How to automate the process of classifying emails?

CLASSIFICATION OF EMAILS

June 2, 2020

Belle Bruinsma
2540661
Vrije Universiteit van Amsterdam
`bellebruinsma@live.nl`

Abstract

Large companies receive thousands of emails each day with various types of questions from customers. These companies often employ a support manager who is responsible for assigning emails to the respective teams, which enables a quicker solution for the customer and improves customer satisfaction. However, manually categorizing emails can be time-consuming and require a significant amount of time. Several large software or cloud-based companies offer automated machine learning (AutoML) as a service and can be used for automatic email classification. The performance of AutoML yields already successive results on the task of emails classification. However, this tool is lacking in terms of interpretability and operates as a black box.

This thesis investigates different interpretable machine learning methods to gain an understanding on how to automate the process of email classification. Furthermore, it strives to gain an insight into the method's respective strengths and weaknesses. This thesis focuses on the comparison of the context-free methods: TF-IDF, Word2Vec and FastText (referred to as the traditional methods), and context included methods BERT, XLNet and AutoML (referred to as the contextual methods). The traditional and contextual methods are compared by evaluating the performances on the classification of labeled and unlabaled emails. Various experiments are designed and implemented using the traditional and contextual methods trained on different domains. The results of the experiments showed that the contextual methods outperform the traditional methods on the performance of email classification among almost all evaluation matrices. XLNet yielded the best overall accuracy score; however, AutoML performed best on the minority classes. Furthermore, embeddings trained on a task-specific domain improved the performance of the traditional methods, whereas no improvement was identified on the performance of the contextual methods when deploying different domains.

Determining what method is recommended to companies who aim to implement email classification, depends on considering the importance of two aspects: the interpretability of a method and the performance of a method. Choosing the right method for a company would require striking a balance between an accurate performing method that complies with the preferences of a company and the ability of its users to interpret the methods process.

Contents

1 Introduction

Companies receive many emails each day with different types of questions. Therefore, large companies often employ a customer support manager who is responsible for assigning emails to the respective teams to enable a quicker solution for the customer, which improves customer satisfaction. However, manually categorizing emails can be a time-consuming task since each email must be skim-read before it can be tagged with a specific label. This process can require a significant amount of time when a company receives hundreds or thousands of emails each day. For this reason, large companies require a system that automatically assigns new incoming emails to the corresponding teams.

This thesis focuses on the automation of the process of assigning emails to pre-defined classes utilizing several machine learning techniques. This thesis was written in collaboration with Ciphix, which is an organization that implements robotic process automation solutions for their clients. Moreover, Ciphix is experienced in implementing the process of automatic email classification for their clients and one of the possible solutions for implementing this task is the automated machine learning (AutoML) tool. The AutoML tool is offered as a service that enables the use of high custom machine learning models. However, despite its robustness and ability to accurately classify emails, the tool is lacking in terms of interpretability and operates as a black box, which means that the machine learning techniques are not transparent to the users. Understanding why machine learning models behave in the way they do has become a vital concern in machine learning, and work in the area of interpretable models drawn renewed interest. This thesis explores the use of different interpretable machine learning methods for the task of email classification and provide insights into their respective strengths and weaknesses. The goal is to achieve results that are similar to or even better than those of the AutoML tool.

A machine is unable to understand raw text, meaning that it is necessary to map words to vectors of real numbers before they can be fed as input to an algorithm. These vectors are called word representations and can capture capture aspects of a language, such as analogous words or semantics of words. Word representation learning is one of the main research areas in natural language process (NLP). Since 2013, various approaches for processing large amounts of textual data to embed words' semantics into low-dimensional vectors have drawn attention [1]. These low dimensional vectors are so-called word embeddings, and shown to be effective on downstream tasks [2]. This thesis compares two types of word embedding methods: traditional and contextual. Traditional methods tend to infuse semantic similarities between words into the embeddings. However, they fail to capture the ambiguous meanings of a word in different contexts, this is known as the issue of polysemy. For example, in the following two sentences, while its meaning is different, the word "bank" has the same representation: "I deposited 100 euros in the bank" and "She was enjoying the sunset on the left bank of the river." Considering the traditional methods, "bank" would have the same representation regardless of its context. Hence, they fail to capture the polysemy of words. Contextual embeddings, aim to include word semantics in different contexts into the embeddings to capture their polysemous nature. At the time of writing, the most prominent contextual methods use a bidirectional way of learning based on the use of a transformer mechanism to combine left-to-right embeddings to right-to-left embeddings, which allows the method to learn the whole context of a document. Hence, a growing area of study is focused on the use of bidirectional learning while integrating pre-trained embeddings, which can be fine-tuned on downstream tasks.

Bidirectional contextual methods have achieved state-of-the-art results in the task of text classification [3] [4]. Indeed, multiple researchers have compared traditional and contextual methods with regard to the integration of downstream tasks and found that capturing the polysemy of words into embeddings are advantageous in text classification [5]. However, the differences between the use of traditional and contextual methods for email classification have not yet been thoroughly investigated. Only one previous study has compared the use of these methods to classify emails; this work concluded that the majority of the contextual methods outperformed traditional ones, nevertheless, some state-of-the-art contextual methods showed poor performances [6]. The reason for this discrepancy is unknown and requires further investigation.

Furthermore, until now, most experiments on email classification (please refer to Section 2.1) use machine learning models that require to train the model from scratch. Therefore a lot of labeled training data is necessary to create a reasonably performing classification model, and labeling training data is a time-consuming and expensive task. Transfer learning is a type of learning that uses knowledge learned from a previous task to improve learning in a related target task [7]. Hence, it deploys pre-trained embeddings on a general domain and fine-tunes these embeddings on a task-specific domain. This requires less labeled training examples and seems to be promising, yet not thoroughly investigated on the task of email classification.

This thesis proposes several methods, some of which are state-of-the-art, for the task of email classification. Furthermore it investigates whether methods trained on a general corpus can improve the classification of emails. The main research questions are:

- RQ1 *What is the impact of contextual word embeddings compared to traditional word embeddings on the performance of email classification?*
- RQ2 *How do embedding models trained on different domains impact the performance of email classification?*

In the following chapters, relevant literature on email classification, traditional word embeddings and contextual word embeddings is discussed. This is followed by a description of the selected representations methods used for the experiments in this thesis. Then, a description of the data, the different pre-processing steps and the implementation details of the different representation methods is given. In addition, the results of all experiments are presented and analyzed followed by an answer to the research questions. Finally, the strengths and weaknesses of the different methods are explored and discussed.

2 Related Work

This chapter discusses previous studies on email classification and both traditional and contextual word representations.

2.1 Email classification

Due to the drastic increase in the use of the internet and online textual information, there is an ever-growing demand for methods by which to classify emails. Email classification is considered to fall within the research area of text classification, which has been widely researched since 1980. Most email classification approaches use supervised learning, for which labeled data is required. This need for labeled data leads to the first of the two substantial challenges in the field of email classification. Labeling data is a time-consuming and expensive task. The second significant challenge in this field is the lack of publicly available email datasets. Emails may contain confidential information; thus, it is difficult to identify appropriate datasets of emails that are publicly available. This section provides an overview of various research studies on email classification over the years and outlines several findings that informed choices made in this thesis.

The need for email classification became clear due to the emergence of spam messages in early 1990. From then on, many experiments were conducted to explore the implementation of different classifiers on top of simple count-based representation methods, such as bag-of-words and term frequency-inverse document frequency (TF-IDF) (please refer to Section 3.1). Different classifiers deployed during these years include naïve bayes, logistic regression, support vector machines, and random forest. However, the lack of a large benchmark email dataset has been an obstacle to studying the strengths and weaknesses of these approaches [8]. In 2002, an unusual event occurred: a large set of emails, called the Enron Corpus, was made public. The corpus contained over 600,000 emails generated by 158 employees of the Enron Corporation, an American energy trading company that collapsed in 2001 after massive accounting fraud became public knowledge. The Federal Energy Regulatory Commission decided to make the emails publicly available for academic purposes. The Enron email dataset, used by many researchers, includes over 3,500 different user created labels, which are named after the folder to which the email was assigned by the user.

A study in 2004 compared different classifiers — multinomial logistic regression (the same as logistic regression but for more than two classes), support vector machines, and naïve bayes in addition to a simple count-based method, namely bag-of-words - on the Enron dataset. With a range of accuracy scores between 56.4% and 94.6%, the support vector machine outperformed the other classifiers. However, the more basic multinomial logistic regression did not exhibit worse performance, and indeed performed very slightly better, than the support vector machines classifier [9].

Another study conducted in 2004 focused on classifying emails into different classes representing the intent of the sender, which the researchers named speech acts. Examples of speech acts are: "propose a meeting" or "deliver information". The researchers created an ontology of nouns and verbs covering some of the possible speech acts associated with emails [10]. The experiments were conducted using four different datasets — the PW CALO corpus [1] and three subsets of the CSpace email corpus [2] — with a total of 1,357 emails. The researchers noted that the most important words for separating the speech acts from each other were common words. This finding suggests that TF-IDF weighting was inappropriate for distinguishing between common words, because the TF-IDF gives less weight to common words (please refer to Section 3.1.1). To create a representation of the text, the authors used bigrams with an unweighted bag-of-words, which do not attach less weights to common words like TF-IDF. Finally, for classifying the emails, the authors compared the use of the different classifiers: support vector machine and a decision tree. These classifiers were placed on top of the bag-of-words count-based method. The best performance was a precision above 80% and a recall above 50%.

Prior to the early years of 2000, a significant amount of research was conducted on the performance of classifiers using simple count-based representation methods, such as the previously mentioned bag-of-words and TF-IDF. However, this thesis is mainly focused on the performance of different representation techniques. After 2004, there was a shift toward deep learning in the research field of text classification. Deep learning methods improved performance in terms of accuracy for text classification, but the gains come only when a massive amount of labeled data is available [11]. Finding a massive corpora of labeled emails or labeling large corpora of emails, is a challenge in the field of email classification. However, there was a shift from simple count-based methods to more complex representation methods. Because the Enron email dataset is exceptionally large and labeled, research on email classification focused on this dataset using a deep learning approach. In 2018, experiments involving a convolution neural network and long short-term memory networks (LSTM) (please refer to Section 2.3) was conducted using the Enron dataset [12]. A convolutional layer captures local features, followed by an LSTM layer considering the meaning of a feature related to the memory of information about earlier sentences [6]. In this

[1]http://universal.elra.info/product_info.php?cPath42_43products_id2152
[2]http://universal.elra.info/product_info.php?cPath42_43products_id2151

research, the model detected the intent of an email with 89% accuracy, 90% precision, and 93% recall.

Another study in 2018 explored the impact of transfer learning on determining the intent of an email. A language model was trained on the entire Enron dataset and fine-tuned on a smaller email dataset. This is a form of transfer learning, however the knowledge was gained on a domain that was similar to the domain on which the model was fine-tuned. Researchers compared the performance of the language model to the performance of a language model only trained on the smaller dataset [13]. The results showed the addition of transfer learning resulted in a increase of 85.42% to 88.43% of the accuracy score, with a precision and recall of 84% and 84%, respectively.

Finally, the following research approach is most similar to what this thesis aims to explore. A study in 2020 investigated the difference between context-free and context-dependent word embeddings and the impact of pre-trained embeddings on the task of detecting the intent of an email. Researchers compared context-free word embeddings (Word2Vec and GloVe, please refer to Section 2.2), contextual word embeddings (ELMo and BERT, please refer to Section 2.3) and sentence embeddings (transformer-based universal sentence encoder, not elaborated on this thesis). They obtained the best results for ELMo (contextual embeddings) and achieved an accuracy of 90.10% (compared with 82.02% for Word2Vec, 58.08% for BERT, 86.66% for transformer-based universal sentence encoder) [6]. The reason for the major difference in BERT's performance compared to all other methods, including the context-free ones, is unclear and not clarified in the paper. However, it should be noted that BERT was fine-tuned for 10 epochs on a dataset of 9,055 labeled emails. This is a surprisingly large number of epochs and may have resulted in overfitting. The researchers mentioned that using two to four epochs during fine-tuning should be sufficient because BERT was already extensively tuned while pre-training the embeddings [3]. Nevertheless, the low performance of the state-of-the-art method BERT on the task of email classification raises questions. However, other research on email classification using BERT was not found thus the results cannot be compared to other experiments. Thus, this thesis aims to explore the performance of BERT while comparing the results with context-free methods.

To conclude, there has been a shift from traditional to contextual representation methods over the years in the field of email classification. Although it is difficult to compare studies using differing datasets, an improved performance of contextual as opposed to traditional methods was shown on the Enron dataset. This suggests that capturing the context and the polysemy of words into embeddings is advantageous for classifying emails.

2.2 Traditional word representations

Word representation learning has been one of the main research areas since the beginning of NLP [2]. A machine is unable to understand raw text so it is necessary to map words to vectors of real numbers before they are fed as input to the algorithm. There are two types of traditional word representations: count-based and prediction-based.

The purest form of a count-based method, bag-of-words, uses the frequency of each word as a feature for training a classifier. Another count-based method, TF-IDF, involves evaluating how relevant a word is to a text in a collection of documents. Hence, an advantage of TF-IDF over bag-of-words is that it provides information about which words are more or less important in a document. These two count-based methods are easy to understand and implemented and are used as a baseline method in many research studies [5]. However, despite their simplicity, they have several drawbacks. Firstly, neither methods accounts for similarities between words since each word is independently presented as an index [5]. Secondly, these methods cannot capture the semantics of words. Lastly, every word in a text is encoded as a one-hot-encoded vector.

As the number of words potentially runs into the millions, count-based methods face scalability challenges, also known as the curse of dimensionality

One of the earliest papers on creating count-based word representations was published in 1992. The researchers approach was clustering all words that occur in the same context and assigning these words to classes based on the frequency of their co-occurrence with neighboring words [14]. Researchers of a paper in 2003 were among the first to address the curse of dimensionality associated with count-based methods [15]. This was done by identifying a distributed representation of words, using this to inform the network and predicting the next word in a sentence. Every word can be mapped into a low dimensional vector. This work is very similar to later research that gained a lot of attention.

The approach adopted in the above mentioned work is similar to that used in a later study that attracted much attention. Namely a method named Word2Vec which was introduced in 2013 and proposed for processing massive amounts of data to embed words' semantics into low-dimensional vectors [1]. These word embeddings have demonstrated their effectiveness in storing valuable syntactic and semantic information [2], thereby introducing a prediction-based approach to word representation that models the probability distribution of the predicted next word [6]. There are two different approaches for learning the prediction-based representations. The first approach is named continuous bag-of-words (CBOW) and implies the probability of predicting a target word based on its surrounding words. The second approach is named Skip-gram, and implies predicted surrounding words based on a target word. The network outputs high probability scores for a combination that is likely to occur and low scores for rare combinations. Finally, the prediction-based network is updated by gradient descent, and it back-propagates the derivatives into a word embeddings matrix. Following the finding of Word2Vec, many new methods considering processing massive amounts of data to embed words' semantics were proposed.

However, despite the advances that have been made in embedding semantics, one of the disadvantages is that these methods fail to capture polysemy of words. Each single word type is projected as a single point in the semantic space; thus, regardless of the context in which a word appears and any contextual meaning it may have, only one representation is generated.

2.3 Contextual word embeddings

Contextual word embeddings aim to reflect the ambiguous meanings of words in different contexts (polysemy). To understand the meaning of a document, it is important to consider its context as a whole. This is possible with a bidirectional, as opposed to left-to-right or right-to-left, approach. However, before exploring the bidirectional approach, several methods that include context in their embeddings but make use of a unidirectional approach are described.

A recurrent neural network is one method that includes the context in the embeddings. One of its advantages is that it connects previous information to the present task. It also processes sequences and predicts the next word in a sentence based on previously processed information. A recurrent neural network uses patterns to predict the next word and remembers dependencies between words [16]. On occasion, only recent information is needed to predict a target word, for example, predicting the word "sky" in the sentence "The clouds are in the sky". Hence, for predicting the next word in a sentence, the word "sky" is the final word that is predicted, thus the method is able to use all previous words and thus the whole context for predicting the word "sky". However, there are other cases when it is important to know more of the context, such as considering the sentences "I grew up in France" and "I speak fluent French." To predict the word "French" in the second sentence, additional contextual information is required from another area in the text. However, when the gap between sentences grows, a recurrent neural network can no longer connect those dependencies. Therefore, a special type of recurrent neural network was introduced: the LSTM.

Long short-term memory networks, which keep track of long-term dependencies, were introduced in 1997 [17] but have made a comeback recently. Such networks remove and add information to a hidden state, which is subsequently used to predict the next word in a sentence. A simple LSTM fails to consider the whole context of a target word due to its unidirectional approach, but a new version of the LSTM was introduced in 2016. Context2Vec uses a bidirectional LSTM language model to replace the averaged word embeddings in a fixed window and served as the foundation for a great deal of subsequent research [18]. Indeed, researchers have demonstrated that Context2Vec generates high-quality context representations, and it was found to train billions of words in a reasonable time to outperform the traditional word embeddings of the Word2Vec approach on three different tasks.

Since then, much new research has been conducted using a bidirectional approach. For instance, ELMo, created in 2018, uses a three-layered bidirectional LSTM trained on a specific task to create its embeddings [19]. It concatenates all hidden layers for the forward and backward embeddings together, multiplies each concatenated embedding by its weight, and adds all weighted embeddings to generate the contextual embedding.

One year before ELMo was proposed, a paper titles "Attention is all you need" [20] was published; it described the importance of a transformer model, which includes an attention architecture (please refer to Section 3.2.1). The most impactful benefit of a transformer is that it lends itself to running in parallel. The bidirectional LSTM of Context2Vec and ELMo concatenates the hidden layers of the forward and backward embeddings. However, by running in parallel, the transformer creates embeddings in one action instead of concatenating both unidirectional learned embeddings.

The transformer mechanism was mainly used for the NLP task of translation until 2018, when BERT was proposed. BERT is applicable to other NLP tasks, such as entity extraction, question answering, natural language inference, and classification and yielded state-of-the-art results on 11 different NLP tasks. A study on general language understanding evaluation (GLUE) compared the performance of BERT with that of other bidirectional methods such as ELMo and OpenAI GPT (a generative pre-training transformer decoder model [21]). Furthermore, the experiments involved eight datasets (described in [22]). The results show that BERT outperformed all other methods on every dataset with an average accuracy improvement of 4.5% and 7.0%, respectively [3].

Another research conducted experiments on answering questions using the Stanford Question Answering Dataset; the goal was to predict the answer to a question from Wikipedia. The researchers compared BERT to ELMo (and other leaderboard entries that do not have up-to-date public system descriptions available); BERT outperformed all other systems by a wide margin in terms of accuracy. Moreover, BERT outperformed the top leaderboard systems by 1.5% in the F1 score and ELMo by 6.0% [3].

In 2019, a paper introduced a new method named XLNet [4]. XLNet is an improvement on BERT, as BERT neglects dependency in its type of language modeling and suffers from a pre-train and fine-tune discrepancy (see also Section 3.2.3). The researchers responsible for the creation of XLNet explored the difference between XLNet, BERT and some extensions of BERT like RoBERTa and ALBERT. The authors tested the performance of these approaches again on the task of GLUE and used the same eight datasets as used in the previous research on GLUE [22]. XLNet outperformed BERT and RoBERTa on all datasets with average percentages of 90.8%, 86.7% and 90.2% respectively [4].

These results are why it is considered worth exploring the use of the methods BERT ans XLNet in this thesis, which focuses on bidirectional learning.

3 Methods

This chapter describes the details of the selected traditional (3.1) and contextual representation methods (3.2). The section on the latter starts with an explanation of techniques that are necessary to understand the contextual methods used in this thesis.

3.1 Traditional methods

This section describes three popular traditional methods that are used in this thesis. The first method is the count-based method TF-IDF (3.1.1), followed by the prediction-based methods Word2Vec (3.1.2) and FastText (3.1.3).

3.1.1 TF-IDF

As an extension of bag-of-words, TF-IDF is a popular method for determining how important a word is within a document. The method can be divided into two categories, TF and IDF. The term frequency (TF) measures how frequently a word occurs in a document, which is also known as bag-of-words. In contrast, the inverse document frequency (IDF) measures the uniqueness of a word by diminishing the weight of terms that occur frequently in a document and increasing the weight of terms that occur rarely. For example, assume there are 10 documents and the word "embedding" appears in one of the documents multiple times. In contrast, the word "a" appears in 10 of the 10 documents multiple times. Clearly, the word "embedding" provides more differentiation to the document in which it appears. The logic behind TF-IDF is to upgrade the importance of rare terms which suddenly occur often in a document (such as the word "embedding"). The method assigns higher weights and thus higher values to these terms. Furthermore TF-IDF downgrades the importance of terms that occur frequently in many documents (such as the word "a"). The method assigns lower weights and thus lower values to these terms. The formula is as follows:

$$w_{i,j} = tf_{i,j} \cdot \log\left(\frac{N}{df_i}\right) \tag{1}$$

where *tf_i,j* equals the number of words i occuring in a document j, *df_i* equals the number of documents containing i and N is the total number of documents. The word frequencies are distributed exponentially; thus the log of the IDF, weights the words and dampens the effect of the ratio.

