



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

---

# Text analysis for email multi label classification

Master's thesis in Computer science and engineering

SANJIT HARSHA KADAM  
KYRIAKI PANISKAKI

---

Department of Computer Science and Engineering  
CHALMERS UNIVERSITY OF TECHNOLOGY  
UNIVERSITY OF GOTHENBURG  
Gothenburg, Sweden 2020



MASTER'S THESIS 2020

# Text analysis for email multi label classification

SANJIT HARSHA KADAM  
KYRIAKI PANISKAKI



UNIVERSITY OF  
GOTHENBURG

---



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering  
CHALMERS UNIVERSITY OF TECHNOLOGY  
UNIVERSITY OF GOTHENBURG  
Gothenburg, Sweden 2020

Text analysis for email multi label classification  
SANJIT HARSHA KADAM  
KYRIAKI PANISKAKI

© SANJIT HARSHA KADAM, 2020.

© KYRIAKI PANISKAKI, 2020.

Supervisor: Marwa Naili, Department of Computer Science and Engineering

Advisor: Berenice Gudino, Bokio

Examiner: Devdatt Dubhashi, Department of Computer Science and Engineering

Master's Thesis 2020

Department of Computer Science and Engineering

Chalmers University of Technology and University of Gothenburg

SE-412 96 Gothenburg

Telephone +46 31 772 1000

Gothenburg, Sweden 2020

Text analysis for email multi label classification

SANJIT HARSHA KADAM

KYRIAKI PANISKAKI

Department of Computer Science and Engineering

Chalmers University of Technology and University of Gothenburg

## **Abstract**

This master's thesis studies a multi label text classification task on a small data set of bilingual, English and Swedish, short texts (emails). Specifically, the size of the data set is 5800 emails and those emails are distributed among 107 classes with the special case that the majority of the emails includes the two languages at the same time. For handling this task different models have been employed: Support Vector Machines (SVM), Gated Recurrent Units (GRU), Convolution Neural Network (CNN), Quasi Recurrent Neural Network (QRNN) and Transformers. The experiments demonstrate that in terms of weighted averaged F1 score, the SVM outperforms the other models with a score of 0.96 followed by the CNN with 0.89 and the QRNN with 0.80.

Keywords: natural language processing, machine learning, multi label text classification, deep neural networks, bilingual texts, emails, short texts.



## Acknowledgements

We would like to thank our supervisor, *Marwa Naili*, for the continuous feedback, for guiding us and helping us to accomplish this master thesis within the stipulated time. Moreover, we would like to thank *Berenice Gudino*, machine learning engineer at Bokio, for her feedback and guidance by helping us overcome and solve our difficulties. Finally, we would like to thank the company Bokio for giving us this opportunity and providing us with the equipment.

Sanjit Harsha Kadam, Kyriaki Paniskaki, Gothenburg, June 2020





# Contents

<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Purpose and challenges . . . . .	1
1.2 Approach . . . . .	2
1.3 Roadmap . . . . .	2
<b>2 Theory</b>	<b>3</b>
2.1 Data preparation . . . . .	3
2.1.1 Text cleaning . . . . .	3
2.1.2 Text representation . . . . .	4
2.2 Machine learning in text classification . . . . .	5
2.2.1 Naive Bayes . . . . .	5
2.2.2 Support vector machines . . . . .	5
2.2.3 Random forest . . . . .	5
2.2.4 Gradient boosting machines . . . . .	6
2.2.5 Logistic regression . . . . .	6
2.3 Deep learning in text classification . . . . .	6
2.3.1 Recurrent Neural Networks . . . . .	7
2.3.1.1 Model . . . . .	7
2.3.1.2 Variants . . . . .	8
2.3.2 Convolutional Neural Networks . . . . .	10
2.3.2.1 Model . . . . .	10
2.3.2.2 Variants . . . . .	11
2.3.3 Quasi-Recurrent Neural Networks . . . . .	12
2.3.3.1 Model . . . . .	13
2.3.3.2 Variants . . . . .	13
2.3.4 Transformers . . . . .	14
2.3.4.1 Model . . . . .	14
2.3.4.2 Variants . . . . .	14
2.4 Evaluation and Metrics . . . . .	15
2.5 Summary . . . . .	17
<b>3 Methods</b>	<b>19</b>
3.1 Preprocessing . . . . .	19

3.2	Models . . . . .	20
3.3	Evaluation . . . . .	21
<b>4</b>	<b>Results</b>	<b>23</b>
4.1	Dataset . . . . .	23
4.2	Model Performance . . . . .	26
4.2.1	Support vector machines . . . . .	26
4.2.2	Gated Recurrent Units with attention . . . . .	26
4.2.3	Very deep CNN . . . . .	28
4.2.4	Quasi Recurrent Neural Network . . . . .	30
4.2.5	Transformers . . . . .	31
<b>5</b>	<b>Conclusion</b>	<b>33</b>
5.1	Discussion . . . . .	33
5.1.1	Results comparison among the models . . . . .	33
5.1.2	Stemming vs Lemmatization . . . . .	35
5.2	Future work . . . . .	35
5.3	Conclusion . . . . .	35
	<b>Bibliography</b>	<b>37</b>
<b>A</b>	<b>Appendix 1</b>	<b>I</b>

# List of Figures

2.1	Machine learning (Original figure from [32]) . . . . .	5
2.2	Anatomy of a neural network (Original figure from [32]) . . . . .	7
2.3	An unrolled Recurrent Neural Network (Original figure from [13]) . . . . .	8
2.4	LSTM architecture (Original figure from [13]) . . . . .	8
2.5	Bidirectional LSTM architecture (Original figure from [2]) . . . . .	9
2.6	The attention mechanism (Original figure from [23]) . . . . .	9
2.7	Gated Recurrent Units (Original figure from [6]) . . . . .	10
2.8	CNN framework (Original figure from [63]) . . . . .	11
2.9	Computation structure of the QRNN (Original figure from [29]) . . . . .	13
2.10	Structure of a transformer (Original figure from [23]) . . . . .	15
3.1	Pipeline flowchart . . . . .	19
4.1	A histogram demonstrating the length of the emails . . . . .	23
4.2	Email distribution across classes . . . . .	24
4.3	Emails with multiple classes . . . . .	24
4.4	GRU training and validation loss . . . . .	28
4.5	Very deep CNN training and validation loss . . . . .	29
4.6	QRNN training and validation loss . . . . .	31
4.7	Transformers training and validation loss . . . . .	32



# List of Tables

2.1	Confusion matrix (Original table from [50]) . . . . .	16
3.1	Architecture overview of deep learning models . . . . .	21
4.1	Classes description . . . . .	25
4.2	SVM classification report . . . . .	26
4.3	GRU classification report . . . . .	27
4.4	Very deep CNN classification report . . . . .	29
4.5	QRNN classification report . . . . .	30
4.6	Transformers classification report . . . . .	32
5.1	Metrics comparison among the models . . . . .	33



# 1

## Introduction

*Natural Language Processing* (NLP) is a field of artificial intelligence, which deals with a computer's ability to understand and process lingual data. The field of NLP encompasses various tasks such as sentiment analysis, text classification, speech recognition or synthesis and semantic analysis. In today's world, companies exploit a lot of applications of NLP in order to provide and automate services such as machine translation [62], parts of speech tagging [53] or resume (CV) parsing [27].

A process with a main focus on the sorting and transformation of a text into structured, easily manageable data is called *text analysis* and it relies on NLP techniques. Text analysis includes multiple tasks such as the prediction of the next word in an email [7], smart replies for incoming messages [21], automated answers for questions based on the content [20] and so on. A common task in the field of text analysis is the multi class text classification problem. Multi class classification refers to the categorization of a sample of input text into one and only one category out of a given set of classes, which contains at least three classes. A more challenging task however, is the multi label classification. In this case, each input data-point can be classified into more than one class at once. This problem requires a model that is more capable of understanding relationships between features in the input text and the output labels for all the classes that are presented.

### 1.1 Purpose and challenges

This master's thesis focuses on the *multi label text classification* problem on *bilingual short texts (emails)*. The input data set is in English and Swedish, has a size of 5272 emails, there are 107 classes and it is provided by the company Bokio. The task of the trained model, classifier, will be to assign the proper label(s), out of 107 classes, for each input email. The final model will be used by the company Bokio for tagging the incoming emails automatically. Currently, this procedure is being performed manually by the customer support team at the company.

Most of the existing work on NLP, concerns texts that are written in the English language. That means that handling texts in Swedish is a challenge for this master's thesis, especially when the same email contains English and Swedish words at the same time. Actually, the majority of the emails contains the two languages and the same time, which also complicates the preprocessing step. Furthermore, the classifier will handle emails which usually are short texts. Shortness, sparseness and lack of contextual information in short texts are the reasons for degrading the classifier's efficiency. Another major issue is the distribution of the emails among

the classes. The total number of the classes are 107 but for some of those classes there are only a few samples which means that the results for those classes will not be efficient. Moreover, this master's thesis deals with a limited data set of only 5800 examples and this fact will raise the difficulty level. Therefore, the bilingual nature, the short length of the input examples and the relatively small size of the unbalanced data set will be the main field of focus and research contribution of this master's thesis.

## 1.2 Approach

In the machine learning field there are various approaches for dealing with a multi label classification task. The majority of those approaches rely on deep learning, which employs artificial neural networks to resolve the task. Long Short Term Memory [49] and Convolutional Neural Networks [34] have been used for multi class text classification tasks. Additionally, Recurrent Neural Network [42] and Transformers [30] have been employed for multi label text classification tasks.

After a comprehensive investigation on the multi label classification problem and based on the performance of the models, four deep learning models will be developed for this master's thesis. Those models are based on Gated Recurrent Units (GRU), Convolutional Neural Network (CNN), Quasi-Recurrent Neural Network (QRNN) and Transformers. Additionally, to those deep learning models, a more simple model will be developed which relies on support vector machines and it will be used as baseline for comparing the results of the deep learning models with it.

## 1.3 Roadmap

This master thesis starts with Chapter 2, which describes the theory and the key components of the models that will be used for solving the multi label text classification task. Chapter 3 presents a step by step description of the concrete architecture of the final developed models. Chapter 4 exhibits the results of each model and Chapter 5 includes a discussion and concludes this master's thesis.



# 2

## Theory

There are multiple models which can handle and resolve a multi label text classification task. However all the different models rely on three basic steps which are 1) data preparation, 2) model application and 3) evaluation. The next sections describe some data preparation approaches, some machine learning and deep learning classifiers as well as the metrics for evaluating the classifier's results.

### 2.1 Data preparation

Text preprocessing is a crucial step of the classification pipeline which takes place at the very beginning of the whole procedure. It is responsible for transforming the input text into a form that is more digestible to enable the classification algorithms to perform better. Text cleaning and data representation aim to increase the efficiency of the classifier.

#### 2.1.1 Text cleaning

The general steps that are taken while preprocessing a piece of input text involve steps such as the removal of extra white spaces, HTML tags, special characters, numbers, proper nouns, conversion of all text into lower case and so on [1]. The main preprocessing techniques that have been used to cleanup and prepare the input before passing it to the models to train are lemmatization and stemming.

##### **Lemmatization**

Lemmatization refers to the process of bringing every word in the input dataset back to its root form or 'lemma' [19]. This groups together sets of words that are the inflected form of the same 'lemma'. These inflections can be in the form of superlatives, plurals, changed tenses and so on. This process assesses the intended part of speech that each word in the input text belongs to and hence, its intended meaning to reduce it down to its 'lemma' form. The most commonly used package for lemmatization in the English language is the NLTK Lemmatizer. In the case words that are in Swedish however, a system of lemmatization was developed where each word is taken back to its root form by comparing it to a pre-built reference database that is built on a Swedish thesaurus. An example of how lemmatization works is that the Swedish word bokförs (posted) will be transformed into bokföra (post).

### **Stemming**

Stemming refers to the process of reducing inflected words in a piece of text back to their root (word stem) form [19]. The results from stemming however mean that the root word of a particular set of words need not necessarily be a valid word that belongs to the lexicon of the corresponding language. It just ensures that related words map to the same root word or stem. The most commonly used stemmer for the English language is the NLTK stemmer. For the Swedish language, the most commonly used one is the Snowball Stemmer in the same NLTK library. An example of stemming is the word *anpassade* (customized), which will be used as *anpass* (adaptation).

