E-mail classification with machine learning and word embeddings for improved customer support

Anton Borg¹ · Martin Boldt¹ · Oliver Rosander¹ · Jim Ahlstrand¹

Received: 16 August 2018 / Accepted: 3 June 2020 / Published online: 19 June 2020
© The Author(s) 2020

Abstract

Classifying e-mails into distinct labels can have a great impact on customer support. By using machine learning to label e-mails, the system can set up queues containing e-mails of a specific category. This enables support personnel to handle request quicker and more easily by selecting a queue that match their expertise. This study aims to improve a 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. By using the frameworks, TensorFlow, Scikit-learn and Gensim, the authors conduct a number of experiments to test the performance of several common machine learning algorithms, text-representations, word embeddings to investigate how they work together. A long short-term memory network showed best classification performance with an F_1 -score of 0.91. 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 e-mails. Further, the study also presents a Web-based interface that were implemented around the LSTM network, which can classify e-mails into 33 different labels.

Keywords E-mail classification · Machine learning · Long short-term memory · Natural language processing

1 Introduction

Communication is part of everyday business, and it is vital for operations to run smoothly as well as for establishing stable and positive relations with customers. A crucial aspect of the latter is to efficiently resolve various business-related issues that customers encounter, since failing to do so risk negatively affect both the image and the reputation of the corporation. In highly competitive markets, a single negative customer service experience can deter potential

new customers from a company or increase the risk of existing customers to drop out [22], both negatively affecting the sales. Although recent years have shown a shift in the means of communication between customers and customer service divisions within corporations, e.g., using autonomous chat-bots or social network-based communication solutions, traditional e-mails still account for an important means of communication due to both its ease and widespread use within almost all customer age groups. Thus, implementing efficient customer service processes that target customer e-mail communication is a necessity for larger corporations as they receive large numbers of such customer service e-mails each day.

This is also true for customer service in the telecommunication businesses sector, which is mainly based on e-mail and chat correspondence. So, in this study we focus on e-mail communication and we refer to an individual e-mail from a customer as a support ticket. For small or medium-sized companies, it might be sufficient to have a single e-mail inbox for which the whole support team collaborate on customer support tickets. However, this

✉ Anton Borg
anton.borg@bth.se

Martin Boldt
martin.boldt@bth.se

Oliver Rosander
oliver.rosander@student.bth.se

Jim Ahlstrand
jim.ahlstrand@student.bth.se

¹ Department of Computer Science and Engineering, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

approach is not scalable, as the company grows, the support team also grows. Consider a scenario with a large support team divided into smaller specialized teams that each handle different errands.

In order to optimize the performance and minimize 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 labor-intensive. Failing to sort and assign messages to a suitable team would result in both inefficient use of the support personnel as well as inferior replies to the incoming support tickets. This could result in overall decrease quality of service and also that support tickets remain unsolved for longer times. However, automating the sorting and assignment to support teams are not trivial tasks because of the complex natural languages that has to be understood by the model. Any model that is doing processing of natural languages, i.e., a language used by humans to communicate, is performing natural language processing (NLP) [8].

Automating e-mail labeling and sorting requires an NLP model that can differentiate between different types of errands and support requests. Such models must be able to do this even if the e-mail contains spelling mistakes, previous conversations, irrelevant information, different formatting or simply rubbish. One such interesting candidate is the long short-term memory (LSTM) model that is an extended version of the recurrent neural network (RNN) network, and it is a sequential model often used in text classification [44]. Another important part of any NLP solution is the word embedding models that aim to model the words of a language in a vector space and placing words with similar semantic meaning close to each other [27, 33]. This helps the classifier to understand the meaning of the text and therefore improving its ability to predict the correct class [27].

In this work, we investigate the classification performance of a NLP system that uses a machine learning classifier, e.g., LSTM, to tag e-mails based on the contents of the e-mail. The tagged e-mails are then sent to the correct e-mail queue where they are processed by the specialized support personnel.

1.1 Outline

This work is structured as follows, first the use case is presented in more detail in Sect. 2. Then, the related work and the identified research gap is presented in Sect. 3. Next, the background is presented in Sect. 4, from which the experiments, discussions and conclusions are based on. Section 5 describes the method and how each experiment was conducted. The results are presented in Sect. 6, followed by the discussion and conclusion which is presented in Sects. 7 and 8, respectively.

1.2 Aims and objectives

This study aims to investigate how the use of an automated machine learning-based classifier can increase classification performance when it comes to classify incoming customer support e-mails in the Swedish branch of a large telecommunication company. The studied classifiers are evaluated on a dataset labeled using manually handled keyword-based rules, and which acts as the baseline in the study.

An extensive study of relevant variables is conducted in order to model the problem correctly. Thus, the study investigates:

- To which degree the NLP model (e.g., word2vec) affects the classifiers classification performance.
- How well LSTM compare to non-sequential machine learning models in classifying e-mails.
- To which degree the corpora affects the LSTM performance, i.e., whether the model requires only the provided e-mail dataset or whether additional language information is needed.
- To which degree the LSTM network size and depth affect the classifier performance, which is useful during the parameter tuning of the model.
- How the aggregation of class labels affects the classification performance compared to having distinct labels.

2 Case setting

In the Introduction, the problem of scaling customer support were touched upon. The problem description is based on a real case setting, and the results of this study have been implemented in the form of a customer service e-mail management system. This system uses a supervised learning paradigm with a multi-class classifier, see Fig. 1 for an example of the system.