Furthermore, TF-IDF is an efficient and simple algorithm that attempts to overcome the problem of common terms in a document. However, despite its strengths it suffers from three major drawbacks. Firstly, it does not capture the semantics of words. This means that the TF-IDF method fails to create an understanding of the meaning of a word and the content of an email. Understanding the meaning of a word in different contexts, or capturing the polysemousness of words, is therefore not an option. Secondly the TF-IDF does not capture the position of words. For instance, the two sentences "it is bad, not good at all" and "it is good, not bad at all" have the same representation despite the fact that they mean the opposite. The last drawback of TF-IDF is that it suffers from the curse of dimensionality when applied to large documents. As already explained in Section 2.2, the TF-IDF encodes every word in the vocabulary as one-hot-encoded vector and thus the number of dimensions are equal to the number of unique terms. When using very large documents the dimensions can increase dramatically which is problematic for any method that requires statistical significance [5].

3.1.2 Word2Vec

Word2Vec is a word embedding method, first published in 2013 [1], that represents words by low-dimensional fixed-size vectors determined by exploiting a two-layered neural network. The approach is an improvement over TF-IDF because it avoids the curse of dimensionality. Another advantage of Word2Vec over TF-IDF is that it captures the semantic similarity between words. Hence, words with similar meanings tend to have similar vectors. The similarities between these vectors can be computed using the cosine similarity formula:

$$similarity = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} \mathbf{A}_i \mathbf{B}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{A}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{B}_i)^2}} \tag{2}$$

Moreover, words with similar meaning tend to have similar vectors and thus obey the laws of analogy [23]. Similarity exists in terms of several characteristics that are alike, for example, "Woman is to queen as man is to king."

$$v_{queen} - v_{woman} + v_{man} \approx v_{king} \tag{3}$$

These are the word vectors (v) for queen, woman, man and king respectively and they strongly suggest that word vectors encode valuable semantic information about the words that they represent [23]. To create these distributed representations of words, Word2Vec uses two different architectures: continuous bag-of-words (CBOW) and skip-gram. These networks take a large text corpus as input and reconstruct it in the fixed-size vector space. For each word, an N-dimensional vector is created in the semantic space using distributed representations. A disadvantage of the Word2Vec method is that it has a fixed vocabulary size. The vocabulary is configured based on the training data and after the training process, Word2Vec is not able to offer a solution when faced with a new word or, words that are not in its vocabulary, also known as out-of-vocabulary-words. Hence, the vectors of these words unknown words are initialized with zeros or random numbers.

CBOW Figure 1 depicts the architecture of a CBOW. The CBOW is based on a feed-forward Neural Network [15] and aims to predict a target word based on its surrounding words. This is done by minimizing the loss function:

$$E = -\log(p(y_i | x_{Ck})) \tag{4}$$

where y_i represents the target word and $\{w_{1k}..x_{Ck}\}$ represent the sequence of words in the context [2]. Figure 1 shows an input layer, a hidden layer and an output layer. The input layer is of the same size as all words in the vocabulary and encode the context words that surround a target word as one-hot vector representations. Note that CBOW makes use of a fixed sized window of context words. Hence, the context is limited to the words that directly neighbor the target word.

The one-hot encoded input vectors are connected to the hidden layer by a $V \times N$ weight matrix W. Each row of W becomes a N-dimensional vector indicated by h. The hidden layer will represent the average of the input vectors, which are weighted by the matrix W. This is computed using:

$$h = (x_1 \parallel ... \parallel x_c)^T \mathbf{W} = x^T \mathbf{W} \tag{5}$$

where $\parallel$ denotes the vector concatenation and the size of the N-dimensional weight matrix. Recall that this is computed using the following equation:

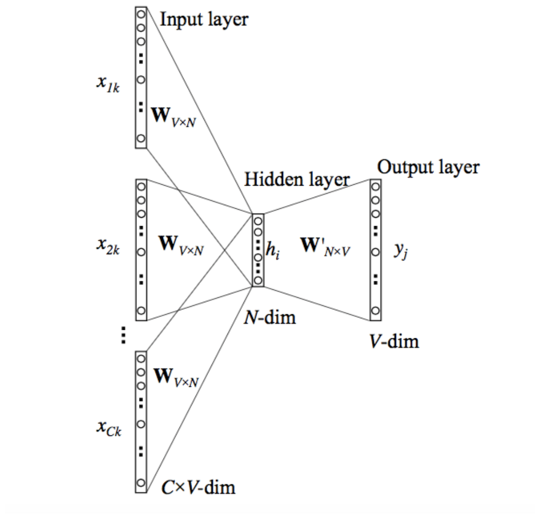


Figure 1: CBOW architecture [24]

$$\mathbf{h} = \mathbf{x}^{\mathrm{T}}\mathbf{W} = \begin{bmatrix} x_1\, x_2\, \ldots\, x_{\mathrm{k}}\, \ldots\, x_{\mathrm{V}} \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} & \ldots & w_{1\mathrm{N}} \\ w_{21} & w_{22} & \ldots & w_{2\mathrm{N}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{\mathrm{k}1} & w_{\mathrm{k}2} & \ldots & w_{\mathrm{kN}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{\mathrm{V}1} & w_{\mathrm{V}2} & \ldots & w_{\mathrm{VN}} \end{bmatrix} \tag{6}$$

$$= [x_k w_{k1}\ x_k w_{k2}\ ...\ x_k w_{kN}]$$
$$= [w_{k1}\ w_{k2} ... w_{kN}]$$
$$= W_{(k,.}$$
$$:= v_{(WI)}$$

The hidden layer is connected to the output layer via a $N \times V$ weight matrix W'. After establishing the hidden layers, one multiplies the hidden layers with the hidden to output the weight matrix:

$$u_j = {v'_{wj}}^T \times h \tag{7}$$

where v'_{wj} is the jth column of the matrix W' [23]. Considering u_j, the measurements of the match between the context and the next word is known. For this value one can use a Softmax classification function to obtain the final distribution embedding of the target word as follows:

$$p(w_j|w_I) = y_j = \frac{exp(u_{cj})}{\sum_{j'=1}^{V} exp(u_{j'})} \tag{8}$$

The error is calculated by summing up the difference between the target word and the predicted word. Finally the back-propagation function is used to calculate the number of adjustments needed to update the weights.

Skip-gram Figure 2 depicts the skip-gram architecture. The goal of the skip-gram is to predict words in the surrounding context of a target word. The concept behind the skip-gram architecture is very similar as that underlying CBOW. However, instead of outputting one multinomial distribution (one target word), the Skip-gram method outputs multiple multinomial distributions (context words). Each output is computed by the following equation:

$$p(w_{cj} = w_{o,c}|w_I) = y_{cj} = \frac{exp(u_{cj})}{\sum_{j'=1}^{V} exp(u_{j'})}. \tag{9}$$

Skip-gram tends to be better than CBOW at predicting rare words because CBOW average the embeddings of the context words in the hidden state to predict the target word which show to be disadvantageous for rare words. Averaging is not required for skip-gram because it predicts context words based on one target word, and thus no additional averaging step is needed. Another advantage of skip-gram is that it works well with small amounts of training data. However, CBOW is significantly faster to train than skip-gram and has been shown to be slightly better at predicting frequent words [25].

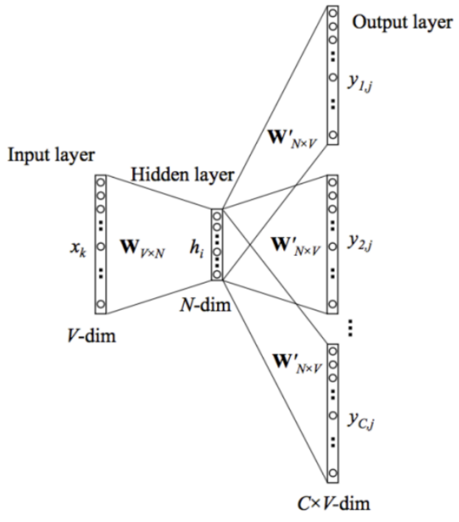


Figure 2: Skip-gram architecture [24]

3.1.3 FastText

FastText is an extension of Word2Vec developed by Facebook's AI research lab and created for learning word representations and capturing the semantic similarities between words [26]. As with Word2Vec, FastText supports both the CBOW and skip-gram architectures. However, the main difference between Word2Vec and FastText is that the latter manages out-of-vocabulary words, whereas Word2Vec does not. When FastText encounters a word not in the vocabulary, it creates a representation of that word by treating it as if it is composed of character n-grams. These corresponding n-grams are summarized using the:

$$s(w, c) = \sum z_g^T v_c. \tag{10}$$

Assuming n=3 for the word "embeddings", the FastText representation of the character n-grams would be$< em, emb, mbe, bed, edd, din, ing, ngs, gs >$. Hence, the concept that words with overlapping n-grams have similar meanings and similar vectors remains. Furthermore, FastText demonstrates superior performance opposed to Word2Vec, at creating embeddings for words that rarely occur in a training corpus [26].

Word2Vec and FastText are considered representation methods, which means that the methods are used for creating a representation of a text and are not able to classify a document yet. However, FastText provides an additional built-in classifier for classifying a document. The classifier architecture is similar to the CBOW of Word2Vec, where the middle word is replaced by a label. The Softmax function is used to compute the probability distribution of the predefined classes [25]. The loss function is computed as follows:

$$-\frac{1}{N}\sum_{n=1}^{N} y_n \log(f(BAx_n)), \tag{11}$$

where y_n is the label, f is the Softmax function, A and B the weight matrices and x_n the normalized bag of features of the n-th document. If the number of labels for a dataset is large, a hierarchical softmax function can be used to improve the running time. However, in this thesis, only a small number of labels are deployed.

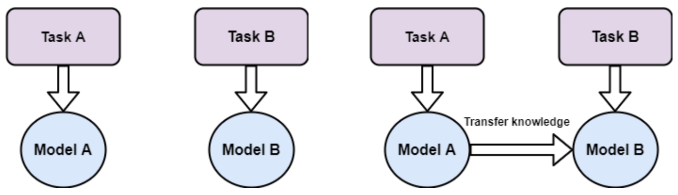
3.2 Contextual models

Before describin the promising contextual methods, this section starts by explaining relevant information that serves as the building blocks for the contextual methods used in this thesis. Thereafter, the details of the selected contextual methods BERT (3.2.2) and XLNet (3.2.3) are provided.

3.2.1 Background

Transfer learning Transfer learning is a fundamental method in modern NLP research. It has emerged as a powerful technique that is widely used in different representation methods [27]. Transfer learning involves storing knowledge that was gained while solving a problem and applying this knowledge to a different problem. For example, it is easier for an individual who already has programming experience to learn a new language compared to a person with no programming language. Transfer learning stores the knowledge in pre-trained embeddings that were trained on a data-rich corpus. Such pre-training enables the method to develop general knowledge of a language [28]. The understanding can range from low-level spelling and character recognition to high-level understanding of abstract textual concepts. This understanding can be leveraged from any source of data, ranging from unlabeled data, which is abundant, to extremely rare and valuable labeled data.

There is a stark difference between traditional machine learning and transfer learning. Classic machine learning does not retain the obtained knowledge, and learning is performed without considering past knowledge learned from other tasks (figure 3(a)). In contrast, transfer learning retains the knowledge learned from previous tasks (figure 3(b)). One of the greatest advantages of transfer learning over classic machine learning is that it requires less training data on the downstream task. This reduces the time and effort associated with manual labeling [29]. Another advantage of transfer learning is that it reduces computational power when training on the downstream task since the embeddings were already heavily trained and tuned during the pre-training phase. One of the major challenges in developing a transfer method is producing a positive transfer between appropriately related tasks while avoiding tasks that are less relevant [7]. While humans can determine what kind of training task is sufficiently similar to the target task, algorithms cannot necessarily do so.



(a) Traditional Machine Learning

(b) Transfer Learning

Figure 3

Transfer learning can be categorized into three subcategories: inductive, transductive, and unsupervised. In the inductive transfer learning setting, the target is different from the source task. Labeled data for the target domain is then required to induce the model to be used in the target domain. For transductive transfer learning, the source and the target domains are different, while the source and target tasks are the same. In this situation, there is a large amount of labeled data available in the source domain but no labeled data in the target domain. The last category, unsupervised transfer learning, is similar to inductive transfer learning but focuses on unsupervised learning tasks in the target domain, such as clustering and dimensionality reduction [27]. In this thesis, the focus is on inductive transfer learning since the learned features must be induced and mapped to a class label. Inductive transfer learning utilizes the inductive biases in the source task to assist in the target task.

Transfer learning comprises two approaches to deploying the learned pre-trained embeddings on the downstream task: feature-based and fine-tuning. Fine-tune approach allows fine-tuning of the pre-trained embeddings on the new task by updating the embeddings utilizing back-propagation. If the dataset of the downstream task is small and the number of parameters is large, fine-tuning may result in overfitting [30]. Hence, the hyperparameters should be gently tuned. The feature-based approach does not allow one to tune the pre-trained embeddings; instead, it freezes the weights of these embeddings.

Transformers In 2017, a paper was titled "Attention is all you need" [31] was published. The paper proposed a transformer architecture that relies entirely on an attention layer. There are two advantages of using a transformer over other networks. Firstly, it uses a bidirectional approach, that incorporates the whole context instead of a left-to-right or right-to-left approach, as is the case of a simple LSTM. Secondly, it does not require processing a sequence in a specific order; hence, it lends itself to running in parallel.

The transformer is also known as a sequence-to-sequence model because it transforms a given sequence of words into another sequence of words. This is useful for translation tasks because it inputs the text of a specific language and outputs the translation of that text without translating it word by word, as is the case in other approaches. Therefore, it makes use of a encoder and decoder mechanism. The encoder maps the input sequence into a higher dimensional vector space, and the decoder does the opposite by mapping these vectors to an output sequence.

The encoder and decoder are fully connected by attention layers, which are comparable to the hidden state of the LSTM. Figure 4 depicts the architecture of an attention layer, which includes two general types of sublayers: multi-head attention and position-wise feed-forward networks.

The purpose of an attention layer is to gather information about the relevant context of a given word and to encode the context into the vector that represent the word. The titles "Attention is all you need" explains that that an attention function maps a query and a set of key-value

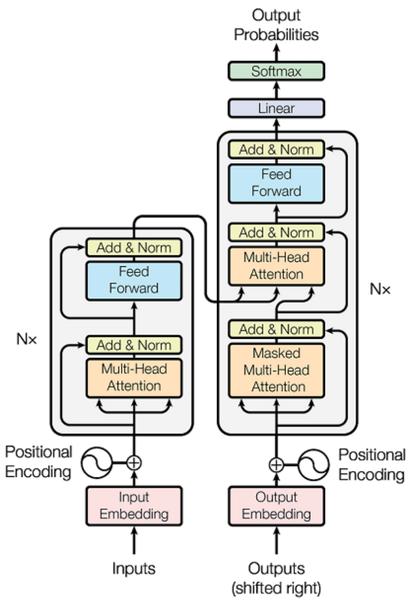


Figure 4: Attention layer [20]

pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key [20]. This is computed using the following equation:

$$\mathit{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{n}})\mathbf{V}. \tag{12}$$

This formula can be explained using an example sentence consisting of three words —x_1, x_2, x_3— and attempting to determine how each word is valued by the other words. The first step is to create three separate vectors of each word: a query, key, and value. These vectors are initialized at random and updated based on the gradient of the loss function using back-propagation during training.

The second step is to score every word against all other words using the query and key (depicted in figure 5(a)). As shown in the figure, the query vector (Q) of every word is multiplied by the key vector (K) of each other word. Before applying a softmax function, the scores are divided by the square root of the dimensions of the vector to ensure stable gradients when the dimensions are high. Finally, the softmax function is applied to all of these scores, which ensures the relative differences between the words and the bounding of the scores.

The third step is depicted in figure 5(b), where each softmax score is multiplied by the value vector (V) of every word. This step creates a new value vector of itself with respect to the other words. For example, when word x2 is not relevant to word $x1$, the score will be small, and the corresponding value vector $V'_{1,2}$ will be reduced.

Finally, step four involves summing up the weighted value vectors of each word and creating the final embedding. The example given is a description of a single-head attention layer using a sentence of three words. However, in practice, a multi-head attention is much more complex. A transformer can process huge amounts of text. Therefore, instead of scoring one word against two other words, it scores one word against all other words in the corpus being analyzed. This is possible using not only a single-head attention layer but also a multi-head attention layer wherein each head considers the original embedding of a word in a different context. Multiple sets of queries (Q), keys (K), and values are learned in parallel for a single word. This approach enables an attention mechanism to incorporate everything it knows about the meaning of a word in different contexts and include this information in one single embedding.

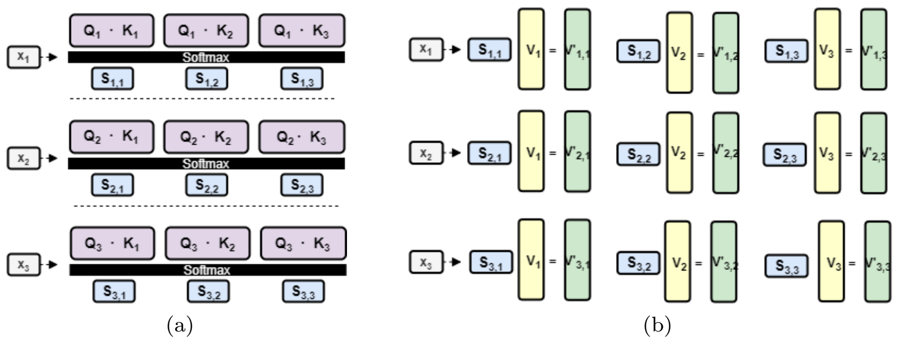


Figure 5: Attention

In addition to the Multi-Head Attention layer, each of the Encoder and Decoder contains a fully connected Feed-Forward network, which is applied to each position separately and identically [20].

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{13}$$

The above function contains two linear transformations with a rectified linear unit (ReLu) activation function between them. It inputs a set of vectors (x_1, x_2, x_3) and outputs another set of vectors (x'_1, x'_2, x'_3) of the same dimensions but mapped to another latent space. A feed-forward network is a simple type of neural network since it contains no cycles or loops. Words in each position of a sequence follow their own path through the encoder and decoder, and while these paths are dependent for the multi-head attention layer, this is not the case for the feed-forward network. The various paths can be executed in parallel while crossing the feed-forward layer.

3.2.2 BERT

Bidirectional encoder representations from transformers (BERT) was presented in a paper published in 2018 [3]. BERT was based on several pre-trained contextual embeddings including generative pre-training [21], semi-supervised sequence learning [32], ELMo [19] , and ULMFit [33]. Moreover, BERT was designed using a transformer mechanism, which generates a language model and is considered bidirectional. The results presented in the paper that introduces BERT show that language models trained bidirectionally are assumed to have a deeper sense of language context and flow than a single-direction language model. Hence, BERT produced state-of-the-art results on 11 NLP tasks [3].

Pre-training BERT The model architecture of BERT consists of a 12-layered bidirectional transformer encoder, and no decoder is included. BERT uses WordPiece for tokenizing the words in a sequence, and creates a fixed-size vocabulary that best fits the language. The vocabulary contains four items: whole words, individual characters, sub-words (occurring at the front of a word), and sub-words (not occuring at the front of a word). As a result of creating sub-words and individual characters, BERT can manage out-of-vocabulary words. For example, the word "embeddings" is not listed in the vocabulary, instead of assigning it an unknown vocabulary token like Word2Vec does, WordPiece divides the word into different characters ["em", "##bed", "##ding", "##s"]. This approach of handling out-of-vocabulary words is similar to that of FastText. Figure 6 presents an example of an input representation of two sentences, which BERT differentiates in two ways: Firstly, the token embedding layer separates sentences with an additional [SEP] token. Secondly, the sentence embedding layer distinguishes two sentences by means of segment embedding, which results in the sentences being separated into A or sentence B. Thereafter, the position of each token in a sentence is created by the transformer positional embedding. Finally, the final representation input of each token is the sum of the token embeddings, the sentence embeddings and the position embeddings.

In order to pre-train BERT in a bidirectional manner, it uses two unsupervised tasks described in the following sections.