### **2.1.2 Text representation**

The data should be transformed into a format so that the model can understand and handle them. The machine learning algorithm expects numbers as input, which means that the text should be converted into numbers. For this task the steps are 1) tokenization and 2) vectorization. The step tokenization refers to how the texts will be divided into words or subtexts or n-grams. This determines the "vocabulary" of the data set, which is constituted by unique tokens. The step vectorization is this one which transforms the tokens into numerical vectors. Below are presented some vectorization techniques [22].

#### **One-hot encoding**

For each token of the vocabulary a unique index is assigned. Then each sample text is represented as a vector indicating the presence (symbolized by 1) or absence (symbolized by 0) of a token in the text.

#### **Count encoding**

In the case of count encoding the vector indicates the count of the tokens in the text.

#### **TF-IDF encoding**

Stands for term frequency-inverse document frequency and implies the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus. TF-IDF determines how relevant a token is in a sample [55].

#### **Word Embedding**

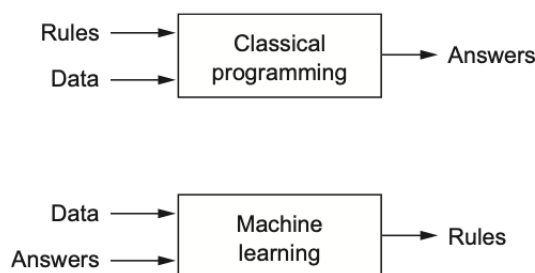
In this case the meaning of the word is taken into account. The vector of each text represents the location and distance between words indicating how similar they are semantically.

Once the text representation has been accomplished the next step is the feature selection. After having determine the tokens the vocabulary might be too large and some of those tokens (features) might not contribute to the label prediction. Feature selection is responsible to measure how much each token contributes to label predictions and for this purpose there are some statistical functions. One example

is the function `f_classif`, which calculates the feature importance [5].

## 2.2 Machine learning in text classification

In classical programming developers give the rules and data in a program and the program produces the answers. With machine learning developers provide the data and the answers and the outcome is the rules, Figure 2.1. A machine learning system is trained rather than programmed [32]. The most popular machine learning algorithms that are used for text classification are presented in the next sections.



**Figure 2.1:** Machine learning (Original figure from [32])

### 2.2.1 Naive Bayes

Probabilistic modeling is the application of statistics to data analysis. One of the most popular algorithms is the Naive Bayes algorithm which relies on Bayes theorem. Applying the theorem assigns to each class the probability that a sample belongs to the specific class [32]. In paper [37] the authors have presented a modified version of Naive Bayes for text classification with unbalanced classes. Their approach significantly improved the area under the ROC curve (Receiver Operating Characteristic, which illustrates the diagnostic ability of a classifier [15]).

### 2.2.2 Support vector machines

Support vector machines (SVM) belong to kernel methods and aim at solving classification problems by finding good decision boundaries between two sets of points belonging to two different categories. SVMs are widely used in natural language processing. At the University of Amsterdam a team of researchers utilizes structural Support Vector Machine (SVM) in order to tackle the hierarchical multi label classification task of social short text streams [57]. In [24] the authors studied the multi label text classification in Arabic and they presented a problem method which relies on SVM and achieved 71% ML-accuracy.

### 2.2.3 Random forest

Random forest is an ensemble decision tree algorithm that builds large numbers of decision trees and then ensembles their outputs [47]. An interesting contribution to

Random Forests is being described in [28] where the authors present their approach for handling a short text classification task. By enriching the data semantically and then applying Random Forests, they increased the accuracy by 34%.

### 2.2.4 Gradient boosting machines

A gradient boosting machine behaves much like a random forest, which means that it is based on assembling decision trees. It uses gradient boosting, which improves any machine learning model by training new models that specialize in addressing the weak points of the previous models. Gradient boosting machines outperforms random forests most of the time [32].

### 2.2.5 Logistic regression

Logistic regression is an algorithm which is based on the concept of probability and there are many applications in the text classification domain. In [47] the authors presented their results for classifying violent events taking place in South Africa on event detection using WhatsApp messages. They experimented with SVM, random forest, gradient boosting and logistic regression and they found that the logistic classifier achieved a 0.899 accuracy against the second best model which was the SVM with 0.895 accuracy.

## 2.3 Deep learning in text classification

Deep learning is a subfield of machine learning that relies on the artificial neural networks and learns representation from data through the use of successive layers of increasing representation stacked in neural networks that are learnt simultaneously [32]. Deep learning offers better performance on many problems. Moreover, with machine learning the developers had to manually find good layers of representations for the data. This is called feature engineering and deep learning automates this step.

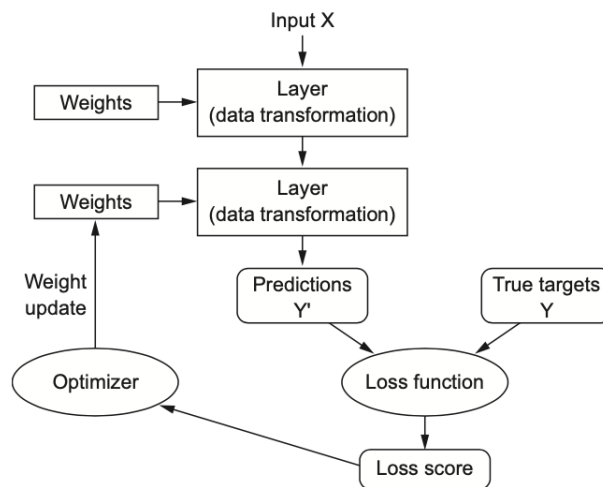
A neural network has some key components which constitute its anatomy as figure 2.2 depicts. Those components are:

- *Layers* which are combined into a network.
- *Loss function* which measures how far the output is from the expected value.
- *Optimizer* which helps the network to update itself based on the data it sees and its loss function.

Initially, the weights have random values, the output is far from the expected value and the loss score is high. The network starts processing the samples (text input), the weights are adjusted a little in the correct direction and the loss score decreases. A network with a minimal loss is one for which the outputs are as close as they can be to the targets and is called trained network.

For handling a text classification task one more important key component is the *last layer activation*. The activation function constrains the network's output. For a text classification task the predicted values should be between 0 and 1. The most

popular activation functions for this purpose are sigmoid, relu and softmax, where the sigmoid function is the most suitable for a multi label text classification problem.



**Figure 2.2:** Anatomy of a neural network (Original figure from [32])

Deep learning has a crucial impact on natural language processing and the deep models have become the new state of the art methods for NLP problems [63]. Long Short Term Memory networks offered improvements on machine translation [60], [38] and language modeling [43]. Convolutional Neural Networks used for sentence classification [45] or for extracting sentiment, emotion and personality features for sarcasm detection [54]. Transformers have been used for document encoding and summarization [65].

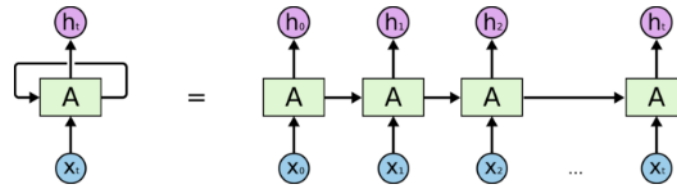
### 2.3.1 Recurrent Neural Networks

The Recurrent Neural Network (RNN) is a neural sequence model that achieves state of the art performance on tasks such as language modeling, sequential labeling or prediction [64]. RNNs are recurrent in nature as they perform the same function for every input data while the output of the current input depends on the hidden state from the previous time step as well. This means that, in other neural networks all the inputs are independent of each other, but in the case of RNNs, the inputs are related to each other through hidden states.

#### 2.3.1.1 Model

An RNN model consists of an input layer, a hidden layer and an output layer. The aspect that makes this network different from a vanilla artificial neural network is the fact that it is time dependant. This means that the input vector to every time step is given by concatenating the input at the current time step with the output of the neuron from the previous time step. For example, as shown in Figure 2.3, the model takes  $X_0$  from the input sequence and outputs the hidden state  $h_0$ . This state  $h_0$  along with  $X_1$  is the input for the next step [13]. This process happens similarly in the next step with  $h_1$  and  $X_2$  being the inputs for the next step and so

on. This gives RNNs the ability to understand relationships between consecutively occurring words in textual data and relate them to one or more classes [52].



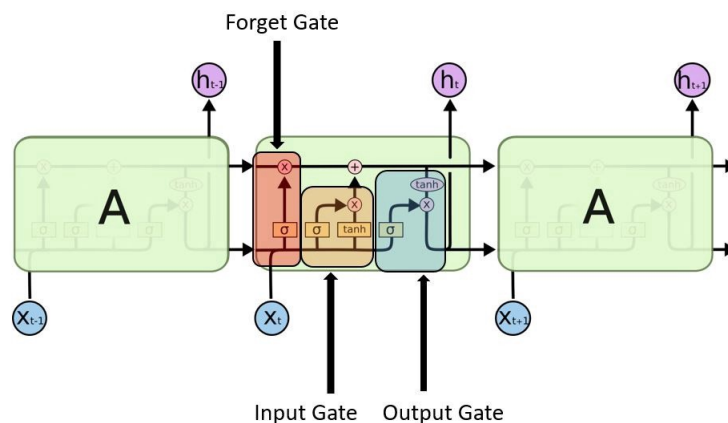
**Figure 2.3:** An unrolled Recurrent Neural Network (Original figure from [13])

### 2.3.1.2 Variants

Based on their architecture and working, a few variants of RNNs are listed below.

#### Long Short Term Memory

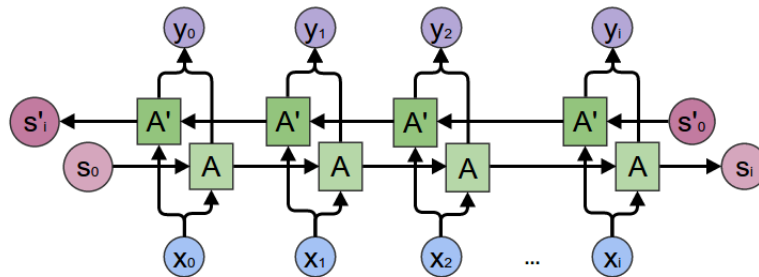
Long short term memory is a modified version of the Recurrent Neural Networks. Architecturally, the LSTM consists of three gates, input gate, output gate and forget gate. The input gate governs the amount of importance or "weight" that is to be given to the input sequence at a particular time step. These weights are the result of a *tanh* function so they are in the  $[-1,1]$  range. The forget gate then considers the previous hidden state ( $h_{t-1}$ ) and the current input ( $X_t$ ) and decides what to omit and what to keep for each number in the previous cell state ( $C_{t-1}$ ). This is done using a *softmax* function which gives out values in the range 0 (omit) to 1 (keep). The input to the current time step and the memory block, that have been weighed accordingly are then used to decide the output for that particular step [40]. These relationships can be seen from Figure 2.4.



**Figure 2.4:** LSTM architecture (Original figure from [13])

### Bi-directional LSTMs

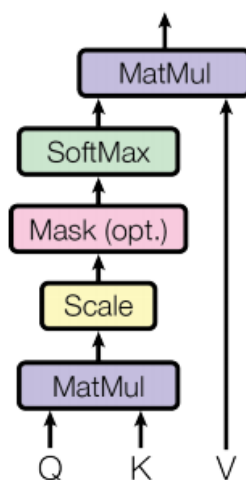
In the field of text classification, having access to the right "environment" for each embedding from the input text is crucial. Using Bidirectional LSTM gives the model access to embeddings that are present to the left and right (past and future) of the current word in question. This helps the model comprehensively understand the relationships between the words in question. This is then the basis by which the model is able to distinguish between input texts from different classes [39].



**Figure 2.5:** Bidirectional LSTM architecture (Original figure from [2])

### Attention

A model's ability to understand and remember feature and word relationships could be further boosted using the concept of attention. Attention refers to mapping a set of "queries" (or input features) to their corresponding sets of "keys" and "values" (vectors that contain information about the related or neighboring features from the text input). The procedure involves the dot product of the input query with the existing keys and then a softmax to give out a "scaled dot product attention score" as Figure 2.6 displays. This obtained score vector is then multiplied by each value and the summed up to give out the final self-attention outcome for that particular query. Interpreting this outcome helps the model understand how much "attention" it should be paying to each word/feature in the block of text [61].