The customer service e-mail management system exists within one of the bigger telecom operators in Europe with over 200 million customers worldwide, and some 2.5 million in Sweden. When these customers experience problems, they often turn to e-mail as their means of communication with the company, by submitting an e-mail to a generic customer service e-mail address. Consequently, such customer service e-mails might be sorted into an global inbox or assigned to random customer service personnel. In the former case, support agents might have to look through several e-mails before they come across one they are suited to handle. In the latter, the support personnel might be assigned an e-mail they are ill-equipped to

Fig. 1 Screenshot depicting the interface used by customer support agents implementing the proposed approach. Agents are able to toggle specific queues, as well as instantly get the topic of an e-mail

handle. That is, the experience and knowledge possessed by customer support personnel concerning the different areas requiring support might differ. A person with knowledge in the financial aspects of the business does not have the same knowledge in the technical aspects. As explained earlier, in this setting this is done by dividing the customer support personnel into teams with different areas of expertise. Each team has their own inbox or e-mail queue. E-mails are assigned to the queues depending on their content. The problem then becomes how to assign the e-mails to the correct support personnel based on the content of the messages. To address this problem, an intelligent model that classifies the content, i.e., the type of issue in an e-mail makes it is easier to direct e-mails to the most suitable handlers.

The implemented model labels each e-mail based on their content, an process that previously was done using a rule-based approach. These labels are then used in the customer support organization, where manager can set up support queues. A support queue consists of a combination of labels decided by a manager, e.g., *queue₁* consists of e-mails that can be labeled with either *ChangeUser*, *Invoice*, *Assignment* and *queue₂* might consist of e-mails that can be labeled with either *Order*, *TechnicalIssue*. The different customer support teams then subscribe to queue decided by their manager. Throughout their workday, customer support personnel picks e-mails from their queue to work with. Less time is spent by customer support personnel locating e-mails of their topics or answering support errands outside of their area of expertise. Consequently, by enabling a high-accuracy labeling of the received e-mails, customer support efficiency is improved.

3 Related work

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

Managing the increased number of e-mails 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 e-mail 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 experiences 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 [40].

Coussement and Van den Poel propose an automatic e-mail classification system that is intended to separate complaints from non-complaints. They present a boosting classifier which labels e-mails as either complaints or non-

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

complaints. The authors also argue that the use of linguistic features can improve the classification performance [12]. Selecting a corpus to train word vectors that are used by sequential models is not a trivial task.

The use of domain-specific language is 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 model [10]. 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, Bag of Words (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 [16].

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 [35].

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 e-mails 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 [48].

Gabrilovich and Markovitch compare SVM with the C4.5, a decision tree (DT) algorithm on text categorization. The C4.5 algorithm outperforms 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 [19].

3.1 Research gap

The research gap of the present study is twofold. First, although there exists research on several of the topics required to successfully classify e-mails [48], i.e., models

that interpret natural language [9, 33, 34, 37] and classifiers that utilize the relations of words in a time series [44]. Little research exists that investigates how the choice of NLP model, corpora, aggregation of classification labels, LSTM network size and depth affect the classification performance. Thus, this is the primary research gap that motivates the present study.

Secondly, there exists much research on various machine learning approach targeting document classification. However, considerably less research exists on e-mails specifically even though e-mails constitutes a distinct group of documents, since they are informal, enables a leveled playing field in terms of social hierarchy, encourages personal enclosure and can become emotional [2]. These distinctions may have to be accounted for when creating the machine learning model.

Additionally, a majority of the recent research has been studied on the English language, and only a few studies has been conducted on the Swedish language.

Taken together this motivates a study that investigates factors affecting NLP classification of Swedish e-mails using LSTM networks, which are compared to other state-of-the-art machine learning candidates as well as a manually managed rule-based classifier.

4 Background

This background covers central concepts that this study rests on, e.g., NLP approaches, text representations and preprocessing methods. The models and algorithms are explained as well as the underlying theory that defines them.

4.1 Natural language processing

A computer that takes any form of natural language and processes it in any way is using NLP [7, 25]. The number of applications is vast, ranging for instance optical character recognition (OCR) that is used by both banks to scan checks as well as post offices for scanning addresses of mails. Another example is voice commands in various settings such as smartphones, which allows the end-user to search the Internet or create notes without touching the device [8].

With the use of natural languages, we can communicate effectively across many domains and situations. However, because natural languages are mostly ambiguous it makes a difficult barrier for computers. Take the phrase “The trophy did not fit in the bag, it was too big” for example, what does “it” refer to, the bag or the trophy? This may seem like a trivial question for a human because we know that big things do 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.

4.2 Text representation

Using machine learning classification requires the text to be represented in a manner that the classification algorithms can process. Transforming the data into the correct format is dependent on the type of data [24]. 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 sentences of variable length. The sentences 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* [17]. 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. 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.

4.2.1 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 [17]. 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 [7]. 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.

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.

4.2.2 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) [24]. 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. Further, all terms have the same weights and are therefore seen as equally important when modeling the document, even this is not the case [9].

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 includes all combinations of adjacent words, i.e., bi-grams.

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 Eq. (1) where N is the total number of documents [9].

$$idf_t = \log \frac{N}{df_t} \quad (1)$$

4.2.3 Word2Vec

The Word2Vec model is based on the assumption that words with similar semantics appear in the same context. This can be modeled 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 [33], skipgram [34], and GloVe [37]. 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 maximize 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 [31, 33].

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 [34].

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

$$P_{ij} = P(j|i) = \frac{X_{ij}}{X_i} \quad (2)$$

This makes GloVe a hybrid method as it models probabilities based on frequencies [37].

4.2.3.1 Average word vector 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 (3) 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 [13].

$$\frac{1}{n} \sum_{i=0}^n w_i \quad (3)$$

4.2.4 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 “funnier”, 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.

$$V_{\text{big}} - V_{\text{bigger}} + V_{\text{fun}} \approx V_{\text{funnier}} \quad (4)$$

These analogies can be formulated as either syntactic or semantic questions. Syntactic analysis focuses 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 [33, 37].

4.3 Classification

Single-label text categorization (classification) is defined as the task of assigning a category to a document given a predefined set of categories [41]. 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 number of labels that correspond with the actual labels of the document [41].