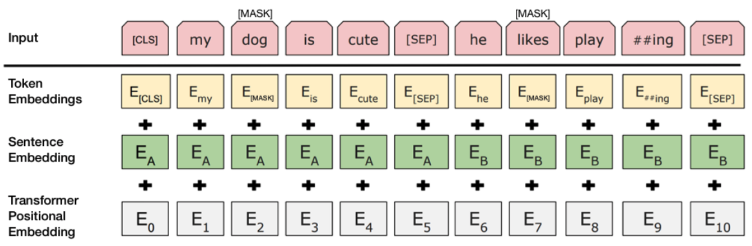

Input
[CLS]
my
[MASK]
dog
is
cute
[SEP]
he
[MASK]
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 6: BERT input representation [3]

Masked language modeling Before explaining what masked language modeling is, is the concept of language modeling itself discussed. The goal of a language model is to learn the joint probability function of sequences of words in a language [15]. Language model pre-training is effective for many NLP tasks and is based on an unsupervised way of learning. This means that language models can process large-scale unlabeled text corpora. Two common pre-training approaches are used for language models: auto-regression and auto-encoding. Auto-regressive-based pre-training means predicting the next word in a sequence based on the previous words. Auto-regression factorizes the likelihood into a forward product $p(x) = \prod_{t=j}^{T} p(x_1|x_{<t})$ or a backward one $p(x) = \prod_{t=j}^{T} p(x_1|x_{>t})$ and notify that this is a uni-directional way of learning. In contrast, auto-encoding-based pre-training deletes some randomly chosen tokens from a given sequence. The task is to predict these deleted tokens simultaneously and not in a particular order. The advantage of auto-encoding as opposed to auto-regression is that the prediction is based on the whole sequence instead of only the left or right part thereof [4].

The authors of BERT proposed a new type of auto-encoding language modeling, named masked language modeling. Masked language modeling randomly selects 15% of the tokens in each sequence, and replaces 80% of these tokens with a [MASK] token. Of the remaining 20% of the randomly selected words, 10% stays unchanged, and 10% is replaced by random tokens. The reason why BERT does not replace all 15% of the chosen tokens with a [MASK] token is that doing so would create a mismatch between pre-training and fine-tuning [3] as the [MASK] token does not appear during fine-tuning. The goal of BERT is to predict the [MASK] tokens based on the other words in the sentence, for which BERT uses an auto-encoding approach.

Task 2: Next Sentence Prediction What is not captured by language modeling is the relationship between two sentences. However, the next sentence prediction captures this relation by predicting the next sentence based on the previous one. As shown in figure 6, BERT separates two sentences by the [SEP] token and distinguish these sentences in the sentence embeddings layer by means of separating them into sentence A or sentence B. Pre-training is designed so that 50% of the time, B is the actual sentence following A (labeled as "IsNext"), and the other 50% of the time it is a random sentence from the corpus (labeled as "NotNext") [3].

Fine-tuning Additionally, BERT can be used for a wide variety of NLP tasks by adding only one additional layer on top of the pre-trained model. For every type of tasks that involves BERT, there is a different type of fine-tuning step. For instance, BERT is used for a classification task in this thesis, and it utilizes fine-tuning for this task by adding a classification layer on top of the output of the [CLS] token. The weights of the classification layer are the only new parameter introduced in the fine-tuning step. The probability of every word and next sentence is calculated using the softmax function:

$$probability = log(softmax(CW^T)), \tag{14}$$

where C represents the final hidden vector of the pre-trained model and W the classification layer weights. The masked language model and next sentence prediction probability are trained simultaneously. Fine-tuning of BERT is relatively inexpensive compared to pre-training.

3.2.3 XLNet

XLNet was the first method to outperform BERT at 18 of 20 tasks, including text classification . XLNet uses a type of language modeling for pre-training the model that differs from that used in BERT and that combines the best of both auto-regressive and auto-encoding language modeling. Although BERT has proven so successful for using a bidirectional way of learning using a transformer mechanism, one of its disadvantages is that it neglects dependencies between masked positions. This disadvantage of BERT is explained further in the following section by an example.

Permutation language model As explained previously, auto-regression predicts the next word in the sentence whereas auto-encoding predicts random chosen words in a sentence based on surrounding words. XLNet combines the best of both approaches by proposing a new type of language modeling: permutation language modeling. Figure 7 presents an example in which the model predicts the token x_3 in four different instances. The figure shows four factorization orders (or the order in which the words will be predicted). Predicting the next word of a factorization order entails an auto-regressive approach since the model predicts the next word in a given order. Considering the top-left example in figure 7, the factorization shows a prediction order of $3 \rightarrow 2 \rightarrow 4 \rightarrow 1$. This means that token x_3 is predicted first, and the model would leverage the memory ($mem^{(0)}$, $mem^{(1)}$) as context for predicting x_3 in this example. Referring to the top-right example in figure 7, the factorization order is $2 \rightarrow 4 \rightarrow 3 \rightarrow 1$, which means that for predicting the word x_3, the model would leverage the memory ($mem^{(0)}$, $mem^{(1)}$), second (x_2) and fourth word (x_4) as context. This type of learning allows the auto-regression language model to utilize a bidirectional context since it is trained on all possible combinations of factorization orders and thus captures the context on both sides.

In practice, XLNet only permutes the factorization order, not the sequence order. Furthermore, the model relies on attention masking and positional encoding to achieve the permutation of the factorization order. As shown in figure 4 (Section 3.2.1), the architecture of an attention layer contains two types of positional encoding for both the encoder and decoder. Positional encoding ensures that the information of words and their positions in a sentence are reflected in the representation.

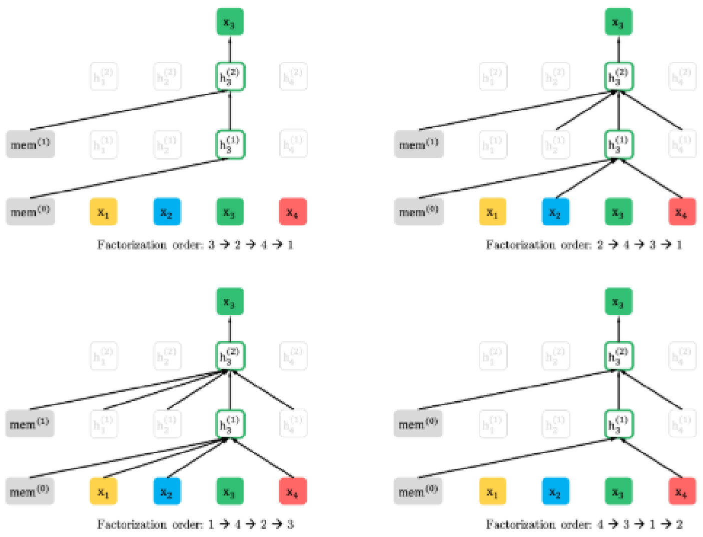

mem(1)
mem(0)
Factorization order: 3 → 2 → 4 → 1
Factorization order: 2 → 4 → 3 → 1
Factorization order: 1 → 4 → 2 → 3
Factorization order: 4 → 3 → 1 → 2

Figure 7: Permutation language modeling [4]

As noted previously, a disadvantage of BERT is that it neglects dependencies between masked positions. Since BERT masks the predicted words in the input, it is not able to model the joint probability of these masked words and assumes that the words are independent of each other [4]. This has its drawbacks, as can be seen in the following example: ”New York is a city.” If both BERT and XLNet select the tokens [New, York] as the prediction target words, and predict these target words based on the context [is, a, city], they would create the following objectives:

$$BERT = \log p(New \mid is\ a\ city) \ + \ \log p(York \mid is\ a\ city) \tag{15}$$

$$XLNet = \log p(New \mid is\ a\ city) \ + \ \log p(York| \ \boldsymbol{New},\ is\ a\ city) \tag{16}$$

As shown in equation 16, XLNet predicts the word ”York” based on the context words and the predicted target word ”New”. This results in that XLNet is able to capture the dependency between the pair (New, York). However, BERT is not able to capture this dependency, because for predicting the word ”York”, BERT only allows to use the context words [is, a, city]. A prediction of BERT can result in a prediction such as (New, Francisco) since ”San Francisco” is another city named by two words, and ”Francisco” is possible prediction of a second word for a city containing two words. Although BERT does learn dependency pairs between the context words and the target (masked) words, XLNet learns more dependency pairs.

The auto-regressive formulation of XLNet overcomes the limitations of BERT and enables learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order [4].

3.2.4 AutoML tool

The use of automated machine learning. . . (AutoML) models for business applications has drawn increasing interest in the recent years. AutoML tools have been developed to save time and effort when performing repetitive tasks such as data pre-processing, feature engineering, model selection, hyperparameter optimization and analyzing prediction results [34]. Large technical companies and cloud providers offer AutoML tools as services or standalone products. For example, Google offers the Cloud AutoML tool[3], Microsoft offers the AzureML tool[4], Salesforce offers TransmogrifAI[5], and Uber offers Ludwig [6]. The main disadvantage of these tools is that they almost all operates as black boxes. This means that they lack of interpretability and cannot be improved by its users. Note that not all of the examples provided above are complete black boxes, as some of them do leverage certain insights into the stored features to help debug models. However, this thesis explores the performance of Google's AutoML, as this is the tool Ciphix implements with its clients before. The AutoML tool does not provide any insights into its working and operates as a black box, however, it claims to incorporate state-of-the-art methods proposed by Google.

4 Experimental set-up

This chapter describes the dataset (4.1), the different pre-processing steps (4.2), the various representation methods and details of the implementations (4.3), and the performance metrics used to evaluate the methods (4.4).

4.0.1 Problem setting

One of Ciphix's clients is a large international company that is referred to as Company X in this thesis. Company X receives thousands of emails each month and the company is currently interested in distinguishing emails referring to invoices from all other emails. To be more specific, Company X aims to separate emails into three different classes: Invoice, Reminder and Other. The Invoice class refers to an email containing information about a new incoming invoice. The information about an invoice is either included in an attachment or mentioned in the body of the email. The Reminder class, refers to an email reminding the recipient of a previously sent invoice that has not yet been paid. Finally the Other class refers to all other emails that Company receives. The goal of Company X is to send push messages to certain employees regarding invoices for which a Reminder was sent, with the goal to remind these employees of the need to pay these invoices. Reminder emails should contain invoice data which is automatically extracted. This extracted invoice data is checked against the database, to determine whether the invoice mentioned in an email is indeed logged on the system and has not been paid. In the event that the invoice is not found in the system, no push message will be sent. Furthermore, when an email belonging to the Reminder class does not contain invoice data, no push message will be sent. The push messages serve to notify the employees of Company X of outstanding invoices; however, emails belonging to the Reminder class are also checked. This indicates that in addition to the outstanding invoices from which a push message was sent, the emails are checked as well for finding outstanding invoices without invoice data. This thesis does not address the extraction of the invoice data; rather, the focus is on classifying the emails. A high level depiction of the classification problem setting is shown in figure 8. As shown in the figure, for automatically

[3]https://cloud.google.com/automl
[4]https://azure.microsoft.com/en-us/free/machine-learning/search/
[5]https://transmogrif.ai/
[6]https://eng.uber.com/introducing-ludwig/

classifying emails to the classes Invoice, Reminder and Other, two main steps are deployed: (1) cleaning the text of the emails and (2) utilizing machine learning to classify the email as belonging to one of the classes.

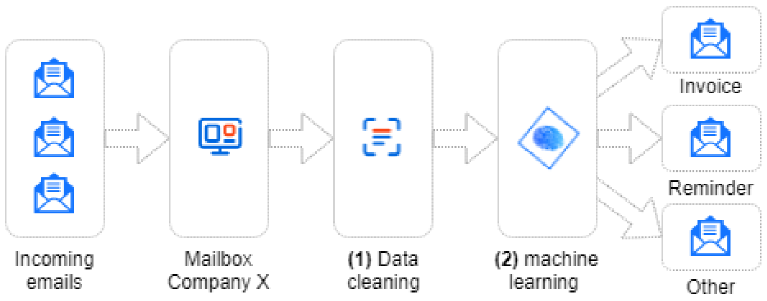


Figure 8: depiction of the problem setting

Company X provides two different datasets, with the one containing labeled and the other unlabeled data. The labeled data is used for supervised learning or supervised fine-tuning whereas the unlabeled data is used for unsupervised learning and unsupervised fine-tuning.

Labeled data For supervised learning, each email should have a corresponding label. These labels were manually applied by three people working at Company X who are referred to as annotators. The machine learning model heavily depends on the accuracy of the labels, which is why the emails should be labeled as objectively and accurately as possible. Therefore, it would be ideal if each email were labeled by all three annotators. However, manual labeling is an extremely time-consuming and therefore expensive activity for companies. Eventually, the labeled dataset of Company X contained a total of 7,061 labeled emails. Of these emails, only 459 were labeled by two annotators, while the remaining 6,566 emails were labeled by one annotator. When two annotators label the same email, there might be disagreement as to the appropriate email. All emails on which the two annotators disagreed were thus removed from the dataset. Inter-annotator agreement is a measure of how well annotators make can make the same decision for a certain class. There are several different ways to measure inter-annotator agreement of which Krippendorff's alpha is one. Krippendorff's alpha measures the reliability of a dataset by calculating the disagreements [35].

This value ranges between 0 and 1, with $\alpha = 1$ indicating a perfect reliability and $\alpha = 0$ absence of reliability. The dataset of Company X obtains a Krippendorff's alpha value of $\alpha = 0.77$. A value between 0.67 and 0.8 is considered as indicating low reliability.

This low value can be explained by the following: Of the 459 emails which were labeled by two annotators, the annotators did not agree on 130. As shown in figure 9, it appears that the annotators had the most difficulties agreeing on emails labeled as Other and Reminder. Examples of common mistakes between these classes were the following. Emails referring to a reminder, but not a reminder of an invoice, therefore, the actual label should be Other. Another mistake was flagging emails which were replies to emails identified as Reminders as Reminders themselves. These emails should be labeled as Other instead of Reminder. Figure 9 indicates that there were few mistakes with regard to the labels Invoice and Reminder. Hence, these emails were easier to distinguish. To ensure agreement among the annotators and the preferences of Company X, several feedback sessions were held with the three annotators, in which the emails and their potential labels were discussed

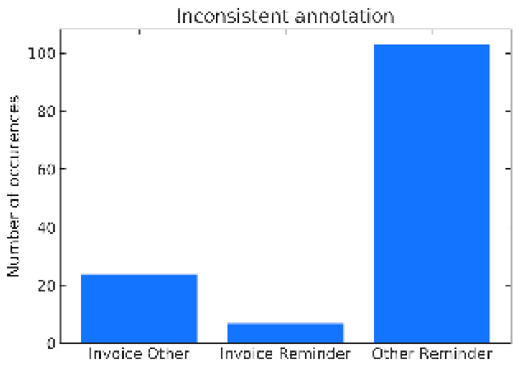


Figure 9: Number of emails that the annotators did not agree upon

After removing the inconsistently labeled emails, the remaining dataset contains of 6,927 emails; table 1 presents a summary of the statistics per label. It appears that the dataset is highly imbalanced because the majority of the emails were labeled was Other and a minority with Invoice. Using an imbalanced training dataset for a classification task can be challenging. Considering a dataset that is not representing all classes equally, may train a classifier that over-fits to the majority class and becomes oblivious to the existence of the minority class. As a result, the emails belonging to the minority class are misclassified more often than those belonging to the majority group [36]. The impact of the class imbalance on the task of classifying emails is explored in the Results section (5.1.1).

	Other	Reminder	Invoice
#docs	4593	1648	691
average doc length	603	632	471
max doc length	24027	13883	52894
min doc length	1	3	6

Table 1: Summary statistics unlabeled dataset

Unlabeled data The unlabeled dataset of Company X considered in this thesis contains a total number of 3,620 emails. This number is fairly low, as the inbox contains significantly more emails. However, it was not possible to obtain additional emails from Company X. The statistics of this dataset are displayed in table 2.

Benchmark dataset A benchmark dataset is not used in this thesis. Benchmark datasets are normally determined to provide a reference against which to compare the obtained result. No benchmark dataset is deployed as a comparable dataset that of Company X was not found. As previously explained, one of the major challenges in the research field of email classification is finding a labeled dataset consisting of real emails. This is mainly due to the fact that emails may

	emails
#docs	3620
average doc length	598
max doc length	22127
min doc length	2

Table 2: Summary statistics unlabeled dataset

contain confidential information. The Enron dataset discussed in Section 2.1 would not have been a viable benchmark dataset because it contains over 3,500 different user-created labels. The labels used in the Enron dataset refer to the folder in which an email was placed. The labels thus rely on the user's own interpretation of the content of an email and his or her foldering strategy. This dataset is thus not comparable with that of Company X.

4.1 Data pre-processing

The text of an email may contain great deal of noise, such as special character in headers and footers, a signature, emoticons, URLs, or misspelled words. This thesis proposes five different pre-processing steps for cleaning and preparing the text before it is fed into the machine: translation, removing headers and footers, removing sensitive information, cleaning the text and removing stop-words.

An important assumption in classification tasks is that many words, symbols, and structures used in languages are not significant in representing the content of the text document without losing information [37]. According to previous studies, removing symbols from text leads to better classification [38]. The impact of these pre-processing steps is explored on the simplest method used in this thesis, the TF-IDF. Considering the impact of the pre-processing steps may help in deciding whether steps should be deployed for the rest of the experiments with the other methods in this thesis. However, TF-IDF is a count-based method, and the impacts of the pre-processing steps are expected to be different for the non count-based methods such as the prediction-based and contextual methods. Hence, the performance of these steps provides decisive insights. Finally, some of the pre-processing steps are required for certain steps performed in this thesis, such as translation and concealing confidential information. Thus, these are included in all experiments, regardless of their performance when incorporated into the TF-IDF method.

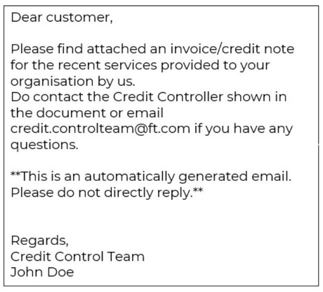
Translation Company X is an international organization that receives emails in several languages. Over 50% of the emails in the labeled dataset were written in English, followed by Dutch, German, and French. Since the majority were written in English, all emails have been translated to that language. Another reason for choosing English is that the most prominent pre-trained embeddings of both contextual and traditional methods are trained on an English corpus.

Translating the text to English was done using Google's AutoML API translator [7]. The use of this translator is not free, and the prices are computed based on the training minutes and the number of characteristics. Feeding the text to the AutoML translator as a whole did not yield the best results, and a portion of the text was not translated correctly. The translation was best when translating the text of an email sentence by sentence.

[7]https://cloud.google.com/translate/automl/docs

Cutting emails The second pre-processing step was removing headers and footers. This was done by creating a list of possible openings and closings of an email. Of the 6,927 emails in the labeled email dataset, 3,980 started with "Dear," 980 started with "Hello," and 1001 started with "Hi." In addition, 1,203 emails ended with "Regards," 908 with "Kind regards," 765 with "Thank you in advance," 687 with "Many thanks" and 567 with "Best regards."

Hiding confidential information The third step was to remove any sensitive or confidential information contained in the emails using a Cloud data loss prevention (DLP) service [8]. The email dataset contains data such as invoice numbers, invoice amounts, and the names of senders and receivers. Figure 10 below provides an example of the content of an email before and after utilizing a DLP services.

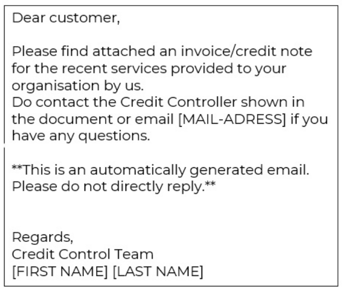

Dear customer,

Please find attached an invoice/credit note for the recent services provided to your organisation by us.
Do contact the Credit Controller shown in the document or email credit.controlteam@ft.com if you have any questions.

This is an automatically generated email. Please do not directly reply.

Regards,
Credit Control Team
John Doe

(a) Without


Dear customer,

Please find attached an invoice/credit note for the recent services provided to your organisation by us.
Do contact the Credit Controller shown in the document or email [MAIL-ADRESS] if you have any questions.

This is an automatically generated email. Please do not directly reply.

Regards,
Credit Control Team
[FIRST NAME] [LAST NAME]

(b) With

Figure 10: Data loss prevention (DLP)

Stop-words Another pre-processing step was stop-word removal. Stop-words are words that are frequently used in a language but do not feature a thematic component and add very little meaning. The English language has around 320 stop-words. Examples of stop-words include determiners (the, a, an, another), coordinating conjunctions (for, an, nor, but, or, yet, so), and prepositions (in, under, towards, before) [39].

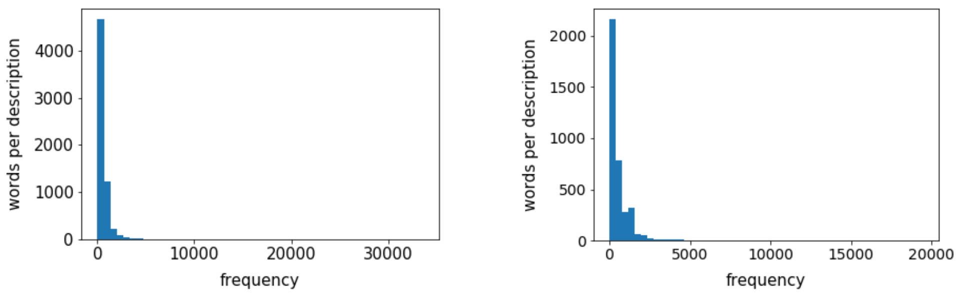
Stop-words may impact the efficiency of information retrieval. Namely, stop-words tend to affect weighting process of determining which word by diminishing the impact of frequency differences among less common words. Hence, the weighting process is subsequently impacted by the length of a document; this can be avoided by removing stop-words. In addition, stop-words carry almost no meaning which may result in unproductive processing [40]. Furthermore, the removal of stop-words removal can increase the execution speed, calculations, accuracy, and efficiency of the algorithm.