**Figure 2.6:** The attention mechanism (Original figure from [23])

### Gated Recurrent Units

Gated Recurrent Units (GRUs) are improved versions of standard recurrent neural networks [56]. They are fairly similar to LSTMs in their architecture and functionality, albeit simpler. These models use update gates and reset gates to govern what data from previous time steps to keep and what to pass through. This means that GRUs can be trained to keep information through a lot of time steps. The Update gate tells the model about how much of the information from the previous time step needs to be passed on to the next step. The reset gate however, tells the model about how much of the past information needs to be forgotten. This structure can be seen from Figure 2.7. The hidden state from the previous time step ( $h_{t-1}$ ) is considered based on the update and reset gates to determine what information is omitted and what is carried over to the next time step.

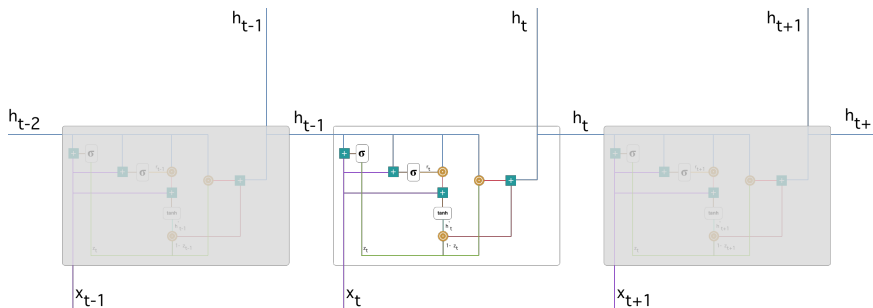


Figure 2.7: Gated Recurrent Units (Original figure from [6])

## 2.3.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a branch of deep neural networks and they are a version of multilayer perceptrons. Multilayer perceptrons are fully connected networks, which means that each neuron in one layer is connected to all neurons in the next layer. CNNs have been used widely for image classification tasks or image and video recognition. One example of a deeper network is the Visual Geometry Group network (VGG-net) which has been invented at Oxford University in 2014 and achieved very good performance on the ImageNet dataset [58]. However, CNNs have been also used for NLP tasks. In paper [44] the researchers exploited a CNN for the semantic modelling of sentences, by introducing a dynamic convolutional neural network. Moreover, a group of researchers [66] trained very deep CNNs to add more expressive power and better generalization for end-to-end automatic speech recognition models. They achieved an 8.5% improvement over the best published result. Another group of researchers presented the implementation of a CNN text classifier and how to integrate it to a question answering problem [25].

### 2.3.2.1 Model

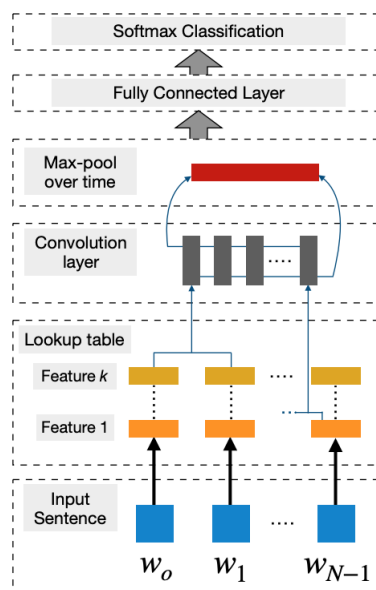
A convolutional neural network is a grid-like topology [48]. The CNN consists of multiple convolutional and pooling layers, followed by one or several fully connected



layers. The current neuron (layer) accepts as input, a subset of neurons of the previous layer. This strategy allows the CNNs to retrieve more abstract representations from the lower layer to the higher layer. Some key components for a convolutional neural network are:

- *Convolution*: This is an operation which finds window of fixed weights for the given element (image or text), where an output unit (pixel or character or subword) produced at each position is a weighted sum of the input units covered by the window.
- *Pooling*: This operation computes a specific norm over small regions on the input. It aggregates small pitches of units (pixels or characters) and thus downsamples the element (image or text) features from the previous layer. The most commonly used pooling operation in CNNs is max-pooling.

In 2008, researchers used CNNs for predictions tasks in NLP such as part of speech tags, name-entity tags and language model [33]. Specifically, they employed a look-up table in order to transform each word into a vector as Figure 2.8 illustrates. The specific approach, using a look-up table, can be considered as a word embedding method whose weights were learned during the training phase of the network [63].



**Figure 2.8:** CNN framework (Original figure from [63])

### 2.3.2.2 Variants

Convolutional Neural Networks have been used widely in text classification tasks in various ways.

#### Very Deep CNN

Facebook AI developed a very deep convolutional neural network for multi-class text classification [34]. Specifically, they presented the new architecture of a very deep CNN which relies on two design principles: 1) it uses the lowest atomic representation of text, which means that it uses characters and 2) it employs convolutions

and max-pooling operations. They tested their architecture on several open source large-scale data sets and they were able to improve the performance by increasing the depth up to 29 convolutional layers.

In order to evaluate their method they used various date-sets. They experimented with the size of the data-set, the number of classes and also with the classification task. They compared their results with the best published results and they noticed that they achieved state of the art results for the most of the data sets. Indicative, for a data set of 3650k, 5 number of classes on the sentiment analysis task they achieved an error value of 37.00 while the previous best published result was 40.43. For another data set case of size 1460k, 10 classes on topic classification task their approach result with an error value of 26.57 against the previous one which was 28.26.

### CNN-RNN

Another group of researchers presented another approach which is an ensemble application of convolutional and recurrent neural networks for multi label text categorization [31]. They implemented a CNN-RNN architecture to model the global and local semantic information of texts, and then they utilize label correlations for prediction. Their approach consists of two parts: 1) the CNN part which extracts text features and 2) the RNN part which is responsible for multi label prediction. Moreover, before training the whole CNN-RNN model, they pretrained a word2vec model for capturing the local features for each word and feed it in the CNN-RNN training.

Through their experiments they ended up that their method depends on the size of the training data set. They found that if the data set is too small then the model overfits, but in case of a large scale data set their approach can achieve state of the art performance. They used two publicly available data sets, Reuters-21578 with 90 labels and RCV1-v2 with 103 labels. They compared their approach with other algorithms such as binary relevance, classifier chain, MLkNN and ML-HARAM over the above mentioned data sets. They observed that for the RCV1-v2 data set, their method outperforms any other method. They achieved a macro-average F1 of 0.712 where the previous best was 0.687 and had been achieved by the method binary relevance.

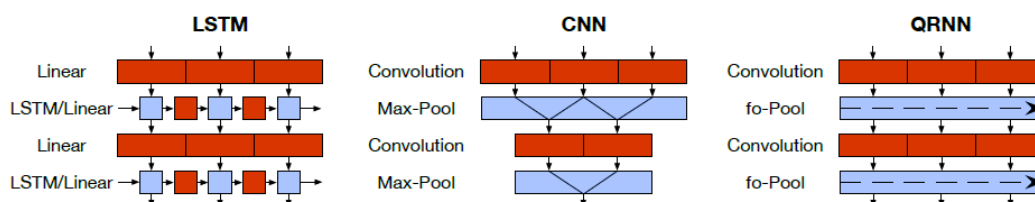
### 2.3.3 Quasi-Recurrent Neural Networks

Recurrent neural networks are a powerful tool but they have a limitation: each timestep's computation is depended on the previous timestep's output. This dependency doesn't allow parallelism. A group of researchers presented the quasi recurrent neural networks (QRNNs) for neural sequence modeling, which is a model that exploits the advantages of CNNs and RNNs. That means that QRNNs allow parallel computation across both timestep and minibatch dimensions, as CNNs. Moreover, QRNNs allow the output to depend on the overall order of elements in the sequence like RNNs. Exploiting parallelism and context, QRNNs have better predictive accuracy than LSTM based models on language modeling, sentiment classification, and character level neural machine translation tasks [29].

### 2.3.3.1 Model

In a quasi recurrent neural network, each layer consists of two subcomponents. Those components are related to convolution and pooling layers in CNNs. The convolutional part allows fully parallel computation across both minibatches and spatial dimensions. The pooling component doesn't make use of the trainable parameters and allows fully parallel computation across minibatch and feature dimensions. A single QRNN layer employs an input dependent pooling, followed by a gated linear combination of convolutional features. Similarly to CNNs, multiple QRNN layers could be stacked to create a more complex model.

Figure 2.9 displays the computation structure of the QRNN in comparison with LSTM and CNN architectures. The red color implies convolutions and the blue color parameterless functions that operate in parallel along the channel/feature dimension. A continuous block means that those computations can proceed in parallel [29].



**Figure 2.9:** Computation structure of the QRNN (Original figure from [29])

### 2.3.3.2 Variants

Quasi recurrent neural network was the base for some other models in the field of the text analysis.

#### Encoder-Decoder

An extension of the QRNN model is the encoder-decoder model [29]. This architecture uses a QRNN as encoder and a modified QRNN, enhanced with attention, as decoder. By feeding the last encoder hidden state (the output of the encoder's pooling layer) into the decoder's recurrent pooling layer would not allow the encoder state to affect the gate or update values that are provided to the decoder's pooling layer. This fact would substantially limit the representational power of the decoder. Instead, the output of each decoder QRNN layer's convolution functions is supplemented at every timestep with the final encoder hidden state. The QRNN encoder-decoder architecture used for machine translation experiments.

#### Multi-fit

At Fast.ai, they presented an approach for exploiting QRNNs. Initially, Universal Language Model Fine tuning (ULMFiT) is a transfer learning method which can be used for NLP tasks [41]. ULMFiT outperforms the state of the art on various text classification tasks (sentiment analysis, question classification, topic classification), reducing the error by 18-24% on the majority of datasets. Recently, in the paper [36], the authors proposed the Multi-lingual language model Fine-tuning (MultiFit)

for training and fine tuning language models efficiently.

The MultiFit model combines the ULMFit, with QRNNs and subword tokenization. The ULMFit model is based on a 3-layer AWD-LSTM (Averaged SGD Weight-Dropped LSTM) model, which is an approach that uses a DropConnect mask on the hidden to hidden weight matrices, in order to prevent overfitting across the recurrent connections [51]. The creators of the MultiFit replaced the LSTM with a QRNN to achieve a faster training. The MultiFit model evaluated on cross-lingual classification datasets and the researchers found that their model outperforms the models LASER, architecture for multilingual sentence representations in 93 languages which uses a BiLSTM encoder [26] and multi-lingual BERT, a language representation model based on the transformer encoder model [35]. Specifically, for the document classification task, MultiFit achieved the best accuracy in comparison with Laser and MultiBert over seven languages. Indicatively, for the Spanish language, Laser achieved an accuracy of 88.75, MultiBERT 95.15 and MultiFit 96.07.

### 2.3.4 Transformers

Transformers are machine learning models that use a combination of stacked encoders and decoders which use the concept of attention [61]. This is implemented in the 'multi-head attention' stage of the model. This gives the model the ability to focus on positions in the environment of the token in consideration. It also gives the model multiple representation subspaces. As each of these attention heads are initialised randomly, each of the input embeddings is projected on to multiple representational subspaces. These models have the ability to understand word relationships very well due to the presence of hidden states at each stage of encoding and decoding along with the use of multi headed attention.

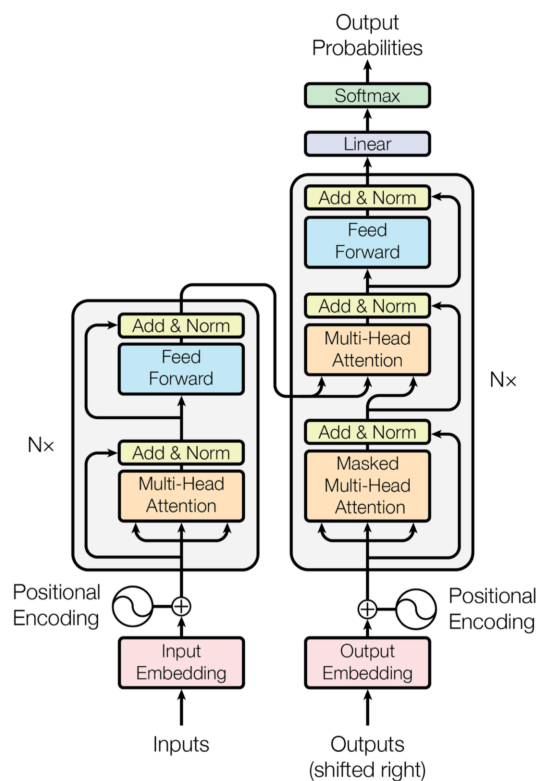
#### 2.3.4.1 Model

A brief layout of the transformers model is show in Figure 2.10. The encoder blocks can be seen on the left and decoder blocks on the right. The input sequence is first embedded using a system called ELMo. ELMo is a contextual word embedding system that has been trained on a large multilingual data set so it embeds grouped words closely together. The embedded input sequence is first passed through the multi-head attention. This part of the model consists of a stacked attention layers that uses the query-key-value system to understand what embeddings to give more "attention" to for each word. Once these scores are obtained, they are passed through the feed forward section of the encoder to obtain the hidden state for that particular encoder. That hidden state is then passed over to the decoder which uses a similar system of multi headed attention to check for embedding relationships and to use a final *softmax* layer for an output probability distribution.