A fundamental goal of classification is to categorize 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, which learn to generalize categories from previously seen documents to previously unseen documents. Typically, machine learning algorithms are divided into three different groups. Namely, geometrical, probabilistic and logic-based models [17]. The different groups of classifiers achieve the same goal but using different methods. These classifiers are hereafter referred to as non-sequential classifiers since they do not handle the words in the e-mails in a sequence. A sequential classifier, such as LSTM, handles each word in the e-mail sequentially, which allows it to capture relations between words better and therefore possibly utilize the content of the e-mail better than a non-sequential classifier.

4.3.1 Machine learning classifiers

The machine learning models included in this study are selected based on their group, diversity and acceptance in the machine learning community. Support vector machine (SVM), Naïve Bayes (Naive Bayes (NB)) and decision trees (DT) are from three different groups of classifiers, each using its own learning paradigm. ADA is used to test a boosting classifier and artificial neural network (ANN) is used to compare a non-sequential neural network against a sequential neural network, such as LSTM.

Support vector machine

SVM are based on the assumption that the input data can be linearly separable in a geometric space [11]. 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 maximizes the margin between two classes [11]. 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 modeled 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 fulfill 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 decided, which can be difficult to achieve with other classifiers.

Naïve Bayes NB is a probabilistic classifier which is build on Bayes' theorem,

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \quad (5)$$

where A is the class and B is the feature vector [14, 29, 50]. The probabilities of $P(B|A)$, $P(A)$ and $P(B)$ are estimated from previously known instances, i.e., training data [14, 29]. The classification errors are minimized by selecting the class that maximizes the probability $P(A|B)$ for every instance [29].

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 dependent [14]. Real-world data does often not meet this criterion, but researchers have shown that NB still perform better or similar to C4.5, a decision tree algorithm in some settings [14].

AdaBoost ADA is built upon the premise that multiple weak learners that perform somewhat good can be combined using boosting to achieve better result [18]. This algorithm performs 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 classifier is given a subset of the training data which each instance in the training data is given a probability that is decided by the previous weak classifier's 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 performs well is given more influence than a classifier that perform bad.

4.3.2 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 [17]. A perceptron is a linear binary classifier consisting of weights and a bias [17]. Connecting several perceptrons in layers allows accurate estimations of complex functions in multi-dimensional space. Equation (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.²

$$a(W \times X + b) \quad (6)$$

² Normally Softmax or Rectified Linear Unit (ReLU) is used as activation functions but several others exists.

The weights and biases in ANN have to be tweaked in order to produce the expected outcome. This is done when training the network which usually is done using backpropagation. The backpropagation algorithm is based on calculating the gradients given a loss function and then edit the weights accordingly given a optimization 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 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 (7) describes the hidden layer h at time t from the RNN network. Equation (8) describes the output layer of the RNN network [20].

$$h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (7)$$

$$y_t = W_{hy}h_t + b_y \quad (8)$$

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 [44].

Long short-term memory The LSTM network was developed in order to avoid the gradient problems introduced in RNN [21, 23, 35, 44]. 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 [35].

Cross-entropy loss The cross-entropy loss, also known as Kullback–Leibler divergence, 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. Minimizing the cross-entropy loss will lead to predictions that are closer to the ground truth.

$$H(p) = - \sum_{i=1}^n p_i \times \log_b p_i \quad (9)$$

$$H(p, q) = - \sum_{i=1}^n p_i \times \log_b q_i \quad (10)$$

Cross-entropy is based on the Shannon entropy function that is calculated according to Eq. (9), where p_i represents the probability for some event i [42]. The cross-entropy is the difference between two probability distributions, e.g., where p and q represent a model and the ground truth, respectively. The outcome is the number of extra bits that are needed to represent the latter using the model, see Eq. (10).

Gradient descent optimiser The gradient descent algorithm is an optimiser which minimizes 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 [4]. Mean square error is shown in Eq. (11), while cross-entropy loss was described previously.

$$C = \frac{1}{2} (D - O(Z, W))^2 \quad (11)$$

4.3.3 Overfitting

A desired trait in machine learning models is its ability to generalize over many datasets. Generalization in machine learning means that the model has low error on examples it has not seen before [38]. Two common measures that usually are used to indicate how well the model fits the data are bias and variance. The bias is a measure of how much the model differ from the desired output over all possible datasets. The variance is a measure of how much the model differ between datasets.

In the beginning of the training, a model’s 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, which 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 generalize over datasets [4]. 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 exist methods to avoid overfitting. Early stopping is one of them and involves stopping the training of the model due to some stopping criterion, e.g., human interaction or low change in loss [38]. 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 neurons are updated at the same time, they each learn to recognize different patterns and therefore reduce the overall overfitting of the network [43].

5 Methods

This section describes the experiments design, the evaluation metrics and procedures, data collection, preprocessing and word representation.

5.1 Experiment design

Two branches of experiments will be conducted with focus on sequential and non-sequential models. As stated earlier, there are three common categories of NLP models when it comes to text processing, *count based*, *prediction based* and *sequential* [17]. Consequently, two sets of experiments are conducted: sequential and non-sequential experiments. This will allow an indication of which approach are more appropriate for this problem setting. While there are differences in how the experiments are conducted that make direct comparisons difficult, the results still indicate the performance of the algorithms in this problem setting.

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. This is conducted through four experiments, detailed in Sect. 5.8. Evaluating all the combinations of corpora, NLP models and classifiers is not feasible due to the increased complexity. 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.

5.1.1 Non-sequential classifier experiments

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 described in Sect. 5.6. Friedman test is used to test if there is a significant difference in the performance. If the Friedman test shows a significant difference, a Nemenyi test is performed to show which algorithms that perform different. The classifiers will be trained on a subset of 10,000 e-mails chosen randomly because of the drastic increase in training time when increasing the number of e-mails.

5.1.2 Sequential classifier experiments

The experiments on the sequential models will evaluate which combination of corpus, text representation and LSTM hyperparameters that shows highest classification performance using the chosen evaluation metrics.