In this research, stop-words were removed using the Scikit learn library [9] which contains a list of 318 such words.

Cleaning from symbols The final pre-processing step was cleaning the text of symbols. This thesis uses symbols as a collective term for punctuation marks, URLs, emoticons, special characters and digits. Punctuation marks correspond to neither the phonemes (sounds) nor lexemes (words and phrases) of a language but serve to indicate the structure of writing, as well as intonation and pauses to be observed when reading aloud [41].

[8]https://pypi.org/project/googleclouddlp/
[9]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

To detect and remove punctuation maarks, the regular expression (Regex) operations of the Python standard library[10] was used. This library is a string-searching algorithm for finding or replacing patterns in strings to clean a text and make it more readable. This thesis employed Regex to clean URLs, emoticons, all punctuation marks (except for periods and commas), double white spaces, new lines, digits, and single-letter words. Figure 11 shows a distribution of the number of words per email before (a) and after (b) the cleaning process.



(a) Before

(b) After

Figure 11: Data pre-processing

Figure 11(b) shows that the number of words decreased after cleaning the text with Regex. This is also due to the number of new lines that emails contain, which also contribute to the high frequency.

4.2 Method of implementation

This section describes the implementation of the representation methods and the classifier used in this thesis. Figure 12 presents an overview of the possible of the different design choices that were considered while designing this research. The different methods deployed in this thesis are TF-IDF, Word2Vec, FastText, BERT and XLNet. The first rectangle refers to the type of domain on which the embeddings can be pre-trained: a specific or a general domain. The second rectangle contains to the different strategies for applying the pre-trained embeddings on downstream tasks, namely feature-based and fine-tuning. The third rectangle refers to the different methods used in this thesis. Note that the TF-IDF method does not provide a pre-trained approach, thus the first and second rectangle are not among the choices for the TF-IDF method. The fourth rectangle refers to the classifier, the multinomial logistic regression which is used for all experiments methods. Note that all these different design decisions do not apply the AutoML tool, since it is unknown what type of pre-trained domain, strategies or classifier it uses.

All details about the different experiments of this thesis are explained in the following section.

4.2.1 TF-IDF

The TF-IDF method is the simplest one utilized in this thesis due to its count-based approach. Therefore, it is used as the baseline method.

Different experiments were conducted with TF-IDF to test the impact of the imbalanced dataset and the various pre-pressing steps on its performance. All of the results of the experiments with TF-IDF are presented in Section 5.1 and compared to those of the other methods in Section 5.4.

[10]https://docs.python.org/3/library/re.html

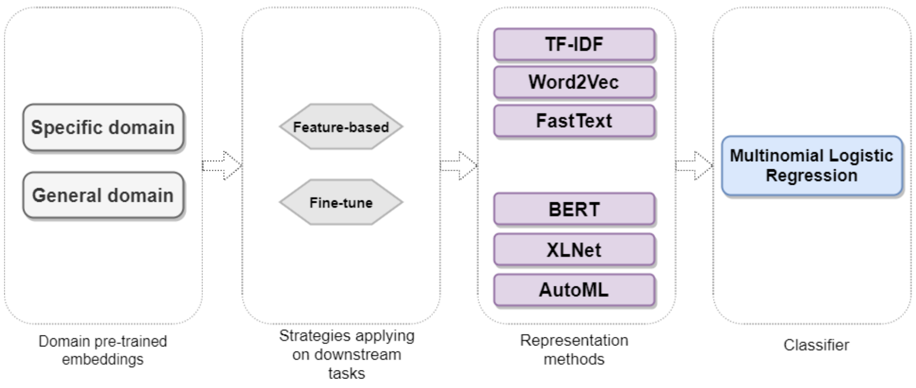


Figure 12: Overview of methods used in this thesis

The first two experiments with TF-IDF used the balanced and imbalanced dataset, respectively. There are two ways to create a balanced dataset: under-sampling and oversampling. Oversampling refers to adding data to the underrepresented classes, and while under-sampling means removing samples from the overrepresented classes. With oversampling, when adding data to the underrepresented classes, synthetic data is created. This synthetic data is artificially generated to replicate the data of the minority classes. Oversampling can easily cause model overfitting by repeatedly visiting duplicated samples [42]. The under-sampling approach removes data from the majority classes, which results in much valuable information being discarded [43]. This thesis exploits the performance of the TF-IDF method using a balanced training dataset created by under-sampling, as oversampling would add less variance to the minority classes and may cause overfitting. This was the reason for correcting the imbalanced dataset, as the minority classes contain fewer samples and less variance and thus result in overfitting. Hence, under-sampling seemed a better preferable for creating a similar distribution for every class.

Hyperparameters For all experiments with TF-IDF, the only hyperparameter that was tuned was the n-gram range. The n-gram range was sweeped with {0, 1, 2, 3}. The n-gram range refers to the number of words that are passed while creating an instance of the TFidfVectorizer. Hence the TFidfVectorizer of the Scikit-learn library[11] was used for conducting the experiments with TF-IDF. This library provides a simple means of both tokenizing a collection of text documents and building a vocabulary.

4.2.2 Word2Vec

The performance of the Word2Vec representation method was explored using pre-trained embeddings on a general domain (Section 6.2) and a specific domain (Section 6.3). A total of four experiments were conducted using the Word2Vec representation method. The first experiment, which applied a feature-based approach on CBOW pre-trained embeddings, trained on a general domain (Section 5.2). The second experiment applied fine-tuning of CBOW pre-trained embeddings, trained on a general domain (Section 5.2). The third experiment applied a feature-based approach on CBOW pre-trained embeddings, trained on a specific domain (Section 5.3). Finally, the fourth experiment applied a feature-based approach on skip-gram pre-trained embeddings, trained on a specific domain (Section 5.3).

[11]https://scikit-learn.org/stable/modules/feature_extraction.html

The reason for underscoring the difference between feature-based and fine-tuning in Section 5.2 is that the impact of fine-tuning is only expected to be useful with pre-trained embeddings on a general domain. In addition, both CBOW and skip-gram architectures are only explored in Section 5.3, while using task-specific pre-trained embeddings, because the skip-gram architecture could not be properly loaded.

Experiments involving pre-trained embeddings on a general domain The first two experiments with Word2Vec involve pre-trained embeddings on a general domain. The embeddings were downloaded from the internet and trained on the Google News Corpus[12] using the CBOW architecture. This corpus contains approximately 100 billion words and phrases with 300 dimensional vectors for 3 million words and created a vocabulary of 51,678 unique words. One experiment used a feature-based approach and the other a fine-tuning approach.

A feature-based approach extracts fixed features from the pre-trained embeddings, which means that all knowledge used for this experiment is found in the pre-trained embeddings. Fine-tuning means that the parameters of the embeddings are fine-tuned on a downstream task, and thus it uses knowledge that was trained on the task-specific domain. This thesis uses the unlabeled and labeled dataset of Company X for fine-tuning the embeddings of Word2Vec.

Experiments involving pre-trained embeddings on a specific domain The third and fourth experiments with Word2Vec involved the pre-trained embeddings on the task-specific dataset. This implies that the dataset of Company X was used and that the embeddings were trained on a dataset containing emails (task-specific). Word2Vec deploy unsupervised pre-training; thus its unlabeled dataset was used. Both CBOW and skip-gram were deployed for these experiments. Ultimately, a vocabulary of around 8,000 unique words was created, which is not very large when compared to the vocabulary of the Google News Corpus (51,678).

Hyperparameters The Word2Vec method depends on several hyperparameters, several of which were already tuned to some extent by the designers of the algorithm. Various hyperparameters such as smoothing the negative sampling distributions or dynamically sized context windows are reported in passing and considered thereafter as part of the algorithm [44].

For deploying hyperparameter tuning, grid searches are used to determine the optimal hyperparameters for Word2Vec, which results in the most accurate predictions. The following three hyperparameters were tuned: learning rate, the window-size and the number of epochs. The learning rate controls the number of adjustment made to the weights concerning the loss gradient. The learning-rate is a value between 0 and 1; smaller learning rates result in smaller changes to the weights in each update, whereas larger learning rates result in rapid changes. The learning rate may be the most important hyperparameter to tune when configuring a neural network. The window-size in Word2Vec incorporates the maximum distance between a target word and words nearby the target word. Larger windows tend to capture more information about a topic whereas smaller windows tend to capture more about the word itself. Finally, the number of epochs is the number of times the algorithm cycle through all training samples. In general too many epochs will cause a model to over-fit on the training data, meaning that it will not be able to generalize well.

For all experiments in this thesis using Word2Vec, the learning rate is sweeped with {0.1, 0.01, 0.02, 0.001}, the window-size with {4, 5, 6, 7} and the epochs with {5, 10, 15, 20}. For the grid-search, the Scikit learn library [2], which evaluates all possible combinations of parameter

[12]https://code.google.com/archive/p/word2vec/
[2]https://scikit-learn.org/stable/modules/grid_search.html

values and retains the best final combination on the accuracy score, is used. Finally, all remaining hyperparameters remain at the default values defined by the researchers who published the original paper on Word2Vec [45]. These parameters are presented in table 3. The dimensionality size of the embeddings is determined by the size on which the pre-trained embeddings were trained, this was 300 for the Google News Corpus.

Hyperparameters	Value
size	300
minimum count	5
negative	5
min-count	2
sample	0.001
workers	3
batch words	1000

Table 3: Hyperparameters default value Word2Vec

All experiments with the Word2Vec method were regulated with the Gensim[13] implementation of Word2Vec in Python, which was originally ported from a C package implementation [14].

4.2.3 FastText

Like Word2Vec, FastText is evaluated using pre-trained embeddings on a general domain (Section 5.2) and on a specific domain (Section 5.3). A total of three experiments were conducted using the FastText representation method. FastText only applies a feature-based approach as strategy for applying the pre-trained embeddings, thus no fine-tuning. The first experiment with FastText used skip-gram pre-trained embeddings on a general domain (Section 5.2), the second experiment used CBOW pre-trained embeddings on the task-specific domain (Section 5.3) and the third experiment used skip-gram pre-trained embeddings on the task-specific domain (Section 5.3).

The architectures of CBOW and Skip-gram are compared in Section 5.3, using task-specific pre-trained embeddings, however, both architecture are not compared using pre-trained embeddings of a general domain in Section 5.2. This was due to the fact that the CBOW embeddings could not properly be loaded and this was the same problem encountered with Word2Vec using Skip-gram pre-trained embeddings.

Experiments involving pre-trained embeddings on a general domain For the first experiment with FastText, the pre-trained embeddings were downloaded from the internet. The pre-trained embeddings were trained on the English Wikipedia corpus [4] using a Skip-gram architecture. This corpus contains approximately 16 billion words and the model was trained using the Skip-gram architecture with 300-dimensional vectors. Eventually, a vocabulary of 21,678 unique words was created.

[13]https://radimrehurek.com/gensim/models/word2vec.html
[14]https://code.google.com/archive/p/word2vec/
[4]https://fasttext.cc/docs/en/pretrainedvectors.html

Experiments involving pre-trained embeddings on a specific domain The third and fourth experiments used FastText and pre-trained word embeddings on the task-specific domain, namely the email dataset of Company X. Like Word2Vec, FastText utilizes unsupervised learning, and thus no labels are required to train the embeddings. To create a unique vocabulary based on the data, the text of all emails was transformed into a list of tokenized sentences, creating a vocabulary of around 7,000 thousand words. The skip-gram was selected to easily capture the local order in a sentence that had previously been used to train the embeddings. This approximates the probability of a target word based on the previous neighboring words and is represented with the parameter word_ngrams which was set to 2. On the word character level, a minimum and maximum number of characters were controlled by the min_n and max_n parameters. For the experiments in this thesis, these were set to 3 and 6 respectively.

Hyperparameters For the first experiment with FastText using embeddings trained on a general domain, no hyperparameter tuning was performed since the embedding layers were frozen due to the feature-based approach. All parameters were set to their default values; these were presented in table 4. For the second and third experiments with FastText, hyperparameter tuning was performed of the learning rate, window-size and number of epochs. The learning rate was sweeped with {0.1, 0.01, 0.02 0.001}, the window-size with {4, 5, 6, 7} and the number of epochs with {5, 10, 15, 20}. For the grid-search, the Scikit learn library [2] evaluates all possible combinations of parameter values and retains the best final combination on the accuracy score.

Hyperparameters	Value
size	300
minimum count	4
negative	5
sample	01e-5
batch words	100000
alpha	0.025

Table 4: Defaults values for FastText

All experiments with FastText were conducted using the Gensim open-source library[2]. This library contains a fast native C implementation of FastText with Python interfaces.

4.2.4 BERT

As with Word2Vec and FastText, BERT is evaluated using pre-trained embeddings of a general domain (Section 5.2) and pre-trained embeddings of a specific domain (Section 5.3). A total of two experiments were conducted using BERT. The first utilized the pre-trained embeddings on a general domain and was additionally fine-tuned on the email dataset. Furthermore, the contribution of each encoding layer to BERT's performance was explored. For the second experiment with BERT, the pre-trained embeddings were further pre-trained on the email dataset of Company X and additionally fine-tuned.

[2]https://scikit-learn.org/stable/modules/grid_search.html
[2]https://radimrehurek.com/gensim/models/fasttext.html

Experiments involving pre-trained embeddings on a general domain The Simple Transformers library, which was created by Thilina Rajapakse[15], was used for implementing BERT. The Simple Transformers library is based on the transformers library by HuggingFace[16] and is useful for training and evaluating transformer models. BERT is pre-trained on a corpus of books and the English Wikipedia, which comprise 13GB of plain text. In addition, the authors of BERT include Giga5 (16GB), ClueWeb, and Common Crawl as corpora for pre-training the network [4]. The paper that introduced BERT presented two versions of this approach: BERT-base and BERT-large. The BERT-base model contains 12 hidden layers, 768 attention heads, and 110 million parameters, while BERT-large contains 24 hidden layers, 16 attention heads, and 340 million parameters. The BERT-base model was selected for all experiments in this thesis because it is less computing-intensive than BERT-large yet still achieves state-of-the-art results on text classification tasks [3].

Beyond BERT-base and BERT-large, there are two other options to consider: BERT-base-uncased or BERT-base-cased. BERT-base-uncased is an approach in which the text is set to lower case (e.g. "John Smith" becomes "john smith"). In contrast BERT-base-cased does not lowercase the text. Since capitalization does not seem to be relevant in the task of classifying the emails of Company X, the BERT-base-uncased option was selected for the experiments.

experiments involving pre-trained embeddings on a specific domain A second experiment with BERT was conducted to explore its performance when further training the pre-trained embeddings on the email dataset of Company X. The reason BERT was not trained from scratch on the email dataset was that doing so would not be realistic, as it would require significant amounts of training data, time, and computational power. The approach of further pre-training a transformer model was inspired by a 2018 study that proposed a universal language model (ULMFiT) that effectively fine-tuned the model in an unsupervised manner [33].

Because further training the embeddings of BERT allows unsupervised training and thus the unlabaled dataset of Company X was used. The text of the unlabeled dataset was prepared and transformed into a file in which every sentence was separated by a [SEP] token. Additionally, 15% of the words in the dataset were randomly masked. Hence, these preparation steps are the same as proposed in the original BERT paper. The number of epochs for further pre-training was set to three, and the maximum sequence length was, as in the BERT-base-uncased model, set to 128. After the additional pre-training phase, the model was saved as a bin file. Thereafter, the further pre-trained model was loaded into the Simple Transformers library. The embeddings were fine-tuned in a supervised manner using the labeled dataset of Company X.

Hyperparameters For the first experiment with BERT were the learning rate and number of epochs deployed by exploring the behavior of the training and validation loss over the different values of the two hyperparameters. The learning rates were first sweeped by {1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7}, and then sweeped by a smaller range {2e-5, 3e-5, 4e-5, 5e-5}.

The authors of the BERT paper recommend only two to four epochs for fine-tuning BERT on a specific task. However, this thesis explores the use of 20 epochs to explore the behavior of the training and validation data and determine whether this assumption also holds for email classification. Eventually, an optimal number of epochs that avoids under- or over-fitting of the method was determined. Finally, the remaining hyperparameters were set to their default values defined in the original BERT paper; these are displayed in table 5.

[15]https://github.com/ThilinaRajapakse/simpletransformers
[16]https://github.com/huggingface/transformers

Hyperparameters	value
gradient_accumulation_steps	1
window-size	5
min-count	2
weight_decay	0
adam_epsilon	1e-8
batch-size	270
warmup_steps	0
max_grad_norm	1.0
logging_steps	50
save_steps	2000
train_batch_size	8
eval_batch_size	8
max_seq_length	128

Table 5: Default hyperparameters BERT and XLNet

4.2.5 XLNet

To evaluate the performance of XLNet, one experiment was conducted using pre-trained embeddings of a general domain (Section 5.2). XLNet was not additionally pre-trained to test its performance when using task-specific pre-trained embeddings. This choice was made due to the fairly complex architecture of the algorithm and the lack of time.

Experiments involving pre-trained embeddings on a general domain Just as for BERT, the Simple Transformer Library created by Thilina Rajapakse [4] was used to train and evaluate XLNet. XLNet's architecture consist of 12 hidden layers, 768 attention heads, and 110 million parameters and the embeddings were pre-trained on the same corpus as BERT (English Wikipedia and Book corpus). Again, the XLNet-base-uncased option was selected to lowercase the text.

Hyperparameters The hyperparameters learning rate and number of epochs were explored on the basis of the behavior of the validation and training loss. The learning rate was swept by {1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7} and additionally by a smaller range of {2e-5, 3e-5, 4e-5, 5e-5}. Finally, a total number of 20 epochs were evaluated and the best performing parameters were selected. The remaining hyperparameters were set to their default values, which were the same for BERT; these are displayed in table 5.

4.2.6 AutoML

The AutoML tool used in this thesis was hosted on the Google Cloud [17]. The AutoML of the Google cloud is a supervised learning service and therefor it requires labeled data. The service is not for free and the costs are $19.32 per hour. The tool does not require any pre-processing of the data before the data imported, thus no translation or cleaning needs to be done. Furthermore, the same train and test splits are made as the other experiments.

[4]https://github.com/ThilinaRajapakse/simpletransformers
[17]https://cloud.google.com/automl-tables/docs/predict

4.2.7 Classifier

In this thesis, all experiments were utilized with a multinomial logistic regression classifier. A multinomial logistic regression is a supervised learning algorithm which is a generalization of the logistic regression to multi-class problems. It creates a probability distribution over the set of different classes and chooses the label with the higher prediction score as the predicted label.

The traditional methods TF-IDF, Word2Vec and FastText are representation methods and adding the multinomial logistic regression classifier on top of the representation method is an additional step. The classifier of these traditional methods was provided by the Scikit Learn library[18] and applied along with the newton-cg solver, which is a type of algorithm used for optimization.

The contextual methods BERT and XLNet have a multinomial logistic regression (or for binary classes a logistic regression) included into their method. Thus, the multinomial logistic regression classifier of the Scikit learn library was not applied for BERT and XLNet since this was already provided by the Simple Transformer library.

Hyperparameters The only hyperparameter tuned for the classifier was the C value and this was only done for the traditional methods, because the Simple Transformer library did not provide tuning hyperparameters of the classifier. The C value corresponds to the regularization strength and is used for preventing the classifier from overfitting. The C value was swept over {1, 10, 100, 1000} and the optimal C value was selected and listed for the experiments with the traditional methods.

4.3 Evaluation Metrices

This section discuss cross-validation and several evaluation matrices selected to evaluate the performance of the proposed methods.

4.3.1 Cross-validation

Cross-validation is a methods used for resampling data to enlarge the variety examples on which a method is trained and tested. The goal of cross-validation is to prevent a method from overfitting and to assess the generalization ability. Overfitting happens when a method fits to closely to a specific training dataset and is not able to generalize well to unseen examples. Cross-validation can fight this problem by training the parameter estimates on different batches of training examples of a dataset. This ensures that the parameters are learned on different sets are better trained in generalizing.

The K-fold cross-validation is used for all experiments presented in this thesis and the K-fold cross-validation method from the Scikit-learn library[19] was used. K-fold cross validation uses a single parameter k, which refers to the number of groups that a given sample is to be divided into. The value k was set to 5, meaning that the training, validation, and test data is shuffled five times. Notify that the training dataset is used for training the method, the validation set for tuning the hyperparameters and the test set for evaluating the method.