#### 2.3.4.2 Variants

The transformer architecture based model, BERT, is an encoder only model that uses a combination of concepts such as Feature extraction, ELMo (Embeddings from Language models) and masked language modelling [35]. The first stage of

this model involves the multi-lingual embeddings of the words in the input text to obtain features. These features are then passed on to the stacked multi-layer encoder (which are equipped with multi-head attention) architecture to understand and build feature relationships and dependencies. Once the combination of various types of features have been built, this data is then passed on to an output linear layer with a *sigmoid* function to give out the probability distribution for each class. The embedder and encoder sections of the model have been pre-trained where as the final linear classification layers are added on according to the requirements of the classification. The transfer training can be done by either freezing the embedder and encoder sections while training the classifier weights or by fully retraining the classification weights for the entire model (including the embedder and stacked encoder sections).



**Figure 2.10:** Structure of a transformer (Original figure from [23])

## 2.4 Evaluation and Metrics

A classifier which has been employed for a multi label text classification task might predict all the expected labels, a subset of them, or none of the expected labels. Hence, those cases should be considered in order to evaluate the classifier. The metrics for a multi label text classification problem are organized in two main categories which are example based and label based evaluation [59].

**Example based evaluation**

Consider a defined experiment with P positive instances and N negative instances for some condition. The four predicted outcomes can be classified into True Positives (tp), False Positives (fp), True Negatives (tn) and False Negatives (fn) as the Table 2.1 explains.

	Relevant	Non relevant
Retrieved	true positives (tp)	false positives (fp)
Not retrieved	false negatives (fn)	true negatives (tn)

**Table 2.1:** Confusion matrix (Original table from [50])

The example based evaluation metrics that are being used are then defined as:

- Recall: The percentage of predicted true labels to the total number of actual true labels, Equation 2.1.
- Precision: The percentage of correctly predicted true labels to the total number of predicted true labels, Equation 2.2.
- F1-Measure: The harmonic mean of recall and precision, Equation 2.3.
- Hamming loss: The hamming loss is the fraction of labels that are incorrectly predicted [8], Equation 2.4.

Higher value of accuracy, precision, recall and F1- score, means better performance of the learning algorithm. The hamming loss metric, as its name declares, is a loss function so a lower value implies better performance.

$$Recall = \frac{tp}{tp + fn} \quad (2.1)$$

$$Precision = \frac{tp}{tp + fp} \quad (2.2)$$

$$F1 = 2 \frac{Precision \times Recall}{Precision + Recall} \quad (2.3)$$

$$HammingLoss = \frac{1}{|N| \cdot |L|} \sum_{|N|}^{i=1} \sum_{|L|}^{j=1} xor(y_{i,j}, z_{i,j}) \quad (2.4)$$

where  $y_{i,j}$  is the target,  $z_{i,j}$  is the prediction, N is the number of classes and L the number of the instances.

**Label based evaluation**

Label based measures evaluate each label separately and then averages over all labels. All the measures from the example based evaluation can be used for label evaluation.

- Micro averaged measures: Any of the example based evaluation metrics can be computed on individual class labels first and then averaged over all classes.
- Macro averaged measures: Any of the example based evaluation metrics can be computed globally over all samples and all class labels.

- Weighted average measures: Any of the example based evaluation metrics can be computed on individual class labels first and then averaged over all classes with their corresponding class weights. These class weights are given by the distribution of the test data.

## 2.5 Summary

The described models have been chosen as they are relevant to this particular problem of this master's thesis. A model which utilizes tf-idf vectorization and relies on support vector machines will be developed as a baseline model. Gated Recurrent Units with attention seems a safe approach for the specific problem. As the Convolutional Neural Network is based on the character level, the language of the input data will not be relevant in this case and the specific network sounds a promising model. Furthermore, a Quasi Recurrent Neural Network based model will be developed since it enhances the performance. The Transformers model with the Bert multilingual encoder, negates the problem about having multilingual input data and therefore its performance will be evaluated on the given data set.

A major key of the email multi label classification task is the evaluation of the models. The precision aims to answer the question, what proportion of positive identifications was actually correct, while the recall tries to answer, what proportion of actual positives was identified correctly [3]. The F1 measure metric compromises between recall and precision so it is a good candidate. Macroaveraging metrics give equal weight to each class, whereas microaveraging metric give equal weight to each sample classification decision. Microaveraged metrics are a measure of effectiveness of the large classes in a test set. The effectiveness of the small classes can be represented better by the macroaveraged metrics [50]. Therefore the metrics that will be used for evaluating the results of each classifier will be precision, recall and F1 measure. The comparison of the performance among all models will be rely on the metrics weighted average F1, since the data set is unbalanced, and hamming loss.

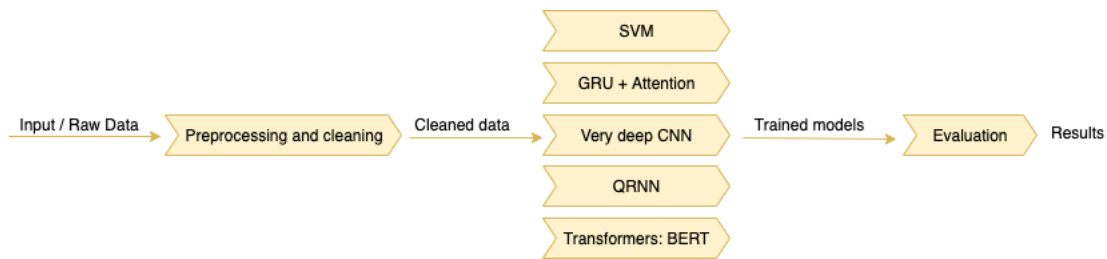




# 3

## Methods

A flowchart representing the steps that are being followed in this project is shown in figure 3.1. The flowchart contains the SVM model, which acts as a baseline model, as well as four deep learning models. The procedure starts by preprocessing and cleaning the input data. After that, five different models (classifiers) will be trained on the data. The 90% of the data set has been used as training and validation data set, while the rest 10% has been used for testing. Then the trained models will be evaluated before delivering the results.



**Figure 3.1:** Pipeline flowchart

### 3.1 Preprocessing

The preprocessing of the input data is done in various stages. Initially, the obtained data set is grouped according to classes that have similar information and this brings the number of labels down from 107 to 14. The first of these stages then involves the removal of confidential information. Data such as names, numbers, web IDs, company names and mail ids were replaced with tags before the data was handed over. These tags have to be removed first. This is then followed by the removal of escape sequences, special characters, forwarded messages and other numbers. After this, greetings and brevity signatures (in Swedish and English) are removed. The obtained text is then trimmed to remove excess spaces. Next, the stop words are removed for both languages. This is done by creating a CSV file with the stop words that are to be considered and then searching for, and removing these words in each data point in the given data set.

The next step involves the stemming or lemmatization of the data. Both these procedures are done in a word by word fashion as they are language specific. Stemming is done by detecting the language of each word and applying the corresponding stemmer using the class *SnowballStemmer* [11] from the *nltk* [10] library. Lemmatization is done by first detecting the language of each word using the *langdetect*

[17] library. The English words are then taken back to their lemma using the *WordNetLemmatizer* [46] from the *nltk* library. The Swedish words are lemmatized by comparing and substituting each word found to a CSV file consisting of words and their corresponding lemmas. The next stage of preprocessing is to remove all the proper nouns in the text. This is done using a combination of a *postagger* and a check to see if the words that are tagged as proper nouns are not actually words that are existent in the Swedish or English language lexicon (from the corresponding word lists).

Finally, the subject and the body of the email have been merged into one text sample. Usually, the subject of the email contains contextual information which enhances the label prediction.

## 3.2 Models

The output of the preprocessing and cleaning data procedure will be the input data set for the training models. One baseline model, SVM, and four deep learning models have been employed for classifying the emails and comparing the results and the performance among them.

### Support vector machines

A basic machine learning model has been built for the multi label text classification task. This model uses the tf-idf tokenizer in order to exploit the weight of the interesting and meaningful terms, which contribute to label prediction. Then, the model employs the *OneVsRestClassifier* [12], which fits one classifier per class. The decision function for the classifier has been set to *LinearSVC* as it is a fast approach [9].

### Gated recurrent units with attention

The model is built using layers of stacked Gated Recurrent Units and self attention with alternating dropout layers in between. This is then followed by a linear layer with fourteen units, each for one class, and finally a sigmoid layer to give out the probability for each class. This model is then trained on the cleaned input data and tuned accordingly to improve performance. It is then evaluated on the test data based on the metrics and the results are noted.

### Very deep convolutional neural networks

The very deep convolutional neural network (VDCNN) is a character based model. The alphabet consists of the English characters, which are common also in the Swedish alphabet, and the Swedish special characters like å, ä and ö. Each sample is converted in a vector where each character of the sample has been replaced by its index in the alphabet. The vectors are padded with zeros so that they have the same length. The network contains 9 convolutions layers and 3 pooling operations. The prediction part is handling by a *k*-max pooling function, which extracts the *k*-highest activations from a sequence. The results of the *k*-max pooling function is the input to a three layer fully connected classifier with ReLU hidden units and sigmoid outputs. The very deep CNN model utilizes a Stochastic Gradient Descent

(SGD) optimizer with momentum [16]. The specific optimizer has been selected because it increased the efficiency for the specific combination of the model, data set and problem.

### Quasi recurrent neural networks

The QRNN uses words as tokens and it is a really simple model. Specifically, it consists of a SpatialDropout1D, a QRNN and a Dense layer while the last activation layer is a sigmoid activation. The SpatialDropout1D layer [18] drops entire 1D feature maps instead of individual elements. The optimizer for this network has been set to RMSprop, which implements the RMSprop algorithm [14].

### Transformers

The transformers model is built using the pre-trained twelve encoder (BERT Base) stack and an additional dense layer and sigmoid activation function to obtain the probability distribution for the classes as an output. Initially the pre-trained weights of the model are frozen with only the final layer weights unfrozen and the model is trained on the training dataset. The model weights are then unfrozen entirely and the model is then trained on the cleaned and pre-processed training set. It is then evaluated on the test set and the results are noted.

The Table 3.1 presents an overview of some key components of the architecture of the deep learning models.

Model	Token	Last layer activation	Optimizer
GRU	byte-pair	sigmoid	SGD
Very deep CNN	character	sigmoid	SGD
QRNN	word	sigmoid	rmsprop
Transformers	byte-pair	sigmoid	SGD

**Table 3.1:** Architecture overview of deep learning models

## 3.3 Evaluation

Once the training procedure has been completed the next step is to evaluate the classifier. After the training procedure each classifier predicts the output on the test data set by producing the *classification report*. The classification report is a text report which shows the main classification metrics like precision, recall and so on for each class. This is a mere task which is performed by calling the classification report function from sklearn [4]. In addition, the *hamming loss* for each classifier is computing by using the homonym function from sklearn [8].