The LSTM network was built using the 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 1. Limiting the e-mails to 100 words was a trade-off between batch size and run-time, since each batch consist of a matrix which had to fit in the graphic card's memory of 11 GB. Increasing the word limit above 100 words did not seem to increase the performance of the classifier during initial studies, but rather increasing the training time significantly.

Orthogonal initialization is a way of reducing the problem with exploding or vanishing gradients which hinder long term dependencies in a neural network [47]. The rest of the settings was set to values which achieved

Table 1 LSTM network hyperparameters

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

the best results, although a systematic hyperparameter tuning may lead to increased performance. Initial experiments were conducted into the number of cells (128, 512, 1024 cells) and the depth layers (1 or 2 layers).

However, the differences were negligible between the different sizes and layers, e.g., the network with 1024 cells and two layers only resulted in about 1% higher Jaccard index measurements compared to the smallest network size. However, the training time was several factors longer for the bigger network, which was infeasible for this study. Consequently, Experiment 5.8.3 and 5.8.4 were based on 128 cells in two layers due to this trade-off between performance and execution time.

The data used in the experiment using LSTM differs from the non-sequential experiments in two ways. First, neural networks often require larger amounts of data (depending on variations in the data, number of layers, drop-out rate, etc.³) compared to non-sequential models. As such, the data used by the sequential model is not subsampled. Secondly, the sequential model is not validated with a 10-times 10-fold cross-validation setup due to time constraints. Instead, a static 90/10 train/test split was used, i.e., the test set consisted of a random 10% sample while the remaining 90% of the data was used for training the model. The sets are randomly chosen from a uniform distribution without class balancing. These experiments are measured using accuracy, precision, recall, F_1 -score and the Jaccard index. As such, it should be noted that the sequential and non-sequential experiments are done using different data. While a direct comparison between the two sets of experiments are not possible, the results still indicate the performance of the algorithms in this problem setting.

While, Friedman test and the Nemenyi post hoc test will be performed when investigating the non-sequential models, the sequential models could not be analyzed using statistical tests since there only were one measurement for those models.

5.2 E-mail dataset

The e-mail dataset used during the experiments consists of 105,195 e-mails from support environment of a large telecom corporation. The e-mails contain support errands regarding for instance invoices, technical issues, number management, admin rights, etc. They are classified with one or more labels, and there are in total 33 distinct labels with varying frequency, as shown in Fig. 2. The label “DoNotUnderstand” is an artifact from the manually constructed rule-based system where an e-mail did not

match any rule, and there exist 31,700 e-mails with the label “DoNotUnderstand”. This results in a classification rate of 69.9% by the currently implemented manual rule-based system. Figure 2 also shows 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 “DoNotUnderstand” label was filtered out and was not used during training or testing of models in this study.

The e-mail labels can be aggregated into queue labels which is an abstraction of the 33 labels into eight queue labels. The merger is performed by fusing e-mails from the same e-mail 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 e-mail labels and their contents. If an e-mail contains two or more labels, it is disregarded since it might introduce conflicting data which is unwanted when training the classifier. Without “DoNotUnderstand” and the multi-label e-mails there are a total of 58,934 e-mails in the dataset.

Each e-mail contains a subject and body which is valuable information for the classifier. The e-mails may also contain Hypertext Markup Language (HTML) tags and meta data which are artifacts from the infrastructure. The length of each e-mail varies; however, the average is 62 characters. Figure 3 shows the length distribution where e-mails under 100 characters is the most common.

5.3 Data preprocessing

An e-mail goes through several preprocessing steps before classification which removes redundant data and increases the overall quality of the information found in the e-mail. First, HTML tags and metadata are removed since it does not contribute to the understanding of the e-mail. Then the e-mail is converted to lower case, and the e-mail subject and body are extracted. Only the latest body is extracted from the e-mail, and no previous parts of the conversation is considered. Next the e-mails are cleaned, and extra newlines, tabs, punctuation, commas, and whitespace are removed. Numbers are replaced with a number token. Further, undesired characters are also removed.

5.4 Data collection for word corpus

When collecting data for word vectors, there are several points to consider. First, the data needs to be extensive, i.e., the more the better as a general rule. To accomplish this the Swedish Wikipedia [32] were used which can be downloaded online,⁴ the 2000–2015 collection of Web crawling

³ https://www.researchgate.net/post/What_is_the_minimum_sample_size_required_to_train_a_Deep_Learning_model-CNN.

⁴ <https://dumps.wikimedia.org/svwiki/latest/>.

Fig. 2 Label frequencies

Fig. 3 Length of each e-mail rounded to nearest 100 characters

from Swedish forums and blogs made available by Språkbanken [15] and lastly the e-mails themselves for increased domain knowledge.

The Wikipedia dataset contains about 380 million words and can be accessed 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 e-mails are formatted in HTML which also were converted into plain text. Only the subject and the body were kept from the e-mail 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 corpus on the targeted domain; in our case, it is the support e-mails and then fill the corpus with data from other sources to make it more extensive.

5.5 Word representation

The models are trained on the largest corpus based on Wikipedia, Språkbanken and e-mails. This is due to skipgram and GloVe being shown to perform better on a larger corpus and that domain-specific language can improve a NLP model [10, 28]. As a comparison, GloVe will be trained on a smaller corpus based solely on the e-mails.

Skipgram and CBoW word vectors are implemented using the Gensim Python package.⁵ The GloVe word vectors are generated using the source code.⁶ published by the GloVe authors [37]. Skipgram-ng word vectors are generated by the framework released by Facebook on Github.⁷

All word vector models are trained with the hyperparameters shown in Table 2. These are the settings that achieved the best results, although a more systematic tuning of hyperparameters may lead to even better performance.

BoW and BoWBi is implemented using Scikit-learn. To reduce the number of features and improve the quality some filtering feature is done by two hyperparameters: *Minimum document frequency* of 0.001, and *Maximum document frequency* of 0.01. The rest of the settings were default. The hyperparameters increased the performance compared to the default values. BoW consist of 2374 features and BoWBi consist of 7533 when trained on the e-mails.