4.3.2 Confusion matrix

A confusion matrix is a simple way of obtaining an overview of how well a method is performing on a classification task. Different metrics can be obtained form a confusion matrix such as,

[18]https://scikitlearn.org/stable/modules/generated/sklearn.linear_modelLogisticRegression.html
[19]https://scikit-learn.org/stable/modules/cross_validation.html

accuracy, macro-recall, macro-precision, and F1-measures.

Figure 13 shows an example of a 3x3 confusion matrix where the columns are the predictions and the rows the actual values and the diagonal visualizes the correct predictions. The total number of test examples per class can be summed by counting the values of each row. Looking at the first row of figure 13, the most left square shows the number of emails that were actually labeled as Other and predicted as Other, also known as true positive predictions. The second and third square of the first row are false negatives predictions, since the actual label should be Other and were incorrectly predicted with Reminder or Invoice. The sum of the values in the corresponding row, excluding the true positive, is the number of false negative predictions for a class. The total number of false positives predictions for a class is the sum of values in the corresponding column, excluding the true positive prediction.

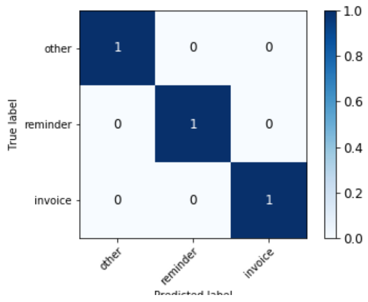


Figure 13: Confusion matrix

4.3.3 Evaluation matrices

Accuracy The accuracy score describes the overall effectiveness of a method. The accuracy score can be a misleading score when deploying multi-class predictions and the accuracy score can be misleading as well when using a imbalanced dataset. This is both the case for the dataset of Company X. The accuracy score can be misleading, because true predictions of the majority class can have a great impact on the value of the accuracy score, while showing a lot of wrong predictions on the minority classes. Therefor, the accuracy score should be used cautiously and alongside other matrices like precision and recall. The formula of the accuracy score is as follows:

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{17}$$

For calculating the precision, recall and F1 score of multi-class problems, two different options can be considered, micro-averaging and macro-averaging. Macro-averaging reduces the multi-class predictions to multiple binary predictions and eventually average these binary predictions by the number of classes. Micro-averaging treats the entire set of data as an aggregate result and calculates one metric instead of averaging the matrices. In this thesis is the macro-averaging score is used for calculating the recall, precision and F1.

4.3.4 Macro-recall

Recall is also known as the completeness and refers to the percentage of total relevant results correctly classified by the classifier. This means that it measures how many examples that should be classified to a specific class, are actually classified to that class. The formula for calculating the recall is:

$$Recall = \frac{\sum_{l=1}^{L} TP_l}{\sum_{l=1}^{L} TP_l + FN_l} \tag{18}$$

Macro-recall considers each class as a binary classification and gives equal weights to each class. The specific case of Company X uses three different classes and thus the the macro-recall divides the three obtained recall scores by three:

$$Macro - recall = \frac{recall1 + recall2 + recall3}{3} \tag{19}$$

4.3.5 Macro-precision

Precision is the ratio of correctly predicted positive observations of the total predicted positive observations. This means that it measures how many of the predicted labels of a particular class, should be actually labeled by the label of that particular class. The formula for calculating precision is:

$$Precision = \frac{\sum_{l=1}^{L} TP_l}{\sum_{l}^{L} = 1TP_l + FP_l} \tag{20}$$

A high score on the precision metrics relates to a low false positive rate. Macro-precision gives equal weights to each of the three classes of Company X and the formular looks like:

$$Macro - precision = \frac{precision1 + precision2 + precision3}{3} \tag{21}$$

4.3.6 Macro-F1

The F1 score is the weighted average of the recall and precision scores. Therefore, it takes both the false negatives and false positives into account. The F1 score is not as easy to understand as the accuracy score but it is more useful to than accuracy when dealing with an imbalanced dataset. The equations for calculating the F1 score is:

$$Macro - F1 = 2(\frac{macro - precision \times macro - recall}{macro - precision + macro - recall} \tag{22}$$

5 Results

This chapter presents the results of the experiments from the experimental design. The findings make it possible to eventually offer an answer to the research questions investigated in this thesis, namely *What is the impact of contextual word embeddings compared to traditional word embeddings on the performance of email classification?* and *How do embedding models trained on different domains impact the performance of email classification?*.

This section starts by exploring the impact of the imbalanced dataset and the pre-processing steps on the performance of TF-IDF (Section 5.1). Experiments were conducted on the traditional and contextual methods using pre-trained embeddings on a general domain (Section 5.2). In addition, the performances of traditional and contextual methods were evaluated when using pre-trained embeddings on the task-specific domain (Section 5.3). Finally, all results presented from Sections 5.1, 5.2, and 5.3 were analyzed, including the strengths and weaknesses of the different methods.

It should be noted that the approaches used for classifying the emails are named after their representation methods. TF-IDF, Word2Vec, and FastText are representation methods that, in contrast to BERT and XLNet, do not contain the built-in classifier. Therefore, when referring to TF-IDF, Word2Vec, and FastText, the representation method as well as the classifier are actually meant.

5.1 Baseline

This section explore the impact of the imbalanced dataset on the performance of the TF-IDF, as well as that of the different pre-processing steps used for cleaning the text. The results of this examination inform the choices made regarding the rest of the experiments.

5.1.1 Imbalanced dataset

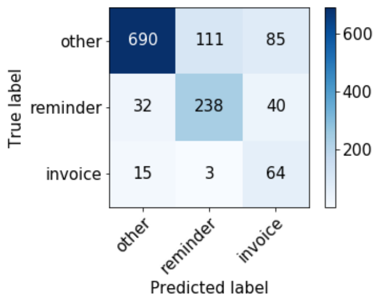
As can been seen in table 1, the dataset of Company X is highly imbalanced. The under-represented class Invoice contains a total of 691 emails while the majority class Other contains 4,593 emails. Considering a dataset that does not represent all classes equally may train a classifier that over-fits to the majority class and becomes oblivious to the existence of the minority class. The results might yield a good overall accuracy, but this score could be overly optimistic when inspecting the other matrices such as macro-recall, macro-precision and macro-F1. Hence, these matrices aggregate the contribution of all classes. The goal of these experiments is to inspect the performance of the TF-IDF using an imbalanced and balanced dataset, and determine whether the classifier shows skewness to one of the classes on the different datasets.

As explained in Section 4.2.1, under-sampling is deployed for creating a balanced dataset. Under-sampling is only applied on the training dataset; the test dataset remains the same imbalance. Table 6 presents the results of the different experiments. As can be observed, the accuracy score obtained using the imbalanced dataset is significant higher compared to the accuracy score using the balanced dataset. In contrast, the macro-F1 score of the balanced dataset is significantly higher.

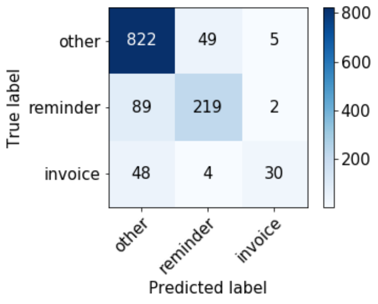
Method	Accuracy	Macro-precision	Macro-recall	Macro-F1
TF-IDF balanced training dataset	0.78	0.78	0.78	0.78
TF-IDF imbalanced training dataset	0.84	0.67	0.82	0.67

Table 6: Measurements of baseline method TF-IDF

Considering the confusion matrix in figure 14(a), the classifier does not seem to be skewed to one of the classes. This is in contrast to the confusion matrix presented in figure 14(b), where it can be seen that the classifier is skewed to the class Other. In this case, the classifier is constrained by its ability to predict minority classes. The significant higher accuracy score for the imbalanced dataset is the result of accurately predicted emails belonging to the Other category. The scores of the macro-F1 is the weighted sum of the macro-precision and macro-recall. Considering the macro-precision and macro-recall scores presented in table **??** for both experiments, the results of the TF-IDF using an imbalanced training dataset indicate a classifier that is not very exact. Due to its high accuracy score and low macro-precision score it seem that it classifies perhaps makes many accurate predictions concerning the majority class, but fails in predicting the minority class.



(a) Balanced



(b) Imbalanced

Figure 14: TF-IDF method with different train dataset

Another way of visualizing the performance of the two experiments is using a precision-recall curve. The precision-recall curve summarizes the trade-off between the recall and precision score and is an appropriate method for comparing methods when dealing with an imbalanced training set. Figure 15 shows the visualization of the precision-recall curve. An optimal curve of precision-recall would lay in the upper-right-corner and the figure shows an inferior performance of the experiment using the balanced training dataset.

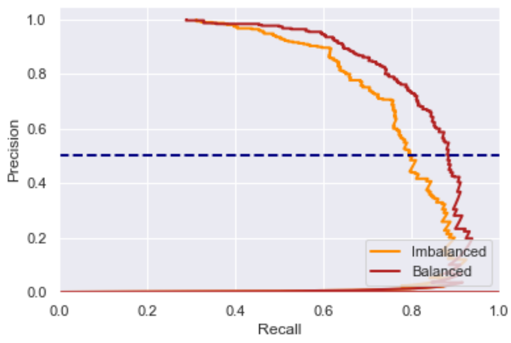


Figure 15: Precision-recall curve

What is learned from these results is that the classifier is not skewed to a specific class when using a balanced training dataset. Thus, choosing the balanced dataset would be a viable option, as using this dataset would positively impact the predictions of the minority classes. However, this would require discarding a great deal of potentially valuable information regarding the majority classes. What should be decisive for what type of dataset should be used is what Company X values the most. Company X may attach more value to either the overall accuracy score on the correctly predicting the emails or predicting whether emails are Reminders. As mentioned previously, company X's ultimate goal is to sent a push message about outstanding invoices that were identified as falling into the Reminder category. This would make correctly predicting the category Reminder to which Company X's emails belong more valuable.

Nevertheless, it was decided to use the imbalanced dataset in this thesis for the rest of the experiments, because creating a balanced dataset is not considered as an option. Under-sampling requires discarding many emails, the labeling of which was extremely time-consuming.

5.1.2 Pre-processing

The different pre-processing steps utilized in this thesis were translation, removing headers/footers, masking confidential information, removing stop-words, and cleaning the data from punctuations marks and other symbols.

To test the impact of the various steps, all combinations were trained, which resulted in a total of 120 runs (24 times five cross-validation splits). The macro F1 score is used for evaluating the performance of TF-IDF because it represents the weighted average of the scores for the macro-precision and macro-recall and is packed into one value. Figure 16 below shows the different pre-processing steps on the x-axis and the macro-F1 score on the y-axis. The blue lines illustrate the average performance of the TF-IDF method when not utilizing the pre-processing step for cleaning the text (false)., while the orange lines show the average performance of TF-IDF when incorporating the pre-processing step (true).

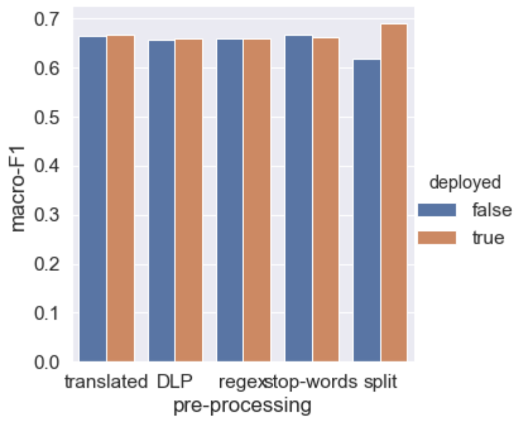


Figure 16: Macro-F1 score of pre-processing steps on TF-IDF

As shown in figure 16, except for the step of removing footers and headers, no great improvements to the macro-F1 score of the TF-IDF method were observed when including the pre-processing step. Cutting headers and footers yields an improvement of the macro-F1 score by 8%. As expected, removing stop-words does not contribute to a better performance of the TF-IDF model. The formula underlying the TF-IDF is intended to reflect how important a word is to a document in a collection of documents, and it thus adds less weight to stop-words.

Stop-words may have a negative impact on the performance of the Word2Vec and FastText methods since they are unable to distinguish the importance of each word in a document compared to all other documents. For these methods, it is more difficult to determine which words hold higher values than others. For the contextual methods BERT and XLNet, stop-words were expected to add more information to the text. At the very least, they were not expected to harm the performance because these methods can learn which stop-words are not useful and do not add redundant attention weight to them[46]. However, the stop-words were kept into the dataset, because BERT and XLNet are the most promising method.

Furthermore, the dataset of Company X contains a great deal of confidential information, such as invoice numbers, invoice amounts, and names. Hence, in this thesis, it was necessary to conceal confidential information as a pre-processing step for all experiments. Even though this step is required, figure 16 shows that its impact on the performance of the TF-IDF method is close to zero. This mean that hiding confidential information does not result in information that might be valuable for the TF-IDF method being discarded.

Finally, cleaning the data of symbols does not seem to have any impact on the performance of the TF-IDF method. Since TF-IDF cannot reflect the location of punctuation or symbols, it does not benefit or suffer from them being included in the text. The other methods proposed in this thesis are able to locate words and they can use symbols to possibly add information and thus better represent a text. However, the text used in all of the other experiments presented in this thesis was cleansed of symbols.

5.2 Pre-trained on general domain

Several experiments were conducted to explore the performance of both traditional and contextual methods using pre-trained embeddings on a general domain. Two existing strategies for applying pre-trained embeddings to downstream tasks will be explored: feature-based and fine-tuning. Note that the traditional embeddings Word2Vec and FastText are by default feature-based methods, and the contextual embeddings BERT and XLNet are by default fine-tuning methods.

Figure 17 presents an overview of the different experiments. To summarize, two experiments with Word2Vec were conducted, one using a feature-based approach and one using a fine-tuning approach. Furthermore, one experiment using FastText was conducted using a feature-based approach. Note that FastText does not apply fine-tuning. Finally, for the two contextual methods BERT and XLNet; supervised fine-tuning was used.

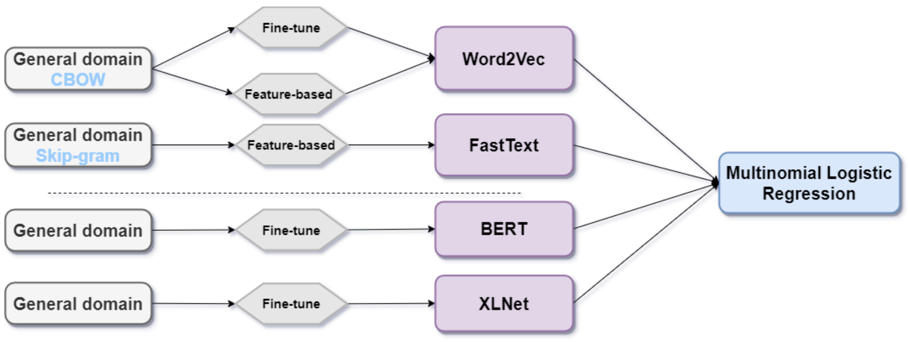


Figure 17: Overview experiments

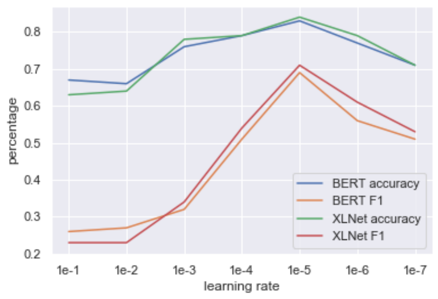
All experiments with the traditional methods Word2Vec and FastText involved hyperparameter tuning of the parameters learning rate, number of epochs, window-size, and the C value of the classifier. This thesis evaluates the estimated performance utilizing a grid-search-based approach that includes cross-validation. ultimately, a total of 960 runs were explored to identify the optimal parameters.

The first experiment with Word2Vec was the feature-based approach. Thus, only the pre-trained embeddings that were trained with a CBOW architecture are deployed, while the weights are frozen while training the classifier from scratch. The best performing parameters for Word2Vec were a learning rate of 0.1, a window-size of 6, 5 epochs, and a C value of 10 for the classifier. The second experiment with Word2Vec used unsupervised fine-tuning of the pre-trained embeddings. The data used for unsupervised fine-tuning was the unlabeled dataset of Company X. Ultimately, optimal parameters were a learning rate of 0.001, a window-size of 6, 15 epochs, and a value of 100 for the C value of the classifier.

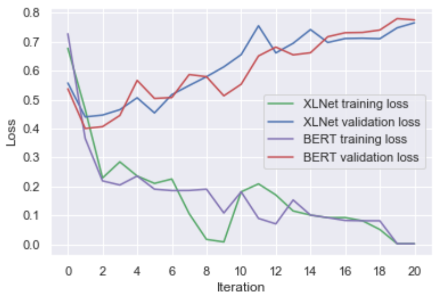
The first experiment with Word2Vec involved the feature-based approach. Thus, only the pretrained embeddings that were trained with a CBOW architecture and that froze the weights while training the classifier from scratch were utilized. The best-performing parameters for Word2Vec were a learning rate of 0.1, a window size of 6, five epochs, and a C value of 10 for the classifier. The second experiment with Word2Vec utilized unsupervised finetuning of the pretrained embeddings. The data used for unsupervised finetuning was the unlabeled dataset of Company X. The eventual optimal parameters were a learning rate of 0.001, window size of 6, number of 15 epochs, and a C value of 100 for the classifier.

The first and only experiment with FastText utilized feature-based extraction. The weights of the pretrained embeddings were frozen to avoid updating them during back-propagation. The best-performing parameters obtained were a learning rate of 0.001, window size of 8, number of 15 epochs, and a C value of 100 for the classifier.

Figure 18 presents the progress in terms of performance of BERT and XLNet on the different hyperparameters' learning rate and number of epochs. It should be noted that these progress scores are the average scores over five cross-validation sets. In terms of the performance of the learning rate (figure 18(a)), optimal performing matrices for both BERT and XLNet are observed around 1e-5. Figure 18(b) shows the loss of the training and validation set over the number of epochs for both BERT and XLNet. The figure shows an increase in the validation loss and a decrease in the training loss after two epochs. When the evaluation loss increases while the training loss decreases, it can be assumed that the model is overfitting the data, which occurs when a model learns the details of the training data to such an extent that it negatively impacts performance. The number of epochs for BERT and XLNet is lower than the optimal number of epochs observed for Word2Vec and FastText. The creators of BERT mention in their paper that the pre-trained embeddings were extensively tuned. The required number of iterations needed for the entire training dataset to pass backward and forward through the network is low.



(a) Learning rate



(b) Epoch

Figure 18: Performance progress BERT and XLNet

Table 7 shows the results of the different experiments. For all measured metrics, the performance of the traditional methods word2Vec and FastText is significantly lower compared to that of the contextual methods BERT and XLNet. Looking at the macro-recall and macro-precision scores of the experiments with Word2Vec and FastText, a significantly lower value for the macro-precision compared to the macro-recall can be observed, which indicates that the classifier is not very exact. Referring to the confusion matrices in figure 20, the classifier of Word2Vec and FastText shows a slight skewness to the majority class Other. This classifier correctly predicts many emails labeled Other as falling into the Other category, but it also predicts many emails labeled Reminder or Invoice as falling into the Other category. Additionally, unsupervised fine-tuning Word2Vec results in an improvement in its performance, but this difference is minor compared to the performance using a feature-based approach. As mentioned previously, Word2Vec is not by default a fine-tuning method. However, since Word2Vec does not provides a solution to out-of-vocabulary words, a positive impact was expected, since the vocabulary is expanded when applying fine-tuning. Since FastText is able to handle out-of-vocabulary words, FastText is expected to benefit less of fine-tuning, even though the benefit for Word2Vec is already minor, FastText might not even benefit from fine-tuning at all.

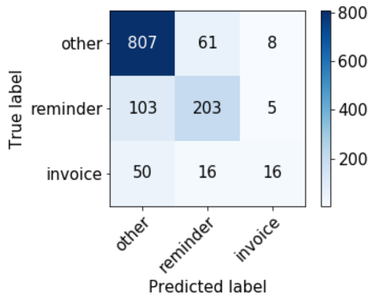
7

Method	Accuracy	Macro-precision	Macro-recall	Macro-F1
Word2Vec CBOW - pre-trained, feature-based	0.80	0.56	0.75	0.60
Word2Vec CBOW - pre-trained, fine-tuned	0.81	0.60	0.72	0.63
FastText skip-gram - pre-trained, feature-based	0.82	0.61	0.71	0.64
BERT - pre-trained, fine-tuned	0.88	0.79	0.80	0.80
XLNet - pre-trained, fine-tuned	0.89	0.82	0.84	0.83

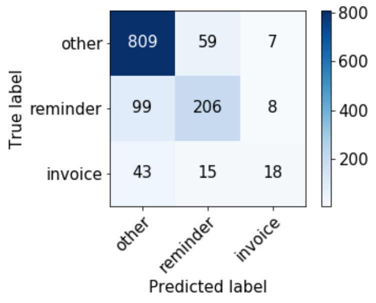
Table 7: Measurements of methods using pre-trained embeddings on general domain

As shown in figure 7, BERT and XLNet show significant better results on all matrices opposed to Word2Vec and FastText. The reasonably higher macro-recall scores and lower macro-precision scores indicate that the classifier is reasonably complete in predicting the total amount of relevant labels that were actually predicted, however it neglects on the fraction of relevant labels among the predicted labels. Considering the confusion matrices of figure 20, one can observe that all methods show a skewness to the class majority class Other by predicting many emails as Other; resulting in many correct predictions of Other, but also a great deal of wrongly predicted emails as Other. However, BERT and XLNet show certainly better performance on the minority classes Reminder and Invoice.