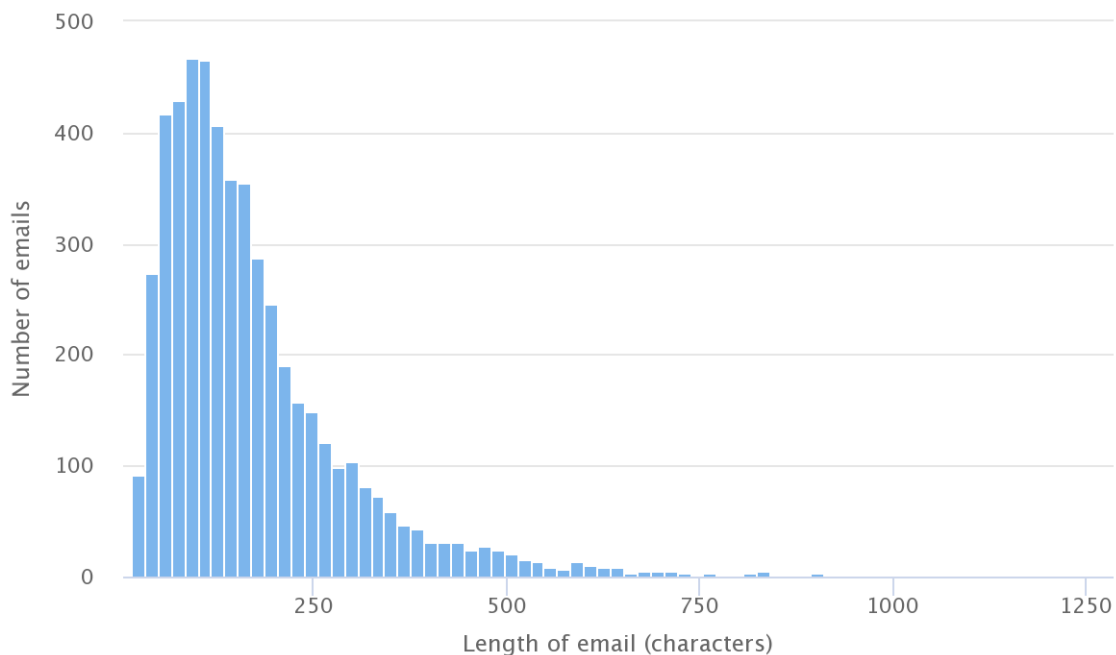
# 4

## Results

This chapter includes a detailed analysis of the data set as well as a comprehensive presentation of the results that have been delivered by the developed models.

### 4.1 Dataset

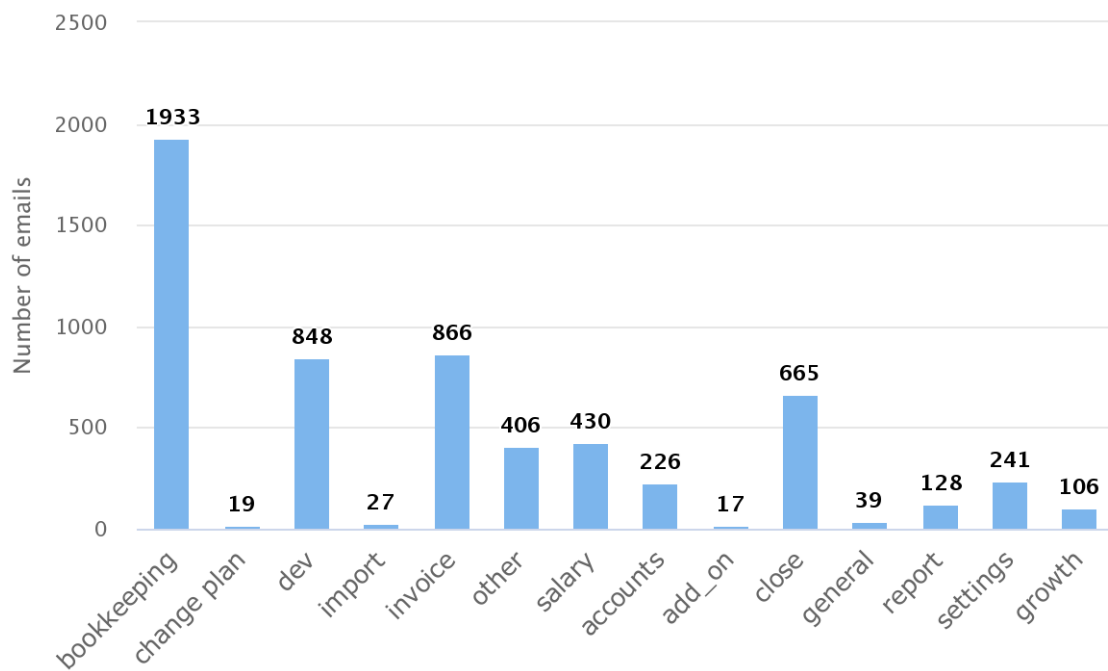
The results from the exploratory data analysis have been shown in figures 4.1 , 4.2 and 4.3. The training data set consists of a total of 5272 emails. Figure 4.1 gives information about the number of characters per input data point as a histogram. According to the histogram the average length of the emails is approximately 370 characters for the majority of the emails. The Figure 4.2 gives an idea about the classwise distribution of data across the input data set. The majority of the emails belongs to the class bookkeeping, while the classes invoice, dev and close are following. Latest Figure 4.3 displays the number of emails which have multiple labels. The vast majority of the emails has only one label and a small portion has two labels.



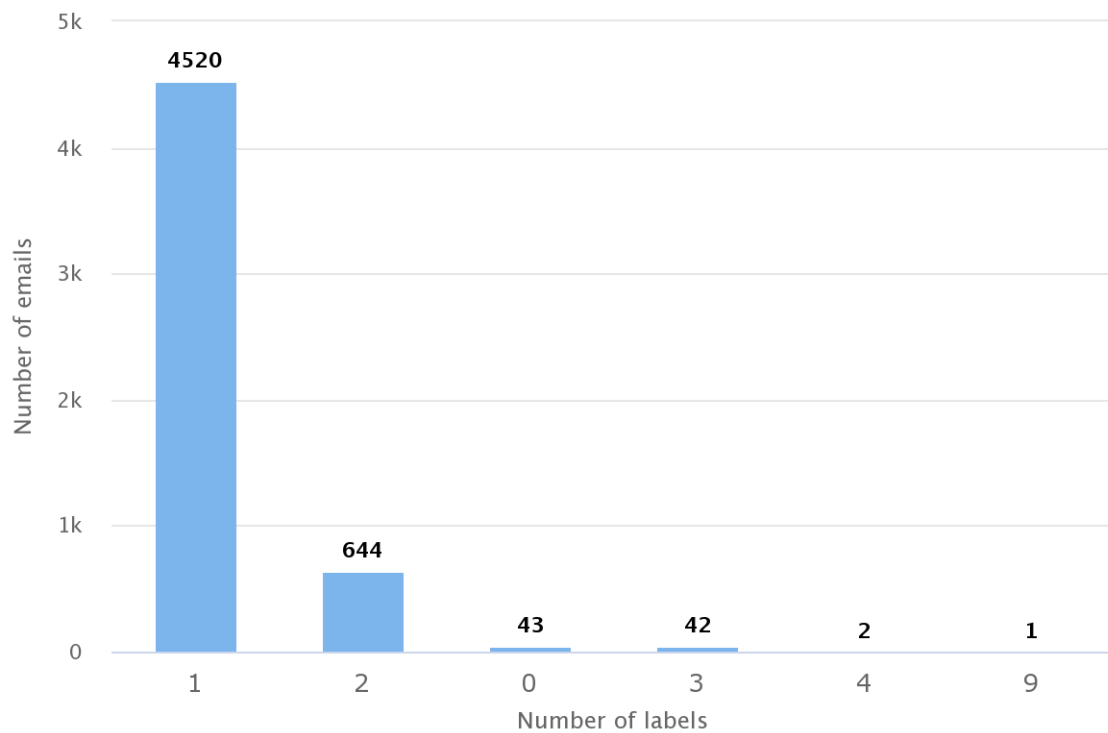
**Figure 4.1:** A histogram demonstrating the length of the emails

## 4. Results

---



**Figure 4.2:** Email distribution across classes



**Figure 4.3:** Emails with multiple classes

### Lingual distribution of emails

Out of a total of 5272 mails, the lingual distribution in the data set is as follows:

- Swedish and English: These emails make up a majority of the data set and they contain both Swedish and English words in the same mail. There are 4973 emails in this category.
- Swedish: These emails consist of only Swedish words. There are 202 emails in this category.
- English: These mails consist of only English words. There are 96 emails in this category.

### Description of the content of the classes

A brief description of the contents of emails in each of the classes is shown at the table 4.1. Moreover, the Appendix A presents some email samples that belong to the below classes.

Class	Description
bookkeeping	emails from customers that have queries or doubts about the bookkeeping services that are offered by the company
change plan	emails that are from customers that request a change in subscription plans and renewals of services
dev	queries or feedback from customers about the actual functionality or usage of the services and doubts about factors like login information and mistakes while reporting or updating information
import	emails that have customer queries about importing invoices and the settings relating to the same. These emails also deal with questions about errors while importing e-invoices
invoice	emails from this class deal with customer queries and questions about invoices in general
other	emails that are received by the company that cannot be categorised into any of the classes belong to this class
salary	queries from companies and organizations that are based on salaries of the employees and their relevant tax benefits
accounts	emails in this category deal with customer feedback and suggestions about the functioning of the accounting systems by the company
add on	emails that are relevant to the addition of services which are related to account and billing
close	emails about the closure and termination of particular services that are offered by the company
general	generic queries about accounting and services.
report	emails deal with the queries about balance reports and verification
settings	emails in this class deal with the queries about settings of the accounts of customers
growth	emails in the growth class deals with problems and shortcomings with the support that has been reported as feedback from customers

**Table 4.1:** Classes description

## 4.2 Model Performance

The next sections present the results, by including the classification report and hamming loss, as they have been delivered by each classifier.

### 4.2.1 Support vector machines

The SVM model delivered a hamming loss equals to 0.0077. The table 4.2 presents the classification report for this baseline model. As it can be concluded by the table, the efficiency of this model is high. The OneVsRestClassifier which has been used for this model employs one classifier per class. Since each class is represented by one and one classifier only, it is possible for each classifier to gain knowledge and draw the contextual window for each particular class.

Class	Precision	Recall	F1-score	Support
bookkeeping	1.00	1.00	1.00	46
change plan	1.00	1.00	1.00	19
dev	0.97	0.97	0.97	36
import	1.00	0.95	0.98	22
invoice	1.00	0.90	0.95	30
other	1.00	0.97	0.99	37
salary	1.00	0.96	0.98	23
accounts	1.00	0.92	0.96	25
add_on	1.00	0.88	0.94	17
close	1.00	0.82	0.90	39
general	1.00	1.00	1.00	23
report	1.00	0.91	0.95	23
settings	1.00	0.83	0.90	23
growth	1.00	0.83	0.91	24
micro avg	1.00	0.93	0.96	387
macro avg	1.00	0.93	0.96	387
weighted avg	1.00	0.93	0.96	387
samples avg	0.97	0.94	0.95	387

**Table 4.2:** SVM classification report

### 4.2.2 Gated Recurrent Units with attention

The Table 4.3 depicts the performance of the Gated Recurrent Unit with attention model. The evaluation has an overall hamming loss of 0.0447 for this model. From figure 4.4, it can be seen that the GRU model overfits the data. This can be attributed to a number of reasons ranging from the distribution of the data across the classes and the nature of the data itself. It can also be seen that despite using class weights to compensate for the unbalanced distribution, the classes with a higher number of samples in the train set (such as bookkeeping, dev and other)



have much better performance than the other classes. It can also be noted that the rather negligible difference between the micro averaged and macro averaged precision scores shows that the major difference in performance due to class imbalances does not affect recall a lot more than it affects precision. The difference between the micro and macro averaged recall is a lot higher, hence, showing that the model more confidently classifies emails that belong to the classes with a larger population.