5.6 Evaluation metrics

For classification problems, it is common to use a confusion matrix to determine the performance [36]. The confusion matrix for a two-class classification is build from four terms, true positive (TP), true negative (TN), false positive (FP) and false negative (FN). Table 3 shows how said positives and negatives are defined and used in this paper.

There exists several metrics that utilize the confusion 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 Eq. (12). In a multi-class problem, in our case it is e-mail labeling with

⁵ <https://pypi.python.org/pypi/gensim>.

⁶ <https://nlp.stanford.edu/projects/glove/>.

⁷ <https://github.com/facebookresearch/fastText>.

Table 2 Word vector hyperparameters

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

Table 3 Positives and negatives definition

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

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% [49]. To gain better insight, we also measure the Jaccard index seen in Eq. (13). The Jaccard index disregard the TN and only focus on the TP which makes the results easier to interpret. Equation (14), precision, measure how many TP there are among the predicted labels, while Eq. (15), recall, measure how many labels that are correctly selected amongst all labels. A classifier that predicts all available labels 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 [17]. A good F_1 -score is only achieved if there both the precision and recall are high. The F_1 -score make an implicit assumption that the TN are unimportant in the operative context, which they are in this context.

Olson and Delen defines the following metrics for evaluating predictive models [36] as described in Eqs. (12), (13), (14), (15) and (16). These measurements are used to give insights in the classifier's performance on previously unseen e-mails.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (12)$$

$$\text{JaccardIndex} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \quad (13)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (14)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (15)$$

$$F_1\text{-score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \quad (16)$$

5.7 Statistical tests

The statistical tests below are used to draw correct conclusions from the results that are generated. The tests are applied to the metrics described above where possible.

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 [17]. The test is nonparametric, based on ranking, and does therefore disregard the distribution of the measurements. 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%.

Nemenyi test The Friedman test does only measure if there is a significant difference in the performance of the algorithms that are compared, since it does not do any pairwise measurement [17]. 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. 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.

5.8 Experiments

In this section, the different experiments are detailed. The results are presented corresponding in subsections in Sect. 6. The experiments were conducted on a computer equipped with 64 GB DDR4 non-ECC memory, Nvidia GTX 1080Ti graphics card and an Intel Core i7-7820X 3.6 GHz processor. For the development environment, Jupyter Notebook were used with a Python3 kernel. TensorFlow [1] v1.3, Scikit-learn [6] v0.19 and Gensim [39] 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.

5.8.1 Experiment 1: NLP semantic and 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 e-mails, by using an analogy dataset. Further, GloVe will also be trained on a smaller corpus based only on the e-mails for comparison. To evaluate the different models, 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 includes 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 2.

5.8.2 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 labeled e-mails with 33 classes. Experiment 1 does not test the NLP models in a classification task, which is the motivation for this experiment. The aim of this experiment is to add 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 e-mails, and it is evaluated on a LSTM network using the hyperparameters in Table 1, except for the hidden layers where 256 cells were used. The NLP models are trained with the hyperparameters shown in Table 2.

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.

5.8.3 Experiment 3: NLP corpus and 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 was set to 128 cells and two layers, which were decided through a pre-study as described in Sect. 5.1.2. In this experiment, we also performed one reboot once the network triggers early stopping or the maximum epochs. These classifiers will be tested on both the 33 e-mail labels and the aggregated eight queue labels.

5.8.4 Experiment 4: Non-sequential models performance

This experiment will be used as a baseline for comparison against the LSTM network.

The models ADA, ANN, DT, NB, and SVM are trained using Scikit-learn implementations with default settings.⁸

⁸ The default values are described in the documentation <http://scikit-learn.org/stable/modules/classes.html>.

For the ANN, we use Scikit-learns MLPClassifier 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 optimized 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. These classifiers will be tested on both the 33 e-mail labels and the aggregated eight queue labels.

6 Result and analysis

In this section, we present the results of the four experiments described in Sect. 5. The performance metrics is presented together with analysis and statistical tests to verify significant differences where applicable. The performance of the word vectors, non-sequential algorithms and the sequential model with both labels and queues are presented.

6.1 Experiment 1: NLP semantic and syntactic analysis

Figure 4 shows the per category accuracy of the semantic and syntactic questions used for evaluating the investigated Word vector models, as well as the total accuracy per model. 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.

Due to the similarity of the different models, it is not possible to recommend any specific approach. It should be noted that GloVe trained on the smaller corpus solely based on the e-mails achieve a total accuracy of 2.1%, which is 64.6% units less than the best model. It can therefore be concluded that the e-mail dataset does not provide enough information to build acceptable word vector models on its own.

While all models struggled with opposite-related questions, which are semantic questions, they excel at capital/country questions that also are semantic.

6.2 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 Sect. 6.1 these results will help in understanding the impact of the different NLP models. The results from Table 4 show that the word vectors are very similar to each other;

Fig. 4 Top figure shows word vector total semantic and syntactic accuracy. Bottom figure shows Word vector semantic and syntactic accuracy per category

the word vector generated by GloVe perform slightly better than the others with regards to Jaccard, Recall, and F_1 -score. However, the differences between the models are small and drawing any general conclusions is therefore difficult. However, the word vectors trained by GloVe showed best performance that model will be used in the following experiments.

6.3 Experiment 3: NLP corpus and LSTM classifier

6.3.1 LSTM classification with eight queue labels

The results from Table 5 show that the full corpus of Språkbanken, Wikipedia and e-mails perform better than the corpus only based on the e-mails. The Jaccard index and F_1 -score are 6% and 3% units higher, respectively, when LSTM is trained on the larger corpus. However, it is interesting that LSTM still achieve acceptable performance with word vectors based on a the significantly smaller corpus even though it scored terrible in the semantic and syntactic analysis as seen in Fig. 4.