There are three potential reasons for the significantly better performance of the contextual methods BERT and XLNet compared to the traditional methods, with regard to the under-represented classes Reminder and Invoice. Firstly, the contextual methods capture the uses of words across varied contexts and thus capture more in-depth semantic information of the information in a text, which might be useful for the under-represented classes. Secondly, the corpus used for pre-training might have a different distribution than the task set. This is explored in Section 5.3 when further pre-training the embeddings on a task specific dataset. Lastly, fine-tuning the contextual methods have a significant contribution to the performance. The benefits of fine-tuning are explore below.

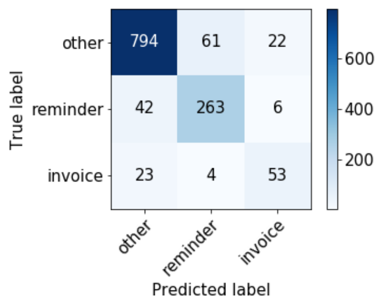


(a) Word2Vec

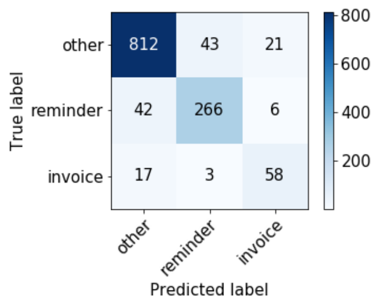


(b) FastText

Figure 19: Confusion matrices - feature-based



(a) BERT



(b) XLNet

Figure 20: Confusion matrices - fine-tune

BERT and XLNet have a layered architecture where different features are learned at different layers. All of these layers are finally connected to the last layer, which is the layer of the classifier. When the encoding layers are frozen, the pre-trained model leverage the weighted layers to extract features but not to update the layers during training. The contribution of each encoding layer of BERT to the metrics measurements is explored, by means of single layer unfreezing.

7

Table 8 shows that only fine-tuning the classifier, and thus applying feature-based extraction, does not achieve performance close to the performance of the traditional methods when applying feature-based extraction or to the performance of BERT while including fine-tuning. This was not expected as BERT was already extensively pre-trained and the embeddings should contain features that are generically useful for different NLP tasks. When looking at the performance of the rest of the 12 attention layers it can be seen that the sixth layer makes the greatest contribution in terms of all metrics measured. This result might indicate two possibilities. Firstly, the lower encoding layers capture a deeper understanding of the semantic relation between words, which is necessary for dissembling the under-represented classes Invoice and Reminder. However, they have difficulty fine-tuning that information for classification. Secondly, when the higher encoding layers are deployed, the model that is trained is larger and perhaps more powerful.

Encoding layer	Loss	Accuracy	Macro-F1
Layer-1	0.71	0.78	0.73
Layer-2	0.60	0.80	0.74
Layer-3	0.58	0.78	0.73
Layer-4	0.54	0.82	0.73
Layer-5	0.58	0.82	0.76
Layer-6	0.51	0.84	0.78
Layer-7	0.49	0.84	0.79
Layer-8	0.50	0.85	0.77
Layer-9	0.47	0.86	0.78
Layer-10	0.48	0.85	0.78
Layer-11	0.44	0.86	0.79
Layer-12	0.43	0.88	0.80
All layers - mean	0.47	0.86	0.80
Classification layer	0.75	0.69	0.27

Table 8: Measurements on different encoding layers of BERT

5.2.1 Insights into pre-trained embeddings

It is often difficult to interpret dimensional vectors of pre-trained embeddings. They are understandable to a computer but not always to a human. However, the use of dimensionality reduction enables visualization of the relationship between words.

T-distributed stochastic neighbor embedding (T-SNE) is a method that reduces the dimensional space and keeps relative pairwise distances between points [47]. Hence, it can be used to explore high-dimensional data.

A Word2Vec representation method creates similar embeddings for words with similar meanings. Figure 21 shows several words from the pre-trained embeddings on the Google News Corpus created by Word2Vec. This figure illustrates the relationship between several words in the vocabulary and provides insight into whether these relationships make sense. The Google News Corpus contains a very large vocabulary features an extensive vocabulary that cannot be clearly presented in a single figure. Therefore, only a selection of words ("invoice," "reminder," "other," "payment," "email," and "company") are shown, along with the 10 words most similar. For example, the word "invoice," the figure shows that the 10 most similar words are almost all related to a payment, such as "refund," "transaction," "remittance," and "disbursement." Furthermore, the 10 most similar words to the other examples all seem to make sense. Another T-SNE figure is shown in the next section (5.3) to illustrate the semantic similarities between words when using pre-trained embeddings on the task-specific domain.

As mentioned previously, it is often difficult to interpret dimensional vectors, so visualizing the semantic similarity provides more insights into whether the pretrained embeddings make sense. Even though the raw embeddings are difficult or impossible to interpret for a human, the reasonability behind the shallow networks of the traditional methods are fairly easy to understand. On the other hand, deep learning models are much more complex, and this topic has been garnering a significant amount of academic attention lately.

Deep learning models lack transparency and reliability, and their predictions are not guaranteed. Transformers are a type of deep learning models, but transformers are commonly believed to be moderately interpretable through inspection of their attention values [48], as shown in table 8. To

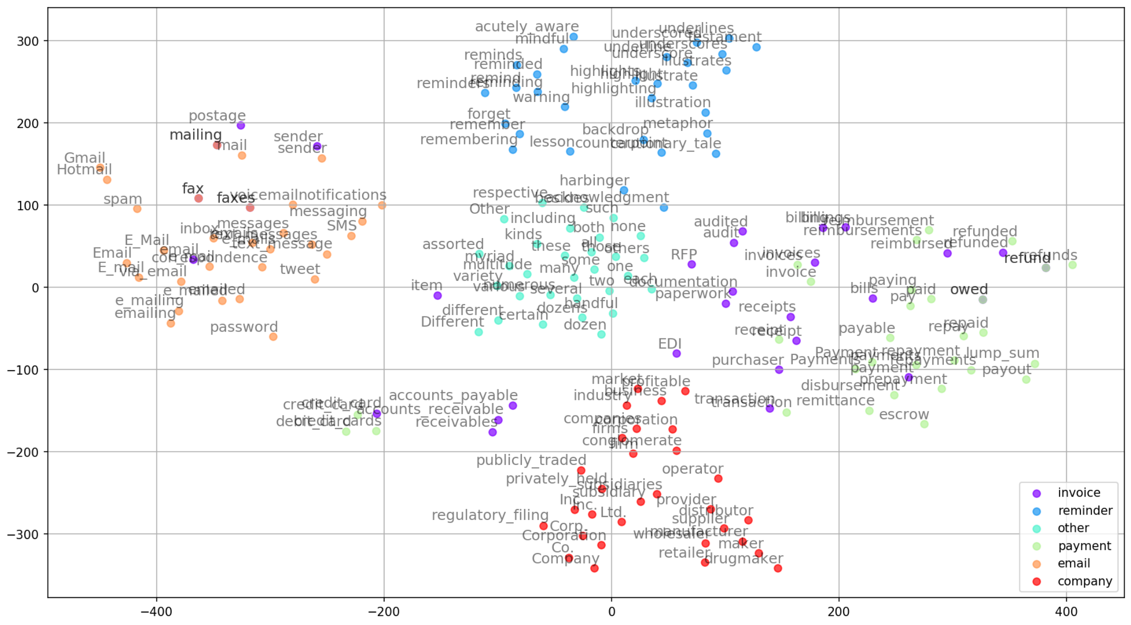


Figure 21: Visualization of the Word2Vec pre-trained embeddings

illustrate the pre-trained embeddings of a transformer model, the context-dependent embeddings must be shown. Figure 22 presents a visualization of the relationship between words in a sentence. As explained in section 3.2.2, BERT separates sentences by tokens and creates a [CLS] token at the start and a [SEP] token at the end of a sentence.

Figure 22 provides examples of different sentences in a two-dimensional space. Notify that the label of the x and y axis are hard to see, but these labels are unnecessary. In the legend there are some sentences framed and these sentences refer to one of the classes Invoice, Reminder and Other. For example, the sentence “This is an invoice to pay” refers to the class Invoice; the sentence “This is a reminder to pay an invoice” relates to the class Reminder, and “This virus affects us all” belongs in the class Other. When one considers the sentences containing the word “invoice,” the figure shows corresponding paths or lines for these sentences. Furthermore, the paths corresponding to the class Other are located higher than the paths corresponding to the classes Reminder and Invoice. Considering this figure, it would be easier to distinguish the class Other from the classes Reminder and Invoice than to distinguish between Reminder and Invoice. Hence, one would expect that BERT predicts a lot of emails as Reminder with the actual label of an Invoice and vice versa. However, considering the confusion matrix of BERT (figure 20(a)), not many of such wrong predictions were made, and instead the class Other was wrongly predicted more often.

5.3 Pre-trained on specific domain

Several experiments were conducted to explore the difference between the traditional representation methods Word2Vec and FastText and the contextual method BERT, when using pre-trained embeddings that were trained on the task specific domain. The specific domain in question is the emails dataset of Company X. Figure 23 presents an overview of the different experiments. The word embeddings of Word2Vec and FastText were trained from scratch using both the CBOW

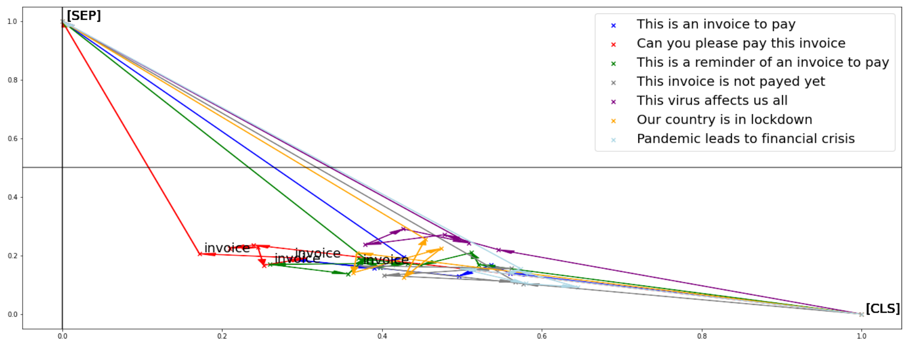


Figure 22: Visualization of the BERT pre-trained embeddings

and skip-gram architectures. BERT uses the pre-trained embeddings of a general domain and further pre-trains them on the task specific dataset. Hence, training BERT from scratch was not realistic. In addition, it was decided to deploy only the feature-based approach for Word2Vec, as fine-tuning did not yield significant improvements and FastText does not apply this option.

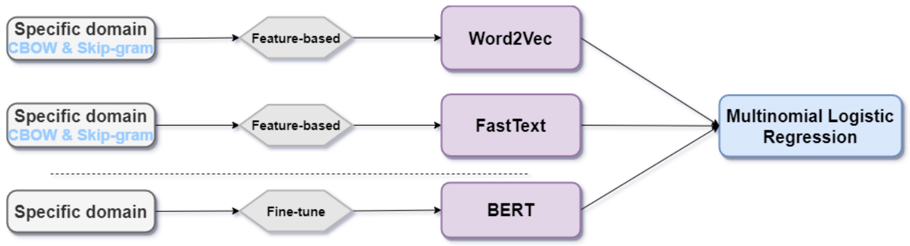


Figure 23: Overview experiments specific domain

Word2Vec was trained from scratch in an unsupervised manner on the unlabeled dataset of Company X. The optimal parameters for training Word2Vec embeddings with a CBOW architecture yield a learning rate of 0.1, a number of 20 epochs, a window of 6 and a C value for the classifier of 10. The optimal parameters using the skip-gram architecture of Word2Vec yield a learning rate of 0.01, a number of 10 epochs, a window of 8 and a C value for the classifier of 100. FastText CBOW architecture yield a best performing learning rate of 0.001, a number of 5 epochs, window of 6 and a C value for the classifier of 10. FastText Skip-gram best performing parameters were a learning rate of 0.001, 15 epochs, window size of 6 and a C value for the classifier of 10.

The hyperparameters learning rate and number of epochs for BERT were duplicated from the previous experiments with BERT. Specifically, a learning rate of 4e-5 and 2 epochs with the remaining parameters being set to their default values. Table 9 presents the performance of the results of the different representation methods on the classification task.

When considering the differences between the two architectures CBOW and skip-gram, which are respectively associated with Word2Vec and FastText, it can be observed that CBOW slightly

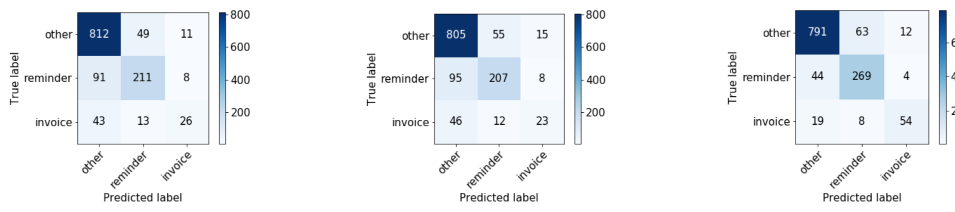
Method	Accuracy	Macro-precision	Macro-recall	Macro-F1
Word2Vec CBOW, feature-based	0.83	0.65	0.75	0.68
Word2Vec Skip-gram, feature-based	0.82	0.63	0.71	0.66
FastText CBOW, feature-based	0.83	0.64	0.73	0.68
FastText Skip-gram, feature-based	0.82	0.60	0.74	0.64
BERT further pre-trained, fine-tuned	0.88	0.83	0.78	0.81

Table 9: Measurements of methods using pre-trained embeddings on specific domain

outperforms the Skip-gram method for both representation methods. This was an unexpected finding, as skip-gram tends to demonstrate better performance with small amounts of training data [1], which was the case in this experiment given that Company X's unlabeled dataset is considerably small. In contrast, CBOW tends to be slightly more accurate in predicting frequent words [1]; this may be an advantage given that the layouts of many emails are similar. When considering confusion matrices a and b in figure 24, one can observe that the CBOW shows slightly better results on all three classes.

When considering the performance of the further pre-trained embeddings of BERT presented in table 9, it can be noted that a significantly higher score is obtained compared to that for the traditional methods. However, it is not completely equitable to compare the traditional and contextual methods among these experiments since BERT utilizes further pre-training of the embeddings on the email dataset. This means BERT can use knowledge that was learned from another task and the traditional methods do not have access to previous learned knowledge. However, training BERT from scratch is not realistic. Another difference between the traditional and contextual methods among these experiments, was that the traditional methods used a feature-based approach and the contextual methods a fine-tuning approach. For comparison between the traditional and contextual methods, it would have been best if the same approach was applied for both techniques. However, FastText does not allow fine-tuning and BERT does not perform properly when using a feature-based approach (as learned from previous experiment with BERT).

Considering the confusion matrices below, figure 24(c) shows a slightly worse performance on the majority class Other compared to Word2Vec (figure 24(a)(b)). However, BERT shows a significant better performance in predicting the under-represented classes Reminder and Invoice.



(a) Word2Vec CBOW (b) Word2Vec Skip-gram (c) BERT+

Figure 24: Confusion matrices

5.3.1 Insights into pre-trained embeddings

The t-SNE was deployed again for visualizing the pre-trained embeddings that were trained on the task specific domain. The same topic words were selected as in the previous section, namely "invoice," "reminder," "other," "payment," "email" and "company." Figure 25 presents the latent spaces of the 10 words that are most similar to these topic words, and all words are visualized together. The figure shows that the data is relatively low clustered, which means that the topics are not clearly separable within the space. Considering the topic words which were clustered based on color, the top 10 most similar words per topic show less semantic similarity than observed while using pre-trained embeddings of a general domain (figure 21). For the topic word "invoice", the semantically closest words are "order," "supplier," "still," "we," "said," "expire." It seems logical that some of these words were mentioned together in an email concerning invoices and thus were trained to have a close cosine similarity. However, the ten most similar words to the word "invoice" obtained from the pre-trained embeddings on the Google News Corpus, were: "refund," "transaction," "remittance" and "disbursement." These words are actually semantically related to the word "invoice." To conclude, even though the pre-trained embeddings of the task-specific domain perform a better performance on classifying the emails, they seem to less accurately identify the semantic similarity. The reason for this latter is that it was trained on the reasonably small dataset of Company X.

It was decided to not visualize the further pre-trained embeddings of BERT, as the results were exactly the same as thoe presented in figure 22.

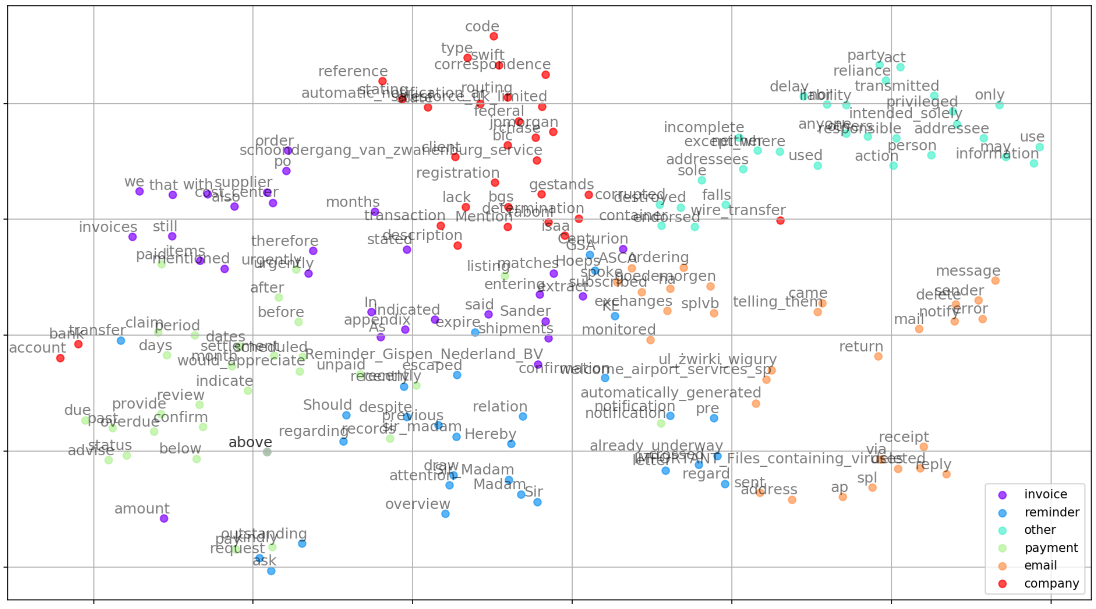


Figure 25: Visualization of the Word2Vec pre-trained embeddings on own domain

5.4 Analysis of the results

In this section, the results of all experiments are compared and analyzed to answer the following research questions: *What is the impact of contextual word embeddings compared to traditional* *word embeddings on the performance of email classification* and *How do embedding models trained on different domains impact the performance of email classification.* While answering these research questions, the performance of the representation methods is compared to that of the baseline method, the TF-IDF. Furthermore, the inclusion of the AutoML tool is considered a best practice, and in this section is determined whether other methods yield results that are comparable, or possibly superior, to those of the AutoML.

Table 10 below, presents an overview of all experiments conducted in this thesis and the results thereof. The top row of the table shows the performance of the TF-IDF. In addition, the results between the first and second dotted line, outlines the performance of the representation methods using pre-trained embeddings on a general domain. Furthermore, the results between the second and third dotted line are the results produced by the representation methods when using pre-trained embeddings on the specific domain. Finally the bottom row shows the results of the AutoML tool.