Class	Precision	Recall	F1-score	Support
bookkeeping	0.94	0.98	0.96	46
change plan	1.00	1.00	1.00	19
dev	0.97	0.89	0.93	36
import	1.00	0.17	0.24	22
invoice	0.77	0.67	0.71	30
other	0.92	0.95	0.93	37
salary	1.00	0.87	0.93	23
accounts	1.00	0.16	0.28	25
add_on	1.00	0.18	0.30	17
close	0.88	0.54	0.67	39
general	1.0	0.87	0.93	23
report	0.86	0.26	0.40	23
settings	1.00	0.22	0.36	23
growth	0.67	0.08	0.15	24
micro avg	0.93	0.61	0.73	387
macro avg	0.93	0.56	0.63	387
weighted avg	0.92	0.61	0.67	387
samples avg	0.74	0.65	0.67	387

**Table 4.3:** GRU classification report

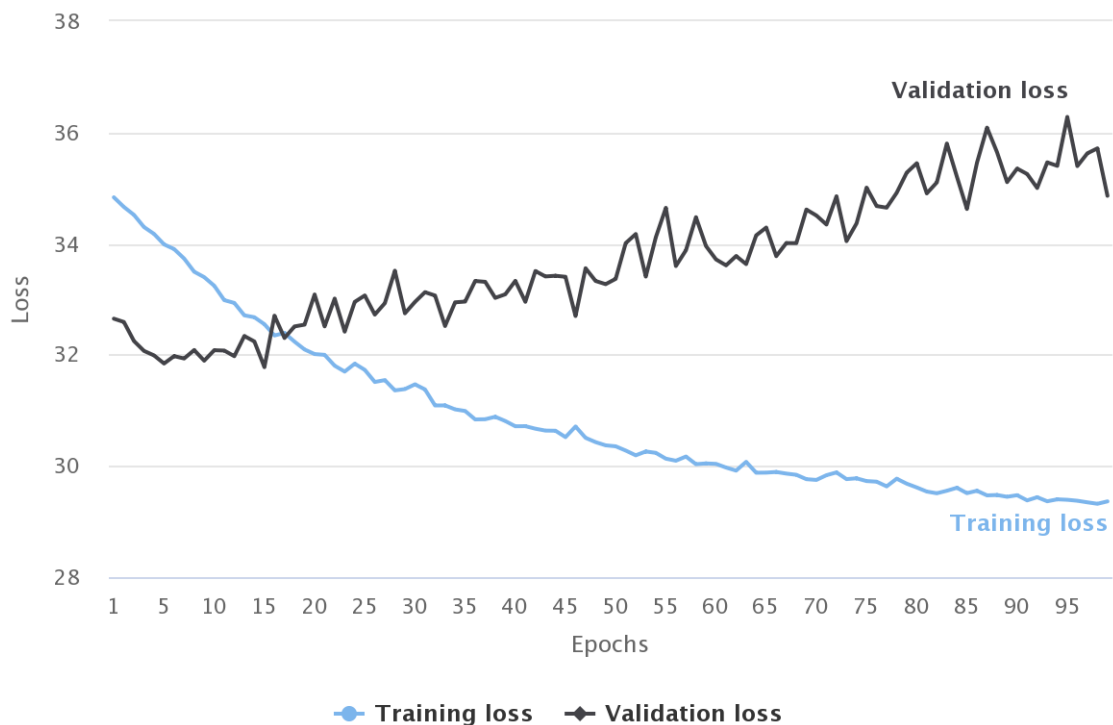


Figure 4.4: GRU training and validation loss

### 4.2.3 Very deep CNN

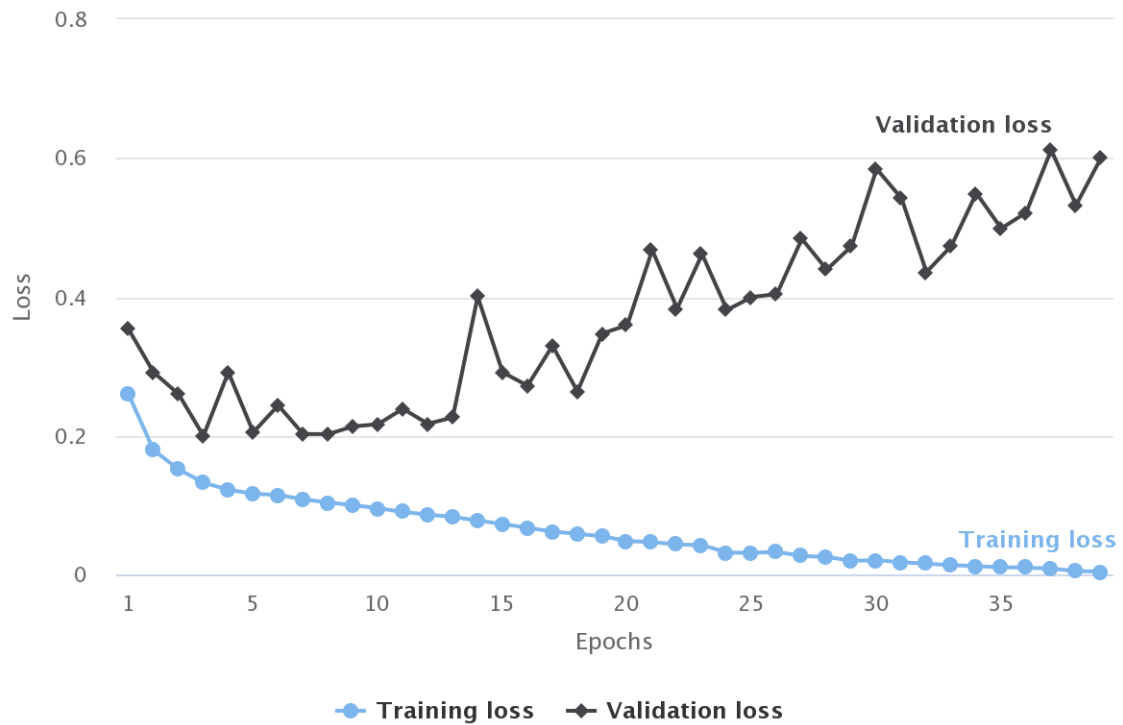
The training of the very deep CNN model ended with a hamming loss equals to 0.0235 and the Table 4.4 illustrates the classification report of this model. Overall the specific model achieved a good performance. Excluding the classes dev, invoice, accounts and growth the rest classes hold a greater or equal precision than recall. This character based model managed to find and learn the representations for the specific problem of the samples, which are containing two languages at the same time. The figure 4.5 displays the training and validation loss function through the 40 epochs which has been used for training this model. In the end of the training the validation loss is almost 0 but the validation loss is 0.6 which means that the model tends to overfit.

This model relies on the research work of the Facebook AI [34] and the contributors show that the performance of this model increases with the depth. They used up to 29 convolutional layers and they reported improvements over the state of the art on several public text classification tasks on large scale data sets. The length of the samples of their data sets varies among them and there was a case where the data set included samples with an average text length of 764 characters. However, for the multi label text classification task on bilingual texts, increasing the depth and using 17 or 29 convolutions layers downgraded significantly the performance of the very deep model. The average text length of this master's thesis data set is around 370 characters which is not far away of the shortest text length that they used. The big difference is the size of the data set. The smallest training data set that they used, consists of 120 thousands examples, while the training data set of this master's thesis is only 5 thousands. However, despite the small size of the data

set the very deep CNN model delivered remarkable results.

Class	Precision	Recall	F1-score	Support
bookkeeping	1.00	0.87	0.93	46
change plan	1.00	1.00	1.00	19
dev	0.92	1.00	0.96	36
import	1.00	0.36	0.53	22
invoice	0.91	0.97	0.94	30
other	1.00	1.00	1.00	37
salary	1.00	0.96	0.98	23
accounts	0.60	1.00	0.75	25
add_on	1.00	0.59	0.74	17
close	0.95	0.95	0.95	39
general	1.00	0.96	0.98	23
report	1.00	0.48	0.65	23
settings	1.00	0.78	0.88	23
growth	0.88	0.96	0.92	24
micro avg	0.92	0.87	0.90	387
macro avg	0.95	0.85	0.87	387
weighted avg	0.95	0.87	0.89	387
samples avg	0.88	0.89	0.88	387

**Table 4.4:** Very deep CNN classification report



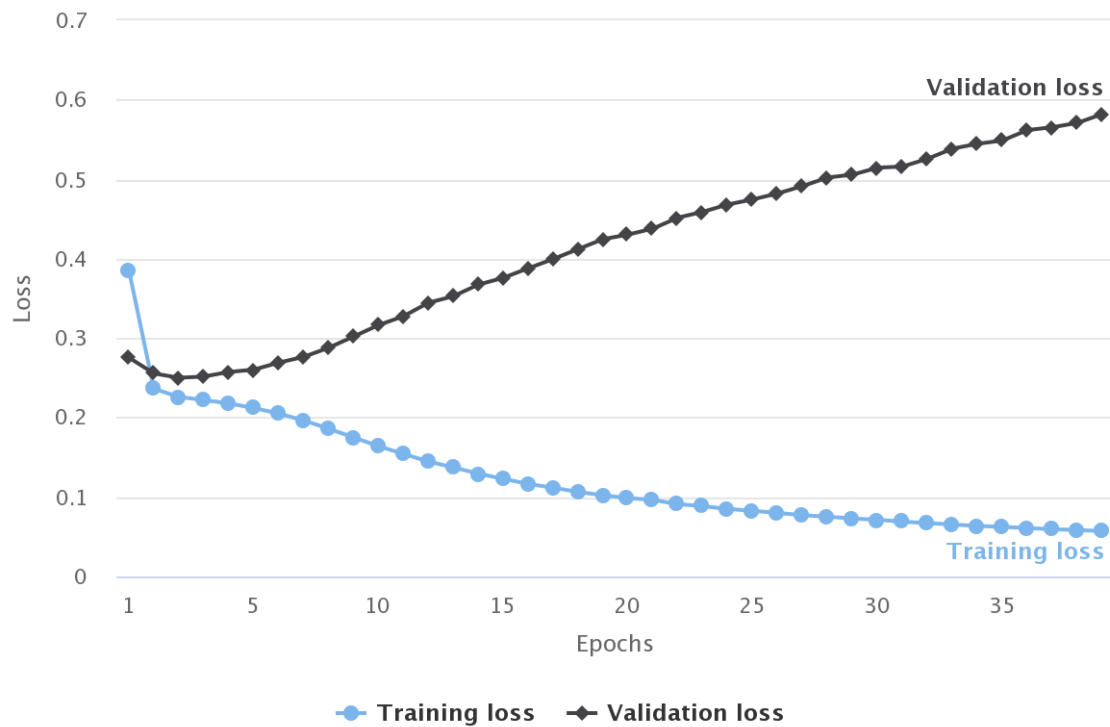
**Figure 4.5:** Very deep CNN training and validation loss

#### 4.2.4 Quasi Recurrent Neural Network

The QRNN model holds a hamming loss of 0.0375 and the Table 4.5 depicts the classification report for the QRNN model. In the QRNN model the recall is greater than the precision for 9 out of 14 classes. Excluding the classes change plan, dev, import, other and add\_on, the rest of the classes hold greater or equal recall rather than precision. The Figure 4.6 illustrates the training and validation loss through 40 epochs. In the end of the training the training loss is around 0.05 while the validation loss is close to 0.6, which means that the QRNN model seems to overfit.