6.3.2 LSTM classification with 33 e-mail labels

Table 6 shows the results when LSTM is trained on two different GloVe word vectors on different corpora. Training LSTM on the larger 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 Sect. 6.3.1 trained on queues. The decrease in F_1 -score may suggest that the corpus based on the e-mails may struggle when the number of classes grows.

6.4 Experiment 4: Non-sequential models performance

6.4.1 Non-sequential classification with eight queue labels

Table 7(a) and (b) shows the different preprocessing algorithms performance when used with different learning algorithms. From Table 7(a), the results show that BoWBi performs 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% units higher, which also is the best result obtained.

A Friedman test confirms that there is a significant difference in the performance when measuring the Jaccard index at an significance level of 0.05, $\chi^2(2) = 7.600$, p value = 0.022. But the test does not confirm a significant difference at significance level 0.01 for the F_1 -score measurements, $\chi^2(2) = 3.600$, p value = 0.166.

A Nemenyi post hoc test evaluates the difference between the preprocessing algorithms on Jaccard index. The results from Table 8 show a significant difference between BoW and BoWBi. Even though AvgWV gain the best result, it is not a significant difference because it performs worse for the rest of the learning algorithms.

When investigating if there was any significant difference in the classification algorithms performance, Table 7(a) and (b) are transformed, by swapping rows and columns. The transformation is done because of the Friedman test, which measures the difference on a column basis.

Given the results in Table 7, ANN and SVM show best performance when trained on AvgWV. The average rank from the transposed Table 7(a) and (b) shows 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. Another Friedman test for the non-sequential algorithms on the Jaccard index results in Table 7(a) shows that there exists significant differences between the candidates at significance level 0.05, $\chi^2(2) = 10.667$, p value = 0.031. The F_1 -score results show the same pattern,

Table 4 Performance metrics for each word vector algorithm used in LSTM classification model

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 5 Comparing the same LSTM network trained on different corpora and eight queue labels

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

Table 6 Comparing the LSTM network trained on different corpora and 33 e-mail labels

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

Table 7 Jaccard index and F_1 -score on queue labels with non-sequential algorithms and different preprocessing algorithms

Algorithm	BoW	BoWBi	AvgWV
(a) Jaccard index			
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
(b) F_1 -score			
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

Bold denotes the highest performance

Table 8 Nemenyi post hoc test on Jaccard index based on Table 7(a)

Algorithm	BoW	BoWBi	AvgWV
BoW		0.031	0.069
BoWBi	*		0.946
AvgWV			

* Significant at $p < 0.05$

** Significant at $p < 0.01$

$\chi^2(2) = 10.237$, p value = 0.037, which also rejects the null hypothesis at significance level 0.05, i.e., that all candidates perform equal.

Table 9 Nemenyi post hoc test on non-sequential algorithms Jaccard index

Algorithm	ADA	ANN	DT	NB	SVM
(a) Jaccard index					
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				*	
(b) F_1 -score					
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 9(a) and (b) shows the results from two Nemenyi post hoc test. These results indicate that there are no significant differences between the candidate algorithms, except for the difference between SVM and NB that is significant at significance level 0.05. Together with the results from Table 7(a) and (b) it is clear that SVM is the best performing candidate.

The box plot in Fig. 5 shows the classification performance over 10 folds for a combination of the best performing preprocessing algorithm and classification algorithm. The variance is low for all algorithms which is a good indication that the model does not overfit and can generalize well for previously unseen e-mails.

Fig. 5 Jaccard index per algorithm, for the best performing combination of preprocessing method and learning algorithm, on the aggregated e-mail queues

6.4.2 Non-sequential classification with 33 e-mail labels

The results in this section indicate how well different NLP models, in combination with non-sequential learning algorithms, perform in classifying e-mail topics. This together with previously shown results allows a comparison of the sequential LSTM network against the non-sequential classifiers, and how the aggregation of the 33 labels into queues affect the classification performance.

Table 10(a) and (b) shows the results when the preprocessing algorithms are tried on the 33 distinct e-mail labels. A Friedman tests on the Jaccard index, $\chi^2(2) = 3.600$, p value = 0.165, and F_1 -score, $\chi^2(2) = 2.800$, p value = 0.247, does not show any significant difference at significance level of 0.05. SVM and ANN does perform about 10% units higher when trained on AvgWV compared

Table 10 Jaccard index and F_1 -score on distinct e-mail labels with non-sequential algorithms

Algorithm	BoW	BoWBi	AvgWV
(a) Jaccard index			
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
(b) F_1 -score			
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

Bold denotes the highest performance

to the other preprocessing algorithms and classification algorithms.

When compared to the results for eight queues (instead of the 33 labels), as shown in Table 7(a) and (b), the performance decreases. This is also expected due to the increased difficulty of more classes and because some of the classes may be closely related to each other. Closely related labels further may be hard to separate for the classifiers, which could explain the drop in performance of e-mail labels compared to the queue labels.

Similarly to the experiment in Sect. 6.4.1, Table 10(a) and (b) are transformed, and evaluated using Friedman test. A Friedman test applied to the classification algorithms does show a significant difference for the Jaccard index, $\chi^2(2) = 10.667$, p value = 0.031, at significance level of 0.05, but not on the F_1 -score, $\chi^2(2) = 7.200$, p value = 0.126. From the results in Table 10, it is clear that SVM performs best using all preprocessing algorithms whereas NB performs worst in all cases.

One significant difference was found between SVM and NB as seen in Table 11 at a significance level of 0.05. There are however differences between the other algorithms although not significant.

Figure 6 shows 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.

6.5 LSTM certainty values

Figure 7 shows the certainty values for each label by the proposed LSTM model. The data are collected by classifying all instances in the test dataset, which contains 5893 e-mails 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

Table 11 Nemenyi post hoc test on non-sequential algorithms performance with e-mail labels

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 $p < 0.01$

Fig. 6 Jaccard index per algorithm, for the best performing combination of algorithm and preprocessing method, on the distinct e-mails topics

label. The circles indicate outliers, i.e., points diverging at least 1.5 times the inter-quartile range.