Method	Accuracy	Macro-precision	Macro-recall	Macro-F1
TF-IDF	0.84	0.67	0.82	0.67
Word2Vec CBOW - pre-trained, feature-based	0.80	0.56	0.75	0.60
Word2Vec CBOW - pre-trained, fine-tuned	0.81	0.60	0.72	0.63
FastText skip-gram - pre-trained, feature-based	0.82	0.61	0.71	0.64
BERT - pre-trained, fine-tuned	0.88	0.79	0.80	0.80
XLNet - pre-trained, fine-tuned	0.89	0.82	0.84	0.83
Word2Vec CBOW, feature-based	0.83	0.65	0.75	0.68
Word2Vec skip-gram, feature-based	0.82	0.63	0.71	0.66
FastText CBOW, feature-based	0.83	0.64	0.73	0.68
FastText skip-gram, feature-based	0.82	0.60	0.74	0.64
BERT - further pre-trained, fine-tune	0.88	0.83	0.78	0.81
AutoML	0.83	0.85	0.89	0.88

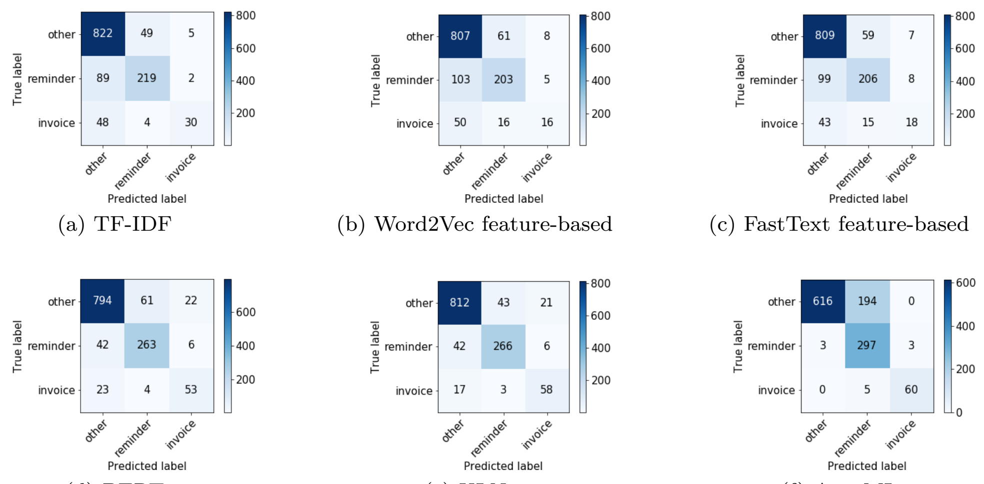
Table 10: Overview statistics of all methods

5.4.1 Answer RQ1

The first research question that this thesis attempts to answer is as follows *What is the impact of contextual word embeddings compared to traditional word embeddings on the performance of email classification.*

Considering all results of figure 10, XLNet showed the best performance on the overall accuracy score with an improvement of +1% compared to BERT, +6% compared to AutoML, +5% compared to TF-IDF and +±6% compared to Word2Vec and FastText. Notify here that the TF-IDF outperforms the AutoML tool as well on the overall accuracy score. However, the significantly higher values obtained by the AutoML tool in terms of macro-recall, macro-precision, and macro-F1 indicate a more precise and complete classifier among the three classes of the AutoML tool. Considering the corresponding confusion matrices of figure 26, AutoML shows superior performances on the classes Reminder and Invoice compared to all other methods. BERT and XLNet show a reasonably good performance on these two classes as well, however, Word2Vec and FastText show a lot of incorrect predictions on the classes Reminder and Invoice. A remarkable find is that the performance of the simple count-based method TF-IDF method is superior to the performances of the prediction-based methods Word2Vec and FastText, even though TF-IDF does not capture any semantics of words. The corresponding confusion matrices (figure 26(a, b, c)) show a greater aberration of Word2Vec and FastText to the class Other, compared to TF-IDF, and they seem to show more difficulties on labeling emails as Reminder when the actual label should be Invoice. Furthermore, the impact of design choices of Word2Vec and FastText, like feature-based compared to fine-tuning and the CBOW and Skip-gram architectures show minor impact on the performance of these methods. However, fine-tuning slightly improves Word2Vec and the CBOW architecture is advantageous to both Word2Vec and FastText on this specific task.

From these results it can be concluded that the contextual methods outperform the traditional methods on the performance of email classification among almost all evaluation matrices.



(a) TF-IDF (b) Word2Vec feature-based (c) FastText feature-based

(d) BERT (e) XLNet (f) AutoML

Figure 26: Confusion matrices among pre-trained embeddings, including TF-IDF

5.4.2 Answer RQ2

The second research question that this thesis attempts to answer is: *How do embedding methods trained on different domains impact the performance of email classification?*

Considering the results of the experiment presented in figure 10, of the traditional and contextual methods using pre-trained embeddings of both a general and specific domain. The traditional methods Word2Vec and FastText show an improvement on the performance when using a task-specific domain compared to using a general domain, even though this improvement is minor. In contrast, no improvement in BERT was identified when comparing the results using pre-trained of a general with the results of pre-trained of a task-specific domain. However, notify here that BERT was further trained on task-specific data and thus BERT had access to knowledge that was learned on a different task which did not apply for the traditional methods. However, training BERT from scratch is not realistic, due to limitation available task specific data.

From these results it can be concluded that the task specific embeddings improved the performance of the traditional methods. In contrast, no improvement was identified on the performance of the contextual method when training embeddings on a general or specific domain. However, the dataset used for further pre-training the contextual method was presumably too small in order to make an impact.

5.4.3 Strength and weaknesses of representation methods

The analysis in this section focuses on determining why certain methods failed to predict the correct label and why other methods succeeded. Therefore, several examples of emails were are presented, along with the actual and predicted labels for each identified by the various methods. In addition, the corresponding confidential values are shown. These confidential values indicate the classifier's certainty in predicting a particular label. Notify that the average of confidential scores was taken when the values of multiple methods were similar.

Example 1 shows the content of an email whose content does not seem to correspond to the actual label, identified by the annotators. This email is referring to a reminder of an invoice and thus should be labeled as Reminder instead of Other. This finding shows that certain emails were still incorrectly labelled despite the fact that sessions were organized with the annotators in which correct labelling practice was discussed. However, since the majority of the email were labeled by only one annotator, there is room for improvement here.

Example 1: `Dear Ms Mr, this is a reminder of a previous sent invoice and a copy of the invoice can be found in the attachment. We look forward to receiving the payment as soon as possible. Sincerely, [FIRST NAME] [LAST NAME]`
Actual label: Other **Prediction all methods:** Reminder (0.74)

Example 2 is an example of an email whose content does not contain useful information. Among all emails with incoherent information, all methods were remarkably consistent in correctly identifying certain emails as belonging to the Other class. The confidential scores for all methods were above 0.70 for these types of emails.

Example 2: `fn` **Actual label:** Other **Prediction all methods:** Other (≥ 0.70)

As indicated previously, Word2Vec and FastText have the most difficulty in predicting emails labeled as Reminder and Invoice, even though there was a slight improvement in the predictions of these classes when using pre-trained embeddings on the task-specific domain. Example 3 presents the content of an email which all traditional methods wrongly predicted as Other when the actual label was Reminder. Example 4 presents the content of an email which was correctly predicted as belonging to the category Reminder by the traditional methods. The confidence score of the traditional methods with regard to predicting the label of example 4 was remarkably high. The traditional methods were very confident in predicting the, initially slightly more complex email shown in example 4, but wrongly predicted the simple email of example 3. A human would very easily be able to determine that example 3 should be labeled as Reminder, because the content of this email immediately refers to a reminder of an invoice. Example 4, in contrast is longer, and it would take a guman slightly more time to determine that this email should be labeled as Reminder.

There might be two reasons for the discrepancy between the predictions of the traditional methods concerning these two emails. Firstly, a possible explanation could be that the email used as example 3 is too brief for the traditional methods to create a reasonable overall idea of its subject. The second reason could be that it depends on the frequency of words indicating a reference to a reminder. Example 3 contains two words that could refer to a reminder of an invoice: “reminder” and “before.” In contrast, example 4 contains six words that could refer to a reminder: “despite,” “yet,” “remind,” “due” and “expired” (twice). The frequency at which these words appear makes it more likely that the classifier would label the email as a Reminder with a higher confidential score.

It should be stated that there are other emails like example 3, of which the traditional methods succeeded in predicting the correct label (Reminder). Thus the explanation presented above does not hold for all these types of emails. However, it did stood out that the traditional methods showed higher confidential scores on emails like that presented in example 4.

Example 3: `Dear [FIRST NAME], this is a reminder of an invoice sent before, see attached. Is it possible to substantiate the payment to send email address? Kind regards`

Actual label: Reminder **Prediction TF-IDF & Word2Vec & FastText:** Other (0.53) (**Prediction BERT & XLNet:** Reminder (0.72)

Example 4: `Dear customer, yes they are being forwarded automatically. I double checked and have sent them to the email address. Despite our email containing the invoices, we have not yet received any payment regarding the invoices below. The payment term has now more than expired and we kindly remind you to pay the total amount of the due invoices within days. After the date of this letter and we have not received the payment from you, within days we will confirm that you have failed to do so. In case you have already paid the outstanding amount you can regard this letter as not being sent. Kind regards`

Actual label: Reminder **Prediction TF-IDF & Word2Vec & FastText:** Reminder (0.93) **Prediction BERT & XLNet:** Reminder (0.86)

Example 5 presents the content of an email that refers to a Reminder of an invoice. The traditional methods all failed in predicting the right label and both the contextual methods succeeded. Notify that this email contains the word ”bank”, which is a polysemous word and has a ambiguous meaning in different contexts. The traditional methods fail to capture the polysemy of words whereas the contextual methods capture the polysemy. Capturing the polysemy of words may contribute to the correct prediction of the contextual methods for this example.

Example 5: `Good afternoon, we checked our bank statement and it seems that the invoice which was sent three months ago was not payed yet. Please have a look at this again. Best, [FIRST NAME] [SECOND NAME]`

Actual label: Reminder **Prediction TF-IDF & Word2Vec & FastText:** Other (±0.58) **Prediction BERT & XLNet:** Reminder (0.73)

Among the most confident prediction scores for both BERT and XLNet for the class Reminder were for emails that referred to an reminder of an invoice of a particular month. Example 6 in an email of this type whereas both XLNet and BERT predicted this email as Reminder with a confidentially score of around 0.91. All traditional methods showed a reasonably high confidence score (at around 0.76) for these type of emails as well.

Example 6: `Hello, please see attached statement of the account for the month of January. Best`

Actual label: Reminder **Prediction TF-IDF & Word2Vec & FastText:** Reminder (±0.76) **Prediction BERT & XLNet** Reminder (±0.91)

Example 7 is a rare example, because this email was actually labeled as Reminder, but was wrongly predicted by TF-IDF, Word2Vec, FastText, and BERT as an Invoice. As presented in the confusion matrices, there are not that many examples of emails as belonging to the Invoice class when predicted as Reminder. However, in this example is XLNet the only method with the correct prediction. Considering example 7, the only word that distinguish the email from an Invoice (instead of a Reminder) is the word "still." The reason why BERT wrongly predicted this email as Invoice might be that BERT masked the word "still" while utilizing masked language modeling, and failed to predict this word based on the context. Since BERT deploys masked language modeling which is an auto-encoding way of language modeling, it neglects the dependencies between masked positions and thus does not include the whole context for predicting these masked words. The advantage of XLNet over BERT was that it combines both auto-encoding and auto-regression modeling by creating all possible permutations of the factorization for predicting a word and thus each word learns to utilize contextual information of the whole context.

Example 7: `Good afternoon, Enclosed you will find our still outstanding invoices. Best, [FIRST NAME] [SECOND NAME]`

Actual label: Reminder **Prediction TF-IDF & Word2Vec & FastText:** Invoice (±0.61) **Prediction BERT** Invoice (0.69), **Prediction XLNet:** Reminder (0.71)

The final example is example 8 below. This email is actually labeled as Other and all methods indeed labeled the email as Other. Considering the difference between Word2Vec and FastText for this email, Word2Vec showed 6 out-of-vocabulary words for this email: "trigger," "arbitrary," "ad," "hoc," "rendering," and "injection". The vectors of out-of-vocabulary words are assigned by Word2Vec with random numbers. In contrast, FastText manages out-of-vocabulary words by treating these words as if it is composed of characters. Even though both methods predicted the right label, FastText predicted this label with a higher confidence score compared to Word2Vec. Handling out-of-vocabulary words is an advantage of FastText over Word2Vec and this was also seen over the experiments that FastText performed slightly better opposed to Word2Vec.

Example 8: `Hello, I should be able to trigger the theme negotiation for an arbitrary ad hoc request and get a proper result back. This may happen and be necessary for rendering theme specific blocks. I was confused by the previous request to route match and injection change already. Speak to you soon, [FIRST NAME]`

Actual label: Other **Prediction TF-IDF:** Other (0.68) **Prediction Word2Vec:** Other (0.53) **Prediction: FastText:** Other (0.76) **Prediction BERT & XLNet** Other (±0.71)

6 Discussion

This thesis aimed to automate the process of classifying emails and provide more insights into the working of the different methods considered. Two research questions were formulated to determine the optimal way to do so. This section provides an overview of the findings and compares them to those of related works. Furthermore, the strengths and weaknesses of the different methods, as well as the decisions that were made, are discussed. This section also discusses the limitations of this thesis and suggests recommendations for future studies.

6.1 Data

The labeled and unlabeled datasets of Company X faces several limitations, which are discussed below. The first limitation is that the labeled dataset of Company X is highly imbalanced, as the class Other contains over six times as many emails than the Invoice class. One of the disadvantages of using an imbalanced dataset is that the classifier learns a skewness toward the majority class and is constrained in its ability to predict the minority classes [49]. This thesis showed that this skewness can results in an overly optimistic view of the accuracy score and that the classifier is consistent in predicting the majority class. The latter was for example observed by the performances of XLNet and AutoML, XLNet showed a greater accuracy score and a better performance on the majority class Other compared to the AutoML tool. However, the AutoML tool was significantly better at predicting the two minority classes, Reminder and Invoice.

Whether an imbalanced training dataset is a problem for the task of classifying emails depends on the preferences of a company. Consider the following example. A retail company employs a support team whose main goal is to answer customer questions via email. The company receives thousands of emails each day, and its goal is to separate the emails containing questions from those containing advertisements (spam). The training dataset is imbalanced as the company receives significantly more emails containing questions than spam emails. The company attaches more value to the completeness of the class of emails containing questions, because these emails must be answered, and doing so requires an additional step. If an email is wrongly identified as spam, the questions of the sending customer's will go unanswered, and this is likely to reduce the customer's satisfaction with the company. This is an example of a case when an imbalanced dataset would not be a disadvantage, as such a dataset would causes the classifier to include many relevant emails among the majority class, which is also the class that the company considers the most important.

An example of a case where an imbalanced training dataset is a limitation for classifying emails is when a company attaches more value to accurately predicting the label of emails from underrepresented classes, indeed, this is the case for the example Company X. The goal of classifying its emails as Invoices, Reminder, and Other is to send push messages concerning outstanding emails for which reminders were previously sent. Therefore, it is most important that the emails whose actual label is Other are labeled as such. It would not necessarily be an issue if emails with the true label of Other or Invoice be assigned the label Reminder. For email belonging to the class Other, probably no invoice data is found, thus no push message sent. Furthermore, for the emails belonging to the class Invoice and idetified as Reminder, probably invoice data is found, however, no outstanding invoice is found in the database since it considers a new invoice.

On the other hand, it is essential to include all emails corresponding to the class Reminder, as these emails are hard to find if they end up in the class Other or Invoice. Hence, Company X prefers that all emails that should be labeled as Reminder be labeled as such, which means setting a high recall score. However, while it is preferred that emails not be incorrectly predicted as belonging to the Reminder category, it would not necessarily be an issue should this occur.

This thesis used the imbalanced labeled training dataset for all experiments even though Company X attaches more value to the minority class Reminder. The reason behind the use of this dataset was creatin a balanced dataset by under-sampling (removing data from the majority class) or oversampling (adding data to the minority class) were not considered good options. Under-sampling entails discarding a great deal of viable information from the majority class. Over-sampling would require creating a syntactic examples of the minority class, which would decreases the variance and causes overfitting of this class. Hence, adding more variance to the minority class was actually the goal of extending this minority class. It would be interesting for future research to counter the effects of the imbalanced dataset by adding more class weights to the minority classes while weighting the loss function.

Another possible limitation associated with the use of the labeled and unlabeled datasets of Company X is that they are both fairly small. This would be a significant disadvantage when employing deep learning models such as a recurrent neural network as these require massive amounts of training data for creating a reasonable estimate model. However, this thesis proposed several methods whichi suppose to work well when limited data is available. The methods Word2Vec and FastText, using a Skip-gram architecture, perform well when limited data is available according to the authors of its paper [25][26]. Furthermore, both methods provide the option of using knowledge obtained from training the embeddings on a general domain. This latter is the same for BERT and XLNet, which are state-of-the-art methods that allow fine-tuning on downstream tasks. Hence, these methods also claim to perform well when limited data is available [3][4].

The last limitation associated with the labeled dataset of Company X is that not all emails were labeled by all three annotators. The reason for this was that labeling is very time-consuming. Even though the annotators were provided with feedback, and the content of many emails and their corresponding labels were discussed, some wrongly labeled examples were found. Machine learning models depend heavily on the accuracy of labels, and therefore it would be recommended to have at least three annotators label all emails.

The last limitation associated with the labeled dataset of Company X is that not all emails were labeled by all three annotators. The reason for this is that labeling emails is extremely time-consuming. Even though the annotators were provided with feedback and the content of many emails and their corresponding labels were discussed,it was possible to identify some wrongly labeled examples. Machine learning models heavily depend on the accuracy of labels, and therefore it would be recommended to have at least three annotators label all emails.

6.2 Pre-processing

This thesis tested the impact of different pre-processing steps on the performance of the TF-IDF method. Because TF-IDF is a count-based method and does not capture any semantic information concerning words, it differs from the other proposed representation methods. Therefore, it does not necessarily mean that the impact of the use of a particular pre-processing step on the performance of the TF-IDF method would necessarily be the same for other, especially since the prediction-based traditional methods and the contextual methods are able to locate words and are possibly positively affected by keeping these symbols in the text. Nonetheless, previous research shows that symbols rarely have an actual positive impact on the performance of machine learning models and do not contribute to creating a general orientation of text [50]. Keeping irrelevant symbols increases the dimensionality, which may negatively affect a classification task [51].

Another pre-processing step, translating emails to a common language, did not affect the performance of the TF-IDF method. An explanation for this lack of improvement could be that TF-IDF is simply not able to understand the various languages, as it only determines the frequency at which a word appears. However, translating the text to another language was expected to have at least some impact on the performance of the TF-IDF method, as it was expected to fail to create a good representative weighting of the words of the underrepresented language. The other methods proposed in this thesis are expected to show a negative impact when training them with two or more languages, as these these methods capture semantics of words.

Furthermore translating text to one common language, can introduce noise, since automatic translation is often approximate [37]. International companies, like Company X receive emails in many different languages. Thus, translating these emails into one common language, may negatively impact the task of email classification. Instead, there is another options listed below for international companies with multilingual emails.

A option for handling multilingual emails is the use of cross-lingual methods. These methods are popular and have becoming increasingly promising due to the rise of transformers. Such transformer combines a dual-encoding architecture and shares knowledge among different languages. The possibility of global knowledge sharing will immensely enrich the information in the knowledge bases and benefit many applications [52]. A cross-lingual version of BERT has already been proposed, which includes a model simultaneously pre-trained on 104 languages that has shown an impressive performance [53].

Besides exploring cross-lingual methods, it would be interesting to explore the performance of the methods when translating the text to other languages than English. Even though translating text may introduce mistakes as mentioned previously, it would be interesting to explore other pre-trained embeddings. For example, Company X is originally a Dutch company and a Dutch version of BERT is recently proposed [54]. Hence, it could be interesting to compare the performance of BERT and BERTje.

6.3 Traditional methods

The traditional methods investigated in this these were separated into a count-based method (TF-IDF) and prediction-based method (Word2Vec and FastText). One of the advantages of the traditional methods over the contextual methods is that the former are simple and easy to train. Furthermore, the prediction-based methods tend to perform equally well on small and large datasets and they are easy-to-scale [55]. Nevertheless, scaling is a problem for the TF-IDF methods, as it addresses the curse of dimensionality.

An advantage of all of the traditional methods is that their representation methods can be trained in an unsupervised manner; therefore, they do not require labeled data for this process. However, note that labeled data was required for these experiments, as the multinomial logistic regression classifier needed to be trained. Despite the robustness of the traditional methods and their ability to capture semantic relations between words (prediction-based method), these methods fail to capture the ambiguous words sense in different context and thus fail to capture the polysemy of words.

The experiments with the traditional methods in this thesis presented a slightly increase in performance when the embeddings were trained on task-specific data. Previous research on impact of task-specific or general domains for pre-training of Word2Vec shown similar results [56]. However, a notable finding in this thesis was the significantly superior performance of the count-based method compared to that of the prediction-based methods. Hence the count-based method outperformed the prediction-based methods in all experiments. The count-based methods are not able to capture any semantic information concerning words whereas the prediction-based methods can capture the semantic similarity between words. However, it seemed reasonable that the prediction-based methods outperform count-based method. Furthermore, previous research has shown that prediction-based methods outperform count-based ones in the task of classifying text [55].

In this thesis, the better performance of the count-based method compared to the prediction-based methods was mainly due to the former being better at predicting the underrepresented classes Reminder and Invoice. There are three potential reasons for this.