Class	Precision	Recall	F1-score	Support
bookkeeping	0.83	0.98	0.90	46
change plan	0.69	0.58	0.63	19
dev	0.97	0.94	0.96	36
import	0.60	0.27	0.37	22
invoice	0.94	1.00	0.97	30
other	1.00	0.95	0.97	37
salary	0.88	1.00	0.94	23
accounts	0.92	0.96	0.94	25
add_on	1.00	0.18	0.30	17
close	0.84	0.97	0.90	39
general	0.48	0.48	0.48	23
report	0.63	0.83	0.72	23
settings	0.83	0.87	0.85	23
growth	0.55	0.71	0.62	24
micro avg	0.81	0.82	0.81	387
macro avg	0.80	0.77	0.75	387
weighted avg	0.82	0.82	0.80	387
samples avg	0.78	0.80	0.77	387

**Table 4.5:** QRNN classification report



**Figure 4.6:** QRNN training and validation loss

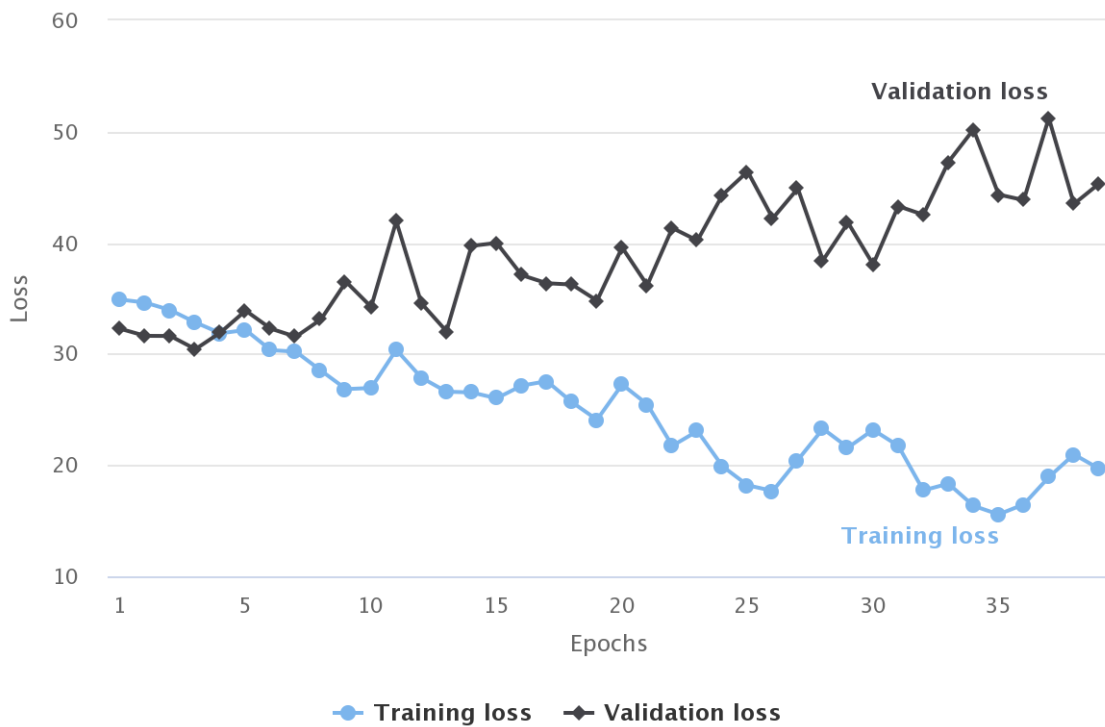
## 4.2.5 Transformers

The Table 4.6 depicts the performance of the BERT Transformers model using transfer learning. The evaluation has an overall hamming loss of 0.0461 for this model. From figure 4.7 it can be seen that the model grossly overfits the dataset. A primary issue is the fact that the pre-trained section of the BERT 12 stack encoder model is fixed in size, which in this case makes the model overfit the input data. A large difference between the micro and macro average in the precision metric also tells us that the BERT model is the most affected by the class imbalances. It can also be seen that the precision is lesser than the recall in the micro, weighted and macro fields, which is in line with the model overfitting. This means that since the model has fit the training data very well it is "falsely confident" about the predictions that are made in the evaluation phase and hence the model tends to make predictions that may not be right, confidently, thereby causing this behaviour.

## 4. Results

Class	Precision	Recall	F1-score	Support
bookkeeping	0.93	0.96	0.95	46
change plan	1.0	1.0	1.0	19
dev	0.15	1.00	0.926	36
import	1.0	0.14	0.22	22
invoice	0.74	0.71	0.73	30
other	0.92	0.95	0.93	37
salary	1.0	0.83	0.91	23
accounts	0.07	0.30	0.12	25
add_on	1.0	0.22	0.36	17
close	0.84	0.87	0.85	39
general	1.0	0.87	0.93	23
report	0.08	0.96	0.15	23
settings	0.10	0.54	0.17	23
growth	0.67	0.08	0.15	24
micro avg	0.24	0.73	0.37	387
macro avg	0.66	0.68	0.51	387
weighted avg	0.63	0.73	0.53	387
samples avg	0.28	0.76	0.38	387

**Table 4.6:** Transformers classification report



**Figure 4.7:** Transformers training and validation loss

# 5

## Conclusion

This chapter concludes this master’s thesis by discussing the methods and results as well as giving ideas for future work.

### 5.1 Discussion

Five different models have been developed for studying the multi label email classification task with the special case of bilingual texts, English and Swedish, at the same time, having to deal also with a size limited and unbalanced data set. One simple model, SVM, and four deep learning models with major differences in their architecture have been employed in order to examine their behaviour and performance in this particular problem. This section provides a discussion for the results that have been delivered from the models, as well as the contribution of the stemming and lemmatization in the final results.

#### 5.1.1 Results comparison among the models

The Table 5.1 presents a summary of the weighted average F1 score and hamming loss among the developed models. The SVM outperforms the four deep learning models. The best deep model is the very deep CNN with a weighted average F1 score of 0.89 against the SVM with 0.96 score. The second best deep learning model is the QRNN with 0.80 weighted avg F1 score.

	SVM	GRU	Very deep CNN	QRNN	Transformers
Weighted avg F1	0.96	0.67	0.89	0.80	0.53
Hamming loss	0.0077	0.0447	0.0235	0.0375	0.0461

**Table 5.1:** Metrics comparison among the models

From the performance tables in chapter 4, it can clearly be seen that the SVM model gives the best results according to the chosen metrics. The main reason is that the SVM uses the OneVsRestClassifier which employs one classifier per class. In this way each classifier is able to learn about word relationships and class associations really good. A known reason which downgrades the performance of the deep learning models involves the size of the data set and the distribution of emails in it. The specific data set is on the smaller side and with an extremely unbalanced distribution.

A comparison between the deep learning models shows that their performance varies a lot among them and their architecture causes a different behaviour. In the

case of the GRU model, the precision is higher (91%) but the lower recall brings the F1-measure down. The very deep CNN delivered the best results among the deep learning models, giving a higher precision. A major factor for its performance is that it uses characters as tokens which proved a real good approach for the specific case. This character based model was able to find good representations, handle the both languages simultaneously and learn sufficiently the relationships between the samples and classes. The QRNN model delivered the second best results among the deep learning models. The QRNN model is using words as tokens and it is a simple model in contrast with the very deep CNN, which delivered the best results, among the deep learning models. Those two models, very deep CNN and QRNN, are greatly differing in terms of architecture perspective, but both of them delivered good results. The CNN employs a convolutional layer while the QRNN utilizes a dense layer. The major difference between those two layers is that the dense layer learns global patterns while the convolutional layer learns local patterns [32] and this is the reason that the CNN model delivered better results than the QRNN model. The performance of the BERT model is the lowest, with respect to F1-measure, precision and recall. This is due to the fact that the byte pair word embedding combined with the relatively large size of the pre-trained section means that the model overfits the data drastically. It is also the most affected by the class distribution of the data set.

In this multi label email classification, the precision represents how many of the predicted labels actually belong to that sample, while the recall represents how many relevant labels from that sample were found. The GRU and the very deep CNN delivered greater or equal precision rather than recall for the most of the classes. Particularly, the precision was higher for the 12 out of 14 classes for the GRU and for the CNN was for the 10 of the 14 classes. That means that the GRU and very deep CNN models were able to predict a big proportion of the positive identifications correctly while they missed some relevant labels. In contrast with the GRU and the very deep CNN, the QRNN model delivered greater or equal recall rather than precision for 9 out of 14 classes. Therefore, the QRNN model found a big proportion of the relevant labels for the samples. For the Bert model the number of classes with higher recall is almost equal to the number of the classes which present higher precision and therefore the Bert model doesn't seem to change drastically the precision and the recall.

Additionally, Chapter 4 shows that the deep learning models overfit during the training, which means that the models learn more than enough information of the training data and so they are not able to make as good predictions on the new data as they delivered on the training data. Preventing overfitting can be done by 1) reducing the network's size (capacity), 2) adding dropout and 3) getting more training data [32]. Reducing the network's size is referred to decreasing the number of the layers. GRU and QRNN models are simple models with only a few number of layers. The CNN is a deep network, however it performs well on the testing data set, therefore the chosen capacity of the network doesn't impair its performance. On the other hand the Bert model is a massively huge model, which might explain why it overfits. By adding a dropout layer in the networks will result in mitigating the noise of the output values, so the network will memorize only the significant patterns which contribute to the label prediction. All of the deep learning models



include at least one dropout layer. Therefore, enriching the training data set would prevent the overfitting.

### 5.1.2 Stemming vs Lemmatization

While using stemming instead of lemmatization as the method of linguistic morphology for the data set, certain differences were noticed in the performance of the models. In terms of weighted average F1 score it noticed that its value was downgraded by 8% for the SVM model, 7% for the CNN and 26% for the QRNN. In the case of BERT and the GRU models, the weighted average f1 score dropped by about 9% on average. One of the reasons for this is the fact that particular classes had a larger population of words such as "general", "organization" and "led", which were wrongly truncated to words that now did not make any sense. This meant that the model was not able to extract the right features from these pieces of text and caused a drop in performance while using a stemmer. To add to this, the QRNN (which has the biggest drop) is a model that uses word embeddings combined with a model that is contextually trying to understand information. This means that the stemmer affecting certain words and putting them out of context has a really large impact on the performance of this model in particular. To an extent, this impact can be seen on the GRU and BERT models as well, as they use sub-word embeddings.

## 5.2 Future work

In terms of future work there are some attractive points which are need further exploration. As it has been already mentioned MultiFit is a multilingual language model, which is currently available in seven different languages, Swedish language is excluded. It employs a QRNN which enhances the performance and according to the creators of the model, the MultiFit performs well on small data sets which contain only 1000 examples and outperforms Bert. Therefore the fact that the specific model outperforms other models even if it has been trained on small data sets, seems promising for the problem of this master's thesis and it would be interesting to study how it performs on this bilingual small data set.

Another experiment with the specific data set and the chosen models it would be the case of enriching the data set before the training procedure. Enriching the initial data set, which is called data augmentation, by an arbitrary way it would result in overfitting due to the extra noise in the data. That's way another approach like semantic enrichment would be interesting and it might improve the performance of the deep learning models.

## 5.3 Conclusion

From the experiments and results that have been obtained after evaluating the pipelines that have been built, the following points can be concluded for the specific problem of this master's thesis: multi label text classification on short text, emails, with two languages, English and Swedish, at the same time and having a size limited

and unbalance data set.

**Simple model performs better.** The simple SVM model performs better than any of the other deep learning models that were evaluated.

**Character based deep learning model performs better.** The very deep character based model delivered the best results among the deep learning models, without a big deviation from the SVM model. In this particular case with Swedish and English input text, the character level encoding performs a lot better than the byte-pair encoding. This could be due to the relatively common nature of the characters present in both languages, in the same data points.

**Depth doesn't improve performance.** The contributors of the paper [34] concludes that the depth improves the performance of the very deep CNN. However in this particular case of the size limited and unbalanced data set the increment of the depth impaired the performance. The major difference between their experiments and those of this master's thesis is the size of the data set. They were working with at least 120 thousands of examples for the training data set against the 5 thousands examples of this master's thesis.

**Lemmatization enhances performance.** Using lemmatization for lingual normalization gives slightly better performance than using stemming does. This difference is noticed only in a few classes due to the occurrence of some words that not dealt with properly by using stemming by need lemmatization.

# Bibliography

- [1] Basic methods of text pre-processing and cleaning. <https://towardsdatascience.com/nlp-text-preprocessing-a-practical-guide-and-template-d80874676e79>.
- [2] Bidirectional lstm. <https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0>.
- [3] Classification: Precision and recall. <https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall>.
- [4] Classification report. [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification\\_report.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html).
- [5] f\_classif feature selection function. [https://scikit-learn.org/stable/modules/generated/sklearn.feature\\_selection.f\\_classif.html](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html).
- [6] Gated recurrent units. <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be>.
- [7] Gmail: Use smart compose. <https://support.google.com/mail/answer/9116836?co=GENIE.Platform%3DDesktop&hl=en>.
- [8] Hamming loss. [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.hamming\\_loss.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.hamming_loss.html).
- [9] Linearsvc. <https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html>.
- [10] Natural language toolkit. <https://www.nltk.org/index.html>.
- [11] Nltk: Snowball stemmer. [https://www.nltk.org/\\_modules/nltk/stem/snowball.html](https://www.nltk.org/_modules/nltk/stem/snowball.html).
- [12] Onevsrestclassifier. <https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html>.
- [13] Recurrent neural networks. <https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e>.
- [14] rmsprop optimizer. <https://keras.io/api/optimizers/rmsprop/>.
- [15] Roc curve. [https://en.wikipedia.org/wiki/Receiver\\_operating\\_characteristic](https://en.wikipedia.org/wiki/Receiver_operating_characteristic).
- [16] Sgd optimizer. <https://keras.io/api/optimizers/sgd/>.
- [17] Spacy langdetect. <https://pypi.org/project/spacy-langdetect/>.
- [18] Spatial dropout layer. [https://keras.io/api/layers/regularization\\_layers/spatial\\_dropout1d/](https://keras.io/api/layers/regularization_layers/spatial_dropout1d/).
- [19] Stemming and lemmatization. <https://www.datacamp.com/community/tutorials/stemming-lemmatization-python>.

- [20] Tensorflow lite: question and answer. [https://www.tensorflow.org/lite/models/bert\\_qa/overview](https://www.tensorflow.org/lite/models/bert_qa/overview).
- [21] Tensorflow lite: smart reply. [https://www.tensorflow.org/lite/models/smart\\_reply/overview](https://www.tensorflow.org/lite/models/smart_reply/overview).
- [22] Text representation. <https://developers.google.com/machine-learning/guides/text-classification/step-3>.
- [23] Transformers. <https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04>.
- [24] N. A. Ahmed, M. A. Shehab, M. Al-Ayyoub, and I. Hmeidi. Scalable multi-label arabic text classification. In *2015 6th International Conference on Information and Communication Systems (ICICS)*, pages 212–217, 2015.
- [25] M. Z. Amin and N. Nadeem. Convolutional neural network: Text classification model for open domain question answering system, 2018.
- [26] M. Artetxe and H. Schwenk. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond, 2018.
- [27] V. Bhatia, P. Rawat, A. Kumar, and R. R. Shah. End-to-end resume parsing and finding candidates for a job description using bert, 2019.
- [28] A. Bouaziz, C. Dartigues-Pallez, C. da Costa Pereira, F. Precioso, and P. Lloret. Short text classification using semantic random forest. pages 288–299, 09 2014.
- [29] J. Bradbury, S. Merity, C. Xiong, and R. Socher. Quasi-recurrent neural networks, 2016.
- [30] W.-C. Chang, H.-F. Yu, K. Zhong, Y. Yang, and I. Dhillon. X-bert: extreme multi-label text classification with using bidirectional encoder representations from transformers, 2019.
- [31] G. Chen, D. Ye, Z. Xing, J. Chen, and E. Cambria. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2377–2383, May 2017.
- [32] F. Chollet. *Deep Learning with Python*. Manning Publications Co., USA, 1st edition, 2017.
- [33] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. pages 160–167, 01 2008.
- [34] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun. Very deep convolutional networks for text classification, 2016.
- [35] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.
- [36] J. Eisenschlos, S. Ruder, P. Czapla, M. Kardas, S. Gugger, and J. Howard. Multifit: Efficient multi-lingual language model fine-tuning, 2019.
- [37] E. Frank and R. Bouckaert. Naive bayes for text classification with unbalanced classes. 09 2006.
- [38] D. Gillick, C. Brunk, O. Vinyals, and A. Subramanya. Multilingual language processing from bytes, 2015.
- [39] A. Graves, N. Jaitly, and A. rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm, 2013.
- [40] S. Hochreiter and J. Schmidhuber. Long short-term memory, 1997.

- 
- [41] J. Howard and S. Ruder. Universal language model fine-tuning for text classification, 2018.
- [42] W. Huang, E. Chen, Q. Liu, Y. Chen, Z. Huang, Y. Liu, Z. Zhao, D. Zhang, and S. Wang. Hierarchical multi-label text classification: An attention-based recurrent network approach. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM '19*, page 1051–1060, New York, NY, USA, 2019. Association for Computing Machinery.
- [43] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling, 2016.
- [44] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences, 2014.
- [45] Y. Kim. Convolutional neural networks for sentence classification, 2014.
- [46] H. Kornilakis, M. Grigoriadou, E. Galiotou, and E. Papakitsos. Using a lemmatizer to support the development and validation of the greek wordnet, 2004.
- [47] E. Kotzé, B. A. Senekal, and W. Daelemans. Automatic classification of social media reports on violent incidents in south africa using machine learning. *South African Journal of Science*, 116(3/4), Mar. 2020.
- [48] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23, 04 2016.
- [49] U. E. Manawadu, T. Kawano, S. Murata, M. Kamezaki, J. Muramatsu, and S. Sugano. Multiclass classification of driver perceived workload using long short-term memory based recurrent neural network. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–6, June 2018.
- [50] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, USA, 2008.
- [51] S. Merity, N. S. Keskar, and R. Socher. Regularizing and optimizing lstm language models, 2017.
- [52] T. Mikolov, M. Karafiat, L. Burget, J. Cernock, and S. Khudanpur. Recurrent neural network based language model, 2010.
- [53] B. Plank, A. Søgaard, and Y. Goldberg. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss, 2016.
- [54] S. Poria, E. Cambria, D. Hazarika, and P. Vij. A deeper look into sarcastic tweets using deep convolutional neural networks, 2016.
- [55] J. E. Ramos. Using tf-idf to determine word relevance in document queries. 2003.
- [56] R. Rana, J. Epps, R. Jurdak, X. Li, R. Goecke, M. Brereton, and J. Soar. Gated recurrent unit (gru) for emotion classification from noisy speech, 2016.
- [57] Z. Ren, M. H. Peetz, S. Liang, W. van Dolen, and M. de Rijke. Hierarchical multi-label classification of social text streams. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 213–222. Association for Computing Machinery, 2014.
- [58] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.
- [59] M. S. Sorower. A literature survey on algorithms for multi-label learning. 2010.

- [60] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks, 2014.
- [61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin. Attention is all you need, 2017.
- [62] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google’s neural machine translation system: Bridging the gap between human and machine translation, 2016.
- [63] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing, 2017.
- [64] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization, 2014.
- [65] X. Zhang, F. Wei, and M. Zhou. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization, 2019.
- [66] Y. Zhang, W. Chan, and N. Jaitly. Very deep convolutional networks for end-to-end speech recognition, 2016.

# A

## Appendix 1

### Examples of emails

Some examples of emails from the data set for each of the classes are shown below.

- bookkeeping
  - hej! jag skall betala en preliminärskatt avs #Number varje månad under #Number . hur bokförs denna?  
med vänlig hälsning, #firstname
  - skulle vilja radera all historik och börja om med fakturametoden #Number
- change plan
  - intäkterna på firman minskar kraftigt i år till väldigt lite pengar , därför försöker jag plocka bort alla löpande kostnader. får jag nytt lyft så uppgraderar jag igen.
  - hej, inget mer samråd med dig.
- dev
  - hej! vill ladda ner momsrapporten, men den sparas i ett format jag inte kan läsa/öppna (.eskd).
  - hej jag försöker lägga till mitt kundregister men namnen kommer inte med hur jag än gör. nu går det inte att ta bort alla på en gång heller. hur ska jag gå tillväga. jag har en excel med alla kunder och har importerat och valt namn, kundtyp osv men det enda som syns efter importen är kundnumret, på namn är det helt blankt så jag kan inte söka upp några kunder.  
hjälp :)  
mvh #firstname
- import
  - hej, jag började använda #compname vid ingången av #Number . har nu fått ett problem med konto #Number som jag fick med mig vid importen men som inte finns uppsatt i #compname . kontot har ett saldo å #Number kr som jag skulle behöva boka bort men det verkar inte som om jag kan boka alls på detta konto. kan ni hjälpa mig så att jag blir av med balansen här?  
hälsningar #customerFName
  - startfråga: går det att importera: kundregister, artikelregister, leverantörsregister, eller annan info och hur gör man detta!? jag har letat men kan ej finna någon info om detta!
- invoice

- jag ser att jag kan fakturera en kund i usa för tjänster men för att det ska klassas som försäljning av tjänster utanför eu och inte läggas på moms så måste jag ha ett giltigt vat-number på kundkortet i #compname . hur gör jag med företag i usa som inte har vat number?
- hej,  
försöker regga mig hos ativo för att kunna sälja ett par fakturor. behöver skicka dem så snart som möjligt, vilket betyder att jag kommer välja en annan fakturatjänst om svaret dröjer för länge.  
får felmeddelande i steget "validera kontouppgifter" efter att jag försökt logga in på seb företag med mitt mobila bankid.  
något ni känner igen?
- other
  - hej, mitt företag är inte momsregistrerat men #compname verkar kräva att jag ska registrera moms. hur får jag bort det?
  - hej jag skulle vilja börja använda #compname helt på ny kula fr.o.m #Number . hur gör jag för att ta bort gamla fakturor etc så att jag kan börja på vitt papper så att säga?  
med vänlig hälsning, #firstname
- salary
  - hej! jag hade tänkt att betala ut min första lön till den #Number :e #Month . tyvärr framgår inte den sociala avgiften av lönespecifikationen i #compname (det står "noll" på sociala avgifter). tack på förhand #customerFName
  - hej. jag har ett problem med lönehandling. efter jag har lagt till lönerna, programmet beräknar inte sociala avgifter utan det står att det är #Number sek som ska betalas på varje lön. skatten stämmer bra, men sociala avgifter visar alltid #Number kr, förra månaden det var inga problem med det.
- accounts
  - hej utgående balans för #Number och ingående balans för #Number stämmer inte, kan ni hjälpa till med att rätta till det? det hade varit en bra funktion i #compname att rätta det själv.
  - jag önskar att man skulle kunna välja år på översikten. nu när årsskiftet passerat kan man inte längre få en bild av #Number där.
- add on
  - hej! jag skulle behöva hjälp av en ekonom som ser över mina utlägg, fakturor och skatt mm. kontaktar jag er då eller är det skatteverket?  
mvh #firstname  
hämta [outlook för ios](#Web
  - hej,  
jag är ingen stjärna på bokföring och lyckas till och med göra fel även med ert jättebra program. jag skulle vilja ha min bokföring genomgången och rättad. ser när jag momsdeklarerar att det är en del som inte stämmer. betalar givetvis för er hjälp.
- close



- jag skulle gärna haft lite hjälp med hur jag ska göra rätt med mitt bokslut för #Number .
- general
  - hej! jag har ett antikvariat och säljer mina böcker via en e-handelsplattform som heter bokbörsen. de tar en avgift för varje beställning jag får. t ex fick jag en beställning där avgiften är #Number .#Number varav #Number .#Number är moms. hur ska jag bokföra den? jag gör insättningar till bokbörsen så att de kan dra av avgiften. vilken kategori tillhör bokbörsen (dvs konto). tack för svar!
- report
  - hej!  
med er nya mall för inbetalning av moms för enskild firma som krediteras #Number (avräkning för...) men detta ledde till att #Number (företagskonto) visar att det har #Number kr (alltså momsen) för mycket i tillgångar.  
vad är det jag har missat? :)  
med vänliga hälsningar,  
#UName
  - hej!  
på mitt bokio står det under "att göra"; att jag har två momsrapporter att stänga.  
vad innebär det och är det bara att stänga eller är det något specifikt jag bör kolla upp?  
tack!  
hälsningar, #UName
- settings
  - hi,  
i have close the year ( \*\*räkenskapsår\*\* ) in wrong date. it suppose to be from #Number -#Number -#Number to #Number -#Number -#Number .  
but it sat \*\*#Month #Number - #Month #Number \*\* which is wrong.  
my company info. :  
#compname #Number -#Number  
e-mail: [#Email ](mailto:#Email )  
how can i open the year ( \*\*räkenskapsår\*\* ) again and set the right year which is from \*\*#Month #Number - #Month #Number \*\*  
thank you in advance,  
#UName
  - hej.  
jag provade ert program med avseende på import av kundregister och fann att det inte gick.  
då jag behöver det så finner jag det onödigt att fortsätta.  
kontot är samma som denna mejl-adress, [#Email ](mailto:#Email )  
tacksam om ni rensar bort alla spår...  
med vänlig hälsning,  
#UName

- growth
  - hej,  
allmänna villkor #Number .#Number punkt #Number .#Number : jag har noterat ett skrivfel under punkt #Number .#Number i ert 'allmänna avtal'. det verkar som om en del text under punkt #Number .#Number har fallit bort.  
om jag förstått det hela rätt, så har #compname rätt att intallera någon slags tredjepartsprodukt på de enheter man använder för att få åtkomst till databasen. det är dock inte skrivet så i ert avtal - för tillfället.
  - hejsan, jag råkade importera fel transaktioner till mitt bokio konto från min bank och man kunde ta bort dem, men de ligger kvar så här? är det något ni skulle kunna hjälpa mig med att få bort så de inte ligger där, så som de gör nu, och sen så ska jag importera in rätt