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 fewer promising results and may need further improvements.

7 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 2–3% units 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 [30]. 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.

In order to determine which word vector model to use, each model was 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 1% unit better than the other models. However, when the word vectors were used in training of the LSTM model, they showed 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 e-mails does model the domain language which may be what the LSTM network utilize. Incorporating domain language in a corpus is therefore 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, we have to consider the efficiency of the algorithm that is used. The energy usage can differ severely between different algorithms depending on several factors. Execution time is one of these factors. Table 12 presents the execution time of the algorithms trained on 10,000 e-mails. 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 1 s. SVM is the slowest of the non-sequential algorithms with a training time of 39 s. 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.

The execution times in Table 12 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 [30]. If execution time or 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 optimize the network does require domain knowledge and experience with gradient-descent classifiers.

Fig. 7 Certainty values per label using the proposed LSTM model based on the test dataset, illustrated with a box plot

Table 12 Execution time in seconds when trained on 10,000 e-mails

Algorithm	Time (s)
LSTM	20,068.00
SVM	39.13
ANN	9.37
DT	6.63
ADA	6.26
NB	0.03

A machine learning-based classifier could help a company to reduce work hours that are spent on e-mail support. The classifier could be trained to forward incoming e-mails to personnel or groups that handle different types of errands. If the company does use a manually created rule-based classifier, 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. A framework was implemented that 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 rule-based system in the company without any larger infrastructure changes.

While 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 not, but it is instead dependent of tuning. It is challenging to control that outcome since the classification is somewhat of a black box. However, 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 high-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 labeled data could be produced by the support team. Temporary labels such as campaigns where good labeled 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 Sect. 5.2. 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 e-mail which can be interpreted as a classification rate of 100%. However, if the proposed model does not understand the e-mail it will still assign the e-mail a label, but with low certainty. The certainty of a classification can be used to determine if the model understands the e-mail or not; however, at what level of certainty this can be determined is not obvious. As shown in Fig. 7, the average certainty values of most of the labels are quite high. If 80% certainty is considered as the model's threshold for "understanding" an e-mail, then 92% of the e-mails 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 e-mails from, the network may assign an e-mail to an employee directly. It might be possible to extend the network with one or more neural networks that specializes in learning which employees that prefer which e-mails. 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 e-mails based on current load, expertise, satisfaction rate etc.

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

Finally, there are a number of potential validity threats related to this study, which are discussed in the following paragraphs. Even though 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,⁹ 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 expected. The downside of aggregating the labels are the reduced flexibility and granularity as the system now has less freedom when sorting e-mails 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 optimize 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 led 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 e-mail 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 questions were used. These questions were defined by the authors and considered extensive. 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 of 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 [46].

8 Conclusion

Of the six different classifiers that are evaluated LSTM performs best on both the individual 33 labels as well as for the aggregated eight queues. The LSTM network achieves almost as good results when using the 33 labels as when using the queue labels. Aggregating the labels does increase the performance, but only nominal. It should be noted that the NLP model used does not significantly affect the classification performance as LSTM seem to compensate for the difference between them.

⁹ Google's Inception network [45] consists of 1000 classes.

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 both BoW and BoWBi if used with a suitable classifier.

When comparing LSTM with the non-sequential algorithms, LSTM perform about 1.5% units better on Jaccard index and 31% units better on F_1 -score compared on the 33 labels. It should be noted that, 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 model 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 it supports, e.g., 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.

9 Future work

Extending the classification to identify emotions in the e-mail 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 e-mail conversation, but often keeps the subject of the original e-mail 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 separately. The two networks may then be interlaced by a fully connected neural network.

Currently, the e-mails are processed before entering the classifier. In the early stages of preprocessing, all other bodies than the first are stripped. The other bodies contain 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 is 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. An approach that would allow the network to adapt continuously to changes in the e-mail environment is of beneficial. As such either online learning or reinforcement learning could be useful approaches. Future work may look closer at the benefits and usefulness of online learning and reinforcement learning in this context.

Finally, the problem of distribution drift also needs to be addressed [26]. Given the problem setting, it is safe to assume that the class prevalence, i.e., the size of each class, change over time. For example something might affect hardware causing a larger amount of technical support messages for a certain week. Two approaches for investigating are of interest to the authors; *conformal prediction* as an indicator of when the model is certain of its predictions [3], and as a *quantification* problem to calculate how well the distribution of the predicted classes fit the distribution of the training classes [26].

Acknowledgements Open access funding provided by Blekinge Institute of Technology.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org
2. Baron NS (1998) Letters by phone or speech by other means: the linguistics of email. *Lang Commun* 18(2):133–170
3. Borg A, Boldt M, Svensson J (2019) Using conformal prediction for multi-label document classification in e-mail support systems. In: International conference on industrial, engineering and other applications of applied intelligent systems. Springer, pp 308–322