The first reason is that count-based methods might be a better option for classifying emails when using an imbalanced training dataset. This thesis explored the impact of a balanced and imbalanced training datasets on the performance of TF-IDF, and the results showed that the

TF-IDF method demonstrated an improved performance on predicting the minority classes when using a balanced training dataset compared to an imbalanced one.

Previous research explored the impact of balanced and imbalanced datasets on the performance on the use of the bag-of-words, TF-IDF, Word2Vec, and FastText methods together with a logistic regression classifier [57]. These studies showed that FastText (and, implicitly, Word2Vec) has a higher dependence on the degree of imbalance when compared to TF-IDF and bag-of-words. The FastText method demonstrated the worst performance when using the most imbalanced training dataset. Hence, the performance increased with a decrease in the imbalance of the classes and the best results were obtained in the experiments using the leas imbalanced dataset. The TF-IDF and bag-of-words methods showed an improvement as a result of using under- and over-sampling to create; however, this improvement was small. The imbalanced dataset might be the reason why the prediction-based methods Word2Vec and FastText performed worse than TF-IDF in all experiments presented in this thesis. It would be interesting to further explore the performance of these prediction-based methods on the task of email classification while using a balanced dataset.

The second reason for the count-based method outperforming the prediction-based methods, might be that the content of emails belonging to the underrepresented classes Reminder and Invoice are very similar, which may be a disadvantage for the prediction-based methods. A count-based method does not capture any semantic information of words, and that prediction-based methods are capable of capturing the semantic similarity between words. For predicting similar classes, it might be preferable to capture no semantics at all than reflecting the semantic similarity of words. Nevertheless, the contextual methods, which aim to incorporate semantic similarity and polysemy of words, do not share the same difficulties as the prediction-based methods in predicting emails belonging to similar classes. Hence, when attempting to predict emails belonging to similar classes, it would seem useful to either capture the polysemy of words and thus develop a deeper understanding of the text being analyzed or to not capture any semantic information concerning words. Capturing high-level understanding of a text, such as semantic similarity between words, can cause difficulties in distinguishing similar classes.

Previous research showed the same problems of prediction-based methods for classifying documents belonging to similar classes. A paper on predicting medical documents [58], showed that methods like Word2Vec and FastText that evalaute words' similarity, fail to distinguish between similarity and relatedness. They provided the following example: "In medical texts, fever and cough might be highly related, but they are distinct symptoms, and thus should be treated differently by a model." This might be the reason that the prediction-based methods could not distinguish between the similar classes Reminder and Invoice.

The third reason for the count-based methods to outperform the prediction-based methods might be the following. Prediction-based methods are representation methods and in order to classify an email, all words of an email must be transformed to one embedding, which eventually enables a classifier to categorize the email. In this thesis, all embeddings representing the words of an email, were averaged to create one embedding. This is a common approach. However, averaging these embeddings may lead to information loss. For instance, the email in the previous chapter, "Good afternoon, Enclosed you will find our still outstanding invoices. Best, [FIRST NAME] [SECOND NAME]" is an example of a reminder, but it was predicted by the prediction-based methods as an invoice. The message "Good afternoon, Enclosed you will find our outstanding invoices. Best, [FIRST NAME] [SECOND NAME]" is very similar with an exception of the word "still." When averaging the embeddings of these examples, they look almost identical, but represent two different classes. Hence, it is challenging for the prediction-based methods to distinguish an invoice from a reminder when using averaging of the word embeddings.

Another solution that would allow the prediction-based methods to represent all words in an email in the form of one embedding would be to use a TF-IDF weighted average embedding, this approach would assign more weight to the absolute values of less common words. Even though the TF-IDF weighting approach of prediction-based methods showed an improved performance on the task of text classification compared to unweighted averaging [59], an important note must be made. The word "still" is considered a stop-word, and the TF-IDF approach would attach less weight to such common words. Thus, the TF-IDF weighting averaging approach may not improve the performance of the prediction-based methods when applied to this email. Nevertheless, it would be interesting to further research whether the TF-IDF weighting of the Word2Vec and FastText embeddings would improve the performance on the task of email classification.

As explained, there are several possible reasons why the count-based methods outperformed the prediction-based methods in this thesis. However, neither of these methods is recommended for Company X for the task of email classification. Company X considers the class Reminder as most important, and all traditional methods have problems identifying emails of this class. For companies who aim to automate the process of classifying emails into very distinct classes, traditional methods are a considerable option. Nevertheless, when the proposed training dataset is imbalanced, this should be addressed and counteracted when using prediction-based methods.

Furthermore, traditional methods have a simpler architecture opposed to that of the contextual methods. Simplicity is often closely linked to interpretability and when the interpretability of machine learning models is highly valued by companies, these traditional methods may be preferred. However, even though these methods are more interpretable they might not be acceptable, since they are not highly accurate and make a lot of mistakes. Therefor, without obtaining a deeper understanding of the text of an email and capturing the polysemy of words, classifying emails will remain a task that is too complex, meaning that traditional methods would be unlikely to be highly accurate when used for this task.

6.4 Contextual methods

One of the main advantages of the contextual methods when compared to the traditional methods is that contextual methods is that the former are able to capture the polysemy of words. Meaning that they capture the different meanings of a word in different contexts. Another advantage of the contextual methods is that they do not require a very large labeled dataset, due to their heavily pre-trained embeddings. The contextual methods outperformed their traditional counterparts in all of the experiments conducted for this thesis. Hence they showed a better performance on the minority classes. This confirmed expectations, as BERT and XLNet are considered state-of-the-art methods for conducting many different NLP tasks, including text classification [3] [4].

As explained previously, the traditional and contextual methods differ in the way in which they are fine-tuned. Word2Vec utilizes unsupervised fine-tuning whereas the contextual methods BERT and XLNet utilizes supervised fine-tuning. The initial concept of this thesis was to compare the traditional and contextual methods together in three different phases, deploying feature-based extraction of pre-trained embeddings on a general domain, deploying feature-based extraction of embeddings trained on a task-specific domain and deploying fine-tuning. However, BERT and XLNet are by default fine-tuning methods and freezing the twelve encoding layers except for the layer of the classifier, did not result in performance close to that of the traditional methods when employing feature-based extraction, nor did it results close performance when including fine-tuning. However, research on the task of questioning and answering has shown that the impact of additional fine-tuning is small compared to using a feature-based approach when the researchers freezed all encoding layers [48]. This was initial confirm expectations, because BERT and XLNet word so well on multiple downstream tasks since the embeddings are heavily trained on a general domain and already contain sufficient information about words and their relations. Thus additional fine-tuning was not expected to have an extreme impact on the performance.

However, other research using BERT for classifying financial papers, reported performance similar to that exhibited by BERT and XLNet in this thesis when freezing all encoding layers and only train the classifier [60]. It seems that only freezing the encoding layers does not provide sufficient information to the classifier. BERT's approach to classification uses a [CLS] token as the input to a dense layer, followed by a multinomial logistic regression classifier. It leverages the hidden states on the [CLS] token on the last layer, and use one linear layer to classify a document [61]. However, this approach may not capture all of the information held in the frozen encoding layers. The classifier might leverage from applying pooling on the hidden states of the encoding layers to capture more features from them and create a weighted combination to feed the classification layer. However further investigation is needed to verify these speculations.

Considering the performance of BERT when further pre-trainnig the embeddings on a task-specific domain. BERT did not show an improvement in its performance while using the further pre-trained embeddings (as opposed to using only the pre-trained embeddings). However, the use of such further pre-trained embeddings of the methods BERT and XLNet is expected to be a promising option with regard to the task of email classification. It is very reasonable to assume that the number of unlabeled emails on which BERT was further pre-trained, was far too small to actually impact the performance. The unlabeled dataset of Company X only contained 3,620 emails, which is an extremely limited amount of text, compared to the massive corpus on which BERT was pre-trained. While Company X receives many emails, the author of this thesis only had access to 3,620 emails. For future research, it might be interesting to further pre-train BERT or XLNet on all the remaining unlabeled emails in the inbox of Company X and determine whether there is a significant improvement. This suggestion would be of interest not only for Company X, but for every company interested in email classification. Large companies receive an extreme number of emails on a daily basis, so the availability of data is not the problem. The only time-consuming task is labeling the emails that are needed for the supervised fine-tuning phase.

Considering the difference between the performance of the AutoML tool and that of BERT and XLNet, the result of the overall accuracy score of the AutoML tool was 6% lower than that of XLNet. However, the AutoML tool was better at predicting the minority classes Reminder and Invoice and showed remarkable results for these classes. Hence, the AutoML tool can satisfy Company X's requirement of accurately predicting which emails belong to the class Reminder. Nevertheless, the AutoML tool operates as a black box and is thus lacking in terms of interpretability. The intepretability of machine learning models is a vital concern in the field of machine learning and has drawn renewed interest. Humans use the predictions of the machine learning model and the extent to which a user can understand and predict a models behavior, impacts his or her trust in a model. Furthermore, if the person who implements a machine learning model understands why that model is making certain predictions, it will be easier for him or her to improve it. The contextual methods BERT and XLNet are more readily interpretable than AutoML as the latter is treated as a black box. However, BERT and XLNet are also commonly believed to be more interpretable than other arbitrary neural networks or random forests with thousands of trees. This is due to the transformer mechanism and the user's ability to interpreted the contribution of each attention layer to the performance of a transformer [48].

In comparison, consider the process of automatically classifying emails at a company: How important is it that the users trust the predictions of the classifier? The medical domain is often taken as example of an area where it is critically important to understand the reasoning behind a prediction, particularly when predictions are not found when they should be [62]. It is not appropriate to compare the interpretability of predictions in a context where lives may be at stake to the interpretability of predictions concerning to which category an email should be assigned. Consider a company that wants to automate the process of email classification. Choosing the right method would require striking a balance between the right "data fit", referring to the predictive accuracy, and "mental fit", referring to the ability of a human to evaluate a method or the results thereof.

Considering what method is recommended to Company X, XLNet showed a better overall accuracy score than the AutoML tool and is more easily interpretable. Furthermore, there are several possibilities for improving the performance of XLNet on the Reminder class, such as tuning hyperparameters (other than the ones already tuned in this thesis), further pre-training the embeddings on email specific data and adding more weights to the class of Reminder.

7 Conclusion

The focus of this thesis has been on exploring the use of different machine learning methods to automate the process of classifying emails. The aim of this study was to gain insights into different representation methods and explore their strength and weaknesses with regard to the task of email classification. Moreover, this thesis has compared the performance of the AutoML tool to that of both traditional and contextual methods. Given the goals of this thesis, the use of the AutoML tool was identified as a best practice due to its previous successes. However, it operates as a black box, which makes it difficult to comprehend the process that the model is based on.

The experiments conducted for this thesis were aimed at identifying the differences between the traditional and contextual methods. Furthermore, this thesis explored how training methods on different domains impacts the performances. Based on the different experiments, it can be concluded that contextual methods outperformed the traditional methods in all experiments on almost all evaluation matrices. In addition, the traditional methods showed a slight improvement when using embeddings trained on a task-specific domain. In contrast, the contextual method did not show any improvement when utilizing further training on a task-specific domain. However, it was expected that the dataset used for the pre-training of the method was too small to make a difference.

The main reason why the contextual methods outperformed the traditional methods was the better performance of the former, on classes that were both the minority classes and very similar. The reason for this superior performance is that the contextual methods capture the whole context within the embeddings due to their bidirectional approach, which enables the methods to create a deeper understanding of the text and captures the polysemy of words.

Even though the traditional methods performed inferior to the contextual methods on the task of email classification, there was a difference between the traditional methods on how they performed on the minority classes. The count-based traditional methods showed a better performance on all classes, however they showed the greatest improvement on the minority classes compared to the prediction-based traditional methods. Three possible reasons for this difference were highlighted in this thesis: Firstly, the count-based traditional methods might be less affected by the imbalanced dataset than the prediction-based traditional methods. Secondly, methods that evaluate words' similarity like the prediction-based traditional methods might fail to distinguish between similar classes. Finally, averaging the embeddings of the prediction-based methods may lead to information loss which makes them fail on distinguishing similar classes.

An essential finding of this research is that the most simple method TF-IDF, and the contextual methods BERT and XLNet showed a better performance compared to the AutoML tool on the overall accuracy. However, the AutoML tool performed best on predicting the minority classes among all experiments.

Determining what method should be recommended to Company X and other companies who intent to implement email classification, depends on two main aspects: how important a company considers interpretability and how important a company considers the performance of a method in terms of accuracy. If a company requires a method to be interpretable before it is acceptable, it would imply that the AutoML tool is no longer considered as an option. Furthermore, if a company attach more value to the interpretability of methods than to their performance, would that suggest the traditional methods are most applicable. Hence, interpretability is often closely linked to simplicity and the traditional methods are the most simple methods deployed in this thesis, particularly the TF-IDF. However, considering the performances of the traditional methods, even though the TF-IDF showed the best performance among the traditional methods, the method still makes a lot of mistakes. Thus, choosing a simple method increases the users ability to understand the reasoning behind the methods predictions, but the user cannot actually rely on the predictions since it makes a lot of mistakes. In the specific case of Company X, the traditional methods are not recommended, as Company X considers correctly identified emails to the Reminder class as most important, and the traditional methods show poor performance on this class.

Based on the experiments in this thesis would XLNet be most recommended to companies who determine to automate the process of email classification, included Company X. Therefore, this method is recommended above the AutoML tool and this is due to the following reasons. The first reason is that XLNet is expected to have the most potential for improvement. XLNet performed best on the overall accuracy score among all methods, and even though the AutoML tool performed best on the class Reminder, the class considered as most important to Company X, XLNet has a lot of potential to improve on this class. Several examples for improving XLNet on the class Reminder are: adding more weights to this class while calculating the loss function, tuning additional hyperparameter or further training the embeddings on the remaining emails of an inbox of a company. The AutoML tool does not allow any adjustments or improvements by its user as it operates as a black box. Therefore, XLNet is considered as having more potential, because despite the method yields already considerably good results, users are able to improve the method based on the preferences of a company.

Secondly, XLNet is commonly believed to be interpretable due to its transformer mechanism. Therefor it enables users to understand the reasoning behind the method's predictions and it does not require the users to blindly trust them, as with the AutoML tool. The importance of interpretability of a method depends on the companies interests. However, from a research perspective, this thesis takes a stand on how important interpretability is for society and the future. Furthermore, academic disciplines needs to be able to collaborate to produce additional knowledge in this field.

References

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

[2] J. Camacho-Collados and M. T. Pilehvar, "From word to sense embeddings: A survey on vector representations of meaning," *Journal of Artificial Intelligence Research*, vol. 63, pp. 743–788, 2018.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[4] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in neural information processing systems*, pp. 5754–5764, 2019.

[5] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.

[6] Z. Alibadi, M. Du, and J. M. Vidal, "Using pre-trained embeddings to detect the intent of an email," in *Proceedings of the 7th ACIS International Conference on Applied Computing and Information Technology*, p. 2, ACM, 2019.

[7] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pp. 242–264, IGI Global, 2010.

[8] B. Klimt and Y. Yang, "The enron corpus: A new dataset for email classification research," in *European Conference on Machine Learning*, pp. 217–226, Springer, 2004.

[9] R. Bekkerman, "Automatic categorization of email into folders: Benchmark experiments on enron and sri corpora," *Computer Science Department Faculty Publication Series*, no. 281, 2004.

[10] W. Cohen, V. Carvalho, and T. Mitchell, "Learning to classify email into "speech acts"," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pp. 309–316, 2004.

[11] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2208–2217, JMLR. org, 2017.

[12] R. Kulkarni, M. Vintró, S. Kapetanakis, and M. Sama, "Performance comparison of popular text vectorising models on multi-class email classification," in *Proceedings of SAI Intelligent Systems Conference*, pp. 567–578, Springer, 2018.

[13] Z. Alibadi and J. Vidal, "To read or to do? that's the task: Using transfer learning to detect the intent of an email," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 1105–1110, IEEE, 2018.

[14] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[15] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[16] R. F. Reinhart and J. J. Steil, "A constrained regularization approach for input-driven recurrent neural networks," *Differential Equations and Dynamical Systems*, vol. 19, no. 1-2, pp. 27–46, 2011.

[17] S. Hochreiter and J. Schmidhuber, "Lstm can solve hard long time lag problems," in *Advances in neural information processing systems*, pp. 473–479, 1997.

[18] O. Melamud, J. Goldberger, and I. Dagan, "context2vec: Learning generic context embedding with bidirectional lstm," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 51–61, 2016.

[19] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, no. Feb, p. 2227–2237, 20018.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[21] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[22] W. Wang, M. Yan, and C. Wu, "Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering," *arXiv preprint arXiv:1811.11934*, 2018.

[23] M. Shi, J. Liu, D. Zhou, M. Tang, and B. Cao, "We-lda: a word embeddings augmented lda model for web services clustering," in *2017 ieee international conference on web services (icws)*, pp. 9–16, IEEE, 2017.

[24] X. Rong, "word2vec parameter learning explained," *arXiv preprint arXiv:1411.2738*, 2014.

[25] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[26] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[27] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[28] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019.

[29] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," *arXiv preprint arXiv:1502.02791*, 2015.

[30] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, pp. 3320–3328, 2014.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[32] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *Advances in neural information processing systems*, pp. 3079–3087, 2015.

[33] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.

[34] A. Truong, A. Walters, J. Goodsitt, K. Hines, B. Bruss, and R. Farivar, "Towards automated machine learning: Evaluation and comparison of automl approaches and tools," *arXiv preprint arXiv:1908.05557*, 2019.

[35] A. F. Hayes and K. Krippendorff, "Answering the call for a standard reliability measure for coding data," *Communication methods and measures*, vol. 1, no. 1, pp. 77–89, 2007.

[36] J. M. Johnson and T. M. Khoshgoftaar *Journal of Big Data*, vol. 6, no. 1, p. 27, 2019.

[37] L. Rigutini, M. Maggini, and B. Liu, "An em based training algorithm for cross-language text categorization," in *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pp. 529–535, IEEE, 2005.

[38] J. Tang, H. Li, Y. Cao, and Z. Tang, "Email data cleaning," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 489–498, 2005.

[39] J. Kaur and P. K. Buttar, "A systematic review on stopword removal algorithms," *Int. J. Future Revolut. Comput. Sci. Commun. Eng*, vol. 4, no. 4, 2018.

[40] I. A. El-Khair, "Effects of stop words elimination for arabic information retrieval: a comparative study," *arXiv preprint arXiv:1702.01925*, 2017.

[41] S. R. Mahmud, "A simple information retrieval technique," *International Journal on Recent Trends in Engineering & Technology*, vol. 8, no. 1, p. 68, 2013.

[42] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[43] C. Drummond, R. C. Holte, *et al.*, "C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling," in *Workshop on learning from imbalanced datasets II*, vol. 11, pp. 1–8, Citeseer, 2003.

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

[45] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR,abs/1301.3781v3*, 2013.

[46] Y. Qiao, C. Xiong, Z. Liu, and Z. Liu, "Understanding the behaviors of bert in ranking," *arXiv preprint arXiv:1904.07531*, 2019.

[47] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[48] B. van Aken, B. Winter, A. Löser, and F. A. Gers, "How does bert answer questions? a layer-wise analysis of transformer representations," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1823–1832, 2019.

[49] V. Ganganwar, "An overview of classification algorithms for imbalanced datasets," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 4, pp. 42–47, 2012.

[50] B. Pahwa, S. Taruna, and N. Kasliwal, "Sentiment analysis-strategy for text pre-processing," *Int. J. Comput. Appl*, vol. 180, pp. 15–18, 2018.

[51] E. Riloff, S. Patwardhan, and J. Wiebe, "Feature subsumption for opinion analysis," in *Proceedings of the 2006 conference on empirical methods in natural language processing*, pp. 440–448, 2006.

[52] L. Pan, Z. Wang, J. Li, and J. Tang, “Domain specific cross-lingual knowledge linking based on similarity flooding,” in *International Conference on Knowledge Science, Engineering and Management*, pp. 426–438, Springer, 2016.

[53] S. Wu and M. Dredze, “Beto, bentz, becas: The surprising cross-lingual effectiveness of bert,” *arXiv preprint arXiv:1904.09077*, 2019.

[54] W. de Vries, A. van Cranenburgh, A. Bisazza, T. Caselli, G. van Noord, and M. Nissim, “Bertje: A dutch bert model,” *arXiv preprint arXiv:1912.09582*, 2019.

[55] M. Baroni, G. Dinu, and G. Kruszewski, “Don’t count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 238–247, 2014.

[56] I. Atanassova, M. Bertin, and P. Mayr, *Mining scientific papers: NLP-enhanced bibliometrics.* Frontiers in Research Metrics and Analytics, 2019.

[57] C. Padurariu and M. E. Breaban, “Dealing with data imbalance in text classification,” *Procedia Computer Science*, vol. 159, pp. 736–745, 2019.

[58] F. K. Khattak, S. Jeblee, C. Pou-Prom, M. Abdalla, C. