework is intended to generate business value for a company by reducing the work hours spent on tuning rule-based systems. Changing to a machine learning based framework does also allows for faster and easier development for features such as sentiment analysis which will add further business value to a company. LSTM is chosen as the main classifier because of its classification performance and the features is supports such as the possible to receive a probability value indicating the certainty of the prediction. The probability can be of much use for a data analyst when improving the model by knowing its strengths and weaknesses.

Chapter 8

Future work

Extending the classification to identify emotions in the email can help the support team deal with angry or dissatisfied customers [5]. Doing so will improve the customer service since the support personnel can cope with the emotions of the customer. This will increase the customer satisfaction and decrease the number of customers that change provider.

Given that the model only classifies the latest response in an email conversation but often keeps the subject of the original email there may be conflicts that causes confusion for the LSTM network. There may be a performance increase by separating the subject from the body and use two LSTM networks to classify each part separate. The two networks may then be interlaced by a fully connected neural network.

Currently the emails are processed before entering the classifier. In the early stages of preprocessing all other bodies than the first is stripped. The other bodies contains previous conversations and may be helpful during classification. However the effect of stripping other bodies versus including two or more is unknown and future work may compare the effect of including several bodies during classification.

Currently the network is trained once and does not change its predictions in production even if they were to be wrong. If the network are to improve over time it has to be periodically retrained. This procedure is both time and computationally costly. It also introduces a delay between the correction and the actual adapting of the model. One way to reduce this time and allow the network to adapt continuously to changes in the email environment is reinforcement learning. Future work may look closer at the benefits and usefulness of reinforcement learning.

Chapter 9

References

[1] G. G. Chowdhury, "Natural language processing," *Annual Review of Information Science and Technology*, vol. 37, no. 1, pp. 51–89, 2003.

[2] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[4] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *International Conference on Machine Learning*, 2015, pp. 957–966.

[5] R. Bougie, R. Pieters, and M. Zeelenberg, "Angry customers don't come back, they get back: The experience and behavioral implications of anger and dissatisfaction in services," *Journal of the Academy of Marketing Science*, vol. 31, no. 4, pp. 377–393, 2003.

[6] M. L. Richins, "Negative word-of-mouth by dissatisfied consumers: A pilot study," *Journal of Marketing*, vol. 47, no. 1, pp. 68–78, 1983.

[7] K. Coussement and D. V. den Poel, "Improving customer complaint management by automatic email classification using linguistic style features as predictors," *Decision Support Systems*, vol. 44, no. 4, pp. 870 – 882, 2008.

[8] A. R. Coden, S. V. Pakhomov, R. K. Ando, P. H. Duffy, and C. G. Chute, "Domain-specific language models and lexicons for tagging," *Journal of biomedical informatics*, vol. 38, no. 6, pp. 422–430, 2005.

[9] P. Fallgren, J. Segeblad, and M. Kuhlmann, "Towards a standard dataset of swedish word vectors," in *Sixth Swedish Language Technology Conference (SLTC)*, 2016.

[10] J. Nowak, A. Taspinar, and R. Scherer, *LSTM Recurrent Neural Networks for Short Text and Sentiment Classification.* Cham: Springer International Publishing, 2017, pp. 553–562.

[11] Y. Yan, Y. Wang, W.-C. Gao, B.-W. Zhang, C. Yang, and X.-C. Yin, "Lstm$\hat{\{2\}}$: Multi-label ranking for document classification," *Neural Processing Letters*, May 2017.

[12] E. Gabrilovich and S. Markovitch, "Text categorization with many redundant features: using aggressive feature selection to make svms competitive with c4. 5," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 41.

[13] H. Bayer and M. Nebel, "Evaluating algorithms according to their energy consumption," *Mathematical Theory and Computational Practice*, p. 48, 2009.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, 2013, pp. 3111–3119.

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

[16] D. M. Christopher, R. Prabhakar, and S. Hinrich, "Introduction to information retrieval," *An Introduction To Information Retrieval*, vol. 151, p. 177, 2008.

[17] N. S. Baron, "Letters by phone or speech by other means: the linguistics of email," *Language & Communication*, vol. 18, no. 2, pp. 133 – 170, 1998.

[18] C. McCormick, "Word2vec tutorial - the skip-gram model," 2017. [Online]. Available: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

[19] C. De Boom, S. Van Canneyt, T. Demeester, and B. Dhoedt, "Representation learning for very short texts using weighted word embedding aggregation," *Pattern Recognition Letters*, vol. 80, pp. 150–156, 2016.

[20] F. Sebastiani, "Machine learning in automated text categorization," *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp. 1–47, 2002.

[21] P. Flach, *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.

[22] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep 1995.

[23] P. Domingos and M. Pazzani, "Beyond independence: Conditions for the optimality of the simple bayesian classifer," in *Proc. 13th Intl. Conf. Machine Learning*, 1996, pp. 105–112.

[24] H. Zhang, "The optimality of naive bayes," *Association for the Advancement of Artificial Intelligence*, vol. 1, no. 2, p. 3, 2004.

[25] D. D. Lewis, "Naive (bayes) at forty: The independence assumption in information retrieval," in *European conference on machine learning*. Springer, 1998, pp. 4–15.

[26] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *European conference on computational learning theory*. Springer, 1995, pp. 23–37.

[27] A. Graves, A. r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 6645–6649.

[28] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602 – 610, 2005, iJCNN 2005.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[31] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade: Second Edition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 421–436.

[32] L. Prechelt, "Automatic early stopping using cross validation: quantifying the criteria," *Neural Networks*, vol. 11, no. 4, pp. 761 – 767, 1998.

[33] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[34] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[35] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[36] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[37] Meta, "Data dumps — meta, discussion about wikimedia projects," 2017, [Online; accessed 18-December-2017]. [Online]. Available: https://meta.wikimedia.org/w/index.php?title=Data_dumps&oldid=17422082

[38] S. R. Eide, N. Tahmasebi, and L. Borin, "The swedish culturomics gigaword corpus: A one billion word swedish reference dataset for nlp," pp. 8–12, 2016.

[39] O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 211–225, 2015.

[40] D. L. Olson and D. Delen, *Advanced data mining techniques*. Springer Science & Business Media, 2008.

[41] Y. Yang, "An evaluation of statistical approaches to text categorization," *Information Retrieval*, vol. 1, no. 1, pp. 69–90, Apr 1999.

[42] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[43] E. Vorontsov, C. Trabelsi, S. Kadoury, and C. Pal, "On orthogonality and learning recurrent networks with long term dependencies," *arXiv preprint arXiv:1702.00071*, 2017.

[44] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[45] I. Loshchilov and F. Hutter, "Sgdr: stochastic gradient descent with restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[47] Y. Tsvetkov, M. Faruqui, and C. Dyer, "Correlation-based intrinsic evaluation of word vector representations," *arXiv preprint arXiv:1606.06710*, 2016.

Blekinge Institute of Technology, Campus Gräsvik, 371 79 Karlskrona, Sweden