4. Bottou L (2012) Stochastic gradient descent tricks. In: Neural networks: tricks of the trade, 2nd edn. Springer, Berlin, pp 421–436
5. Bougie R, Pieters R, Zeelenberg M (2003) Angry customers don't come back, they get back: the experience and behavioral implications of anger and dissatisfaction in services. *J Acad Mark Sci* 31(4):377–393
6. Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, Niculae V, Prettenhofer P, Gramfort A, Grobler J, Layton R, VanderPlas J, Joly A, Holt B, Varoquaux G (2013) API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD workshop: languages for data mining and machine learning, pp 108–122
7. Cambria E, White B (2014) Jumping NLP curves: a review of natural language processing research. *IEEE Comput Intell Mag* 9:48–57
8. Chowdhury GG (2003) Natural language processing. *Annu Rev Inf Sci Technol* 37(1):51–89. <https://doi.org/10.1002/aris.1440370103>
9. Christopher DM, Prabhakar R, Hinrich S (2008) Introduction to information retrieval. *Introd Inform Retrieval* 151:177
10. Coden AR, Pakhomov SV, Ando RK, Duffy PH, Chute CG (2005) Domain-specific language models and lexicons for tagging. *J Biomed Inform* 38(6):422–430
11. Cortes C, Vapnik V (1995) Support-vector networks. *Mach Learn* 20(3):273–297
12. Coussement K, den Poel DV (2008) Improving customer complaint management by automatic email classification using linguistic style features as predictors. *Decis Support Syst* 44(4):870–882
13. De Boom C, Van Canneyt S, Demeester T, Dhoedt B (2016) Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recogn Lett* 80:150–156
14. Domingos P, Pazzani M (1996) Beyond independence: conditions for the optimality of the simple bayesian classifier. In: Proceedings of 13th international conference machine learning, pp 105–112
15. Eide SR, Tahmasebi N, Borin L (2016) The Swedish culturomics gigaword corpus: a one billion word Swedish reference dataset for NLP
16. Fallgren P, Segeblad J, Kuhlmann M (2016) Towards a standard dataset of Swedish word vectors. In: Sixth Swedish Language Technology Conference (SLTC)
17. Flach P (2012) Machine learning: the art and science of algorithms that make sense of data. Cambridge University Press, Cambridge
18. Freund Y, Schapire RE (1995) A decision-theoretic generalization of on-line learning and an application to boosting. In: European conference on computational learning theory. Springer, pp 23–37
19. Gabrilovich E, Markovitch S (2004) 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, p 41
20. Graves A, Mohamed A, Hinton G (2013) Speech recognition with deep recurrent neural networks. In: 2013 IEEE International conference on acoustics, speech and signal processing, pp 6645–6649. <https://doi.org/10.1109/ICASSP.2013.6638947>
21. Graves A, Schmidhuber J (2005) Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Netw* 18(5):602–610 *IJCNN* 2005
22. Halpin N (2016) The customer service report: Why great customer service matters even more in the age of e-commerce and the channels that perform best. <http://www.businessinsider.com/customer-service-experiences-are-more-important-than-ever-in-the-age-of-e-commerce-2016-3?r=US&IR=T&IR=T>
23. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
24. Witten IH, Frank E, Hall M (2011) Data mining—practical machine learning tools and techniques, 3rd edn. Elsevier, Amsterdam
25. Ingersoll GS, Morton TSM, Farris D (2013) Taming text: how to find, organize, and manipulate it, 1st edn. Manning Publications, Shelter Island
26. Kar P, Li S, Narasimhan H, Chawla S, Sebastiani F (2016) Online optimization methods for the quantification problem, pp 1625–1634. <https://doi.org/10.1145/2939672.2939832>
27. Kusner M, Sun Y, Kolkin N, Weinberger K (2015) From word embeddings to document distances. In: International conference on machine learning, pp 957–966
28. Levy O, Goldberg Y, Dagan I (2015) Improving distributional similarity with lessons learned from word embeddings. *Trans Assoc Comput Linguist* 3:211–225
29. Lewis DD (1998) Naïve (bayes) at forty: the independence assumption in information retrieval. In: European conference on machine learning. Springer, pp 4–15
30. Loshchilov I, Hutter F (2016) SQDR: stochastic gradient descent with restarts. arXiv preprint [arXiv:1608.03983](https://arxiv.org/abs/1608.03983)
31. McCormick C (2017) Word2vec tutorial—the skip-gram model. <http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/>
32. Meta: Data dumps — meta, discussion about wikimedia projects (2017) https://meta.wikimedia.org/w/index.php?title=Data_dump&oldid=17422082. Online; Accessed 18 December 2017
33. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv preprint [arXiv:1301.3781](https://arxiv.org/abs/1301.3781)
34. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. *Adv Neural Inform Process Syst* 26:3111–3119
35. Nowak J, Taspinar A, Scherer R (2017) LSTM recurrent neural networks for short text and sentiment classification. Springer, Cham, pp 553–562
36. Olson DL, Delen D (2008) Advanced data mining techniques. Springer, Berlin
37. Pennington J, Socher R, Manning C (2014) Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543
38. Prechelt L (1998) Automatic early stopping using cross validation: quantifying the criteria. *Neural Netw* 11(4):761–767
39. Řehůřek R, Sojka P (2010) Software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks, pp 45–50. ELRA, Valletta, Malta. <http://is.muni.cz/publication/884893/en>
40. Richins ML (1983) Negative word-of-mouth by dissatisfied consumers: a pilot study. *J Mark* 47(1):68–78
41. Sebastiani F (2002) Machine learning in automated text categorization. *ACM Comput Surv (CSUR)* 34(1):1–47
42. Shannon CE (2001) A mathematical theory of communication. *ACM SIGMOBILE Mob Comput Commun Rev* 5(1):3–55
43. Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15(1):1929–1958
44. Sundermeyer M, Schlüter R, Ney H (2012) LSTM neural networks for language modeling. In: Thirteenth annual conference of the international speech communication association
45. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with

- convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
46. Tsvetkov Y, Faruqui M, Dyer C (2016) Correlation-based intrinsic evaluation of word vector representations. arXiv preprint [arXiv:1606.06710](https://arxiv.org/abs/1606.06710)
47. Vorontsov E, Trabelsi C, Kadoury S, Pal C (2017) On orthogonality and learning recurrent networks with long term dependencies. arXiv preprint [arXiv:1702.00071](https://arxiv.org/abs/1702.00071)
48. Yan Y, Wang Y, Gao WC, Zhang BW, Yang C, Yin XC (2017) Lstm²: multi-label ranking for document classification. Neural Process Lett. <https://doi.org/10.1007/s11063-017-9636-0>
49. Yang Y (1999) An evaluation of statistical approaches to text categorization. Inform Retrieval 1(1):69–90
50. Zhang H (2004) The optimality of Naive Bayes. Assoc Adv Artif Intell 1(2):3

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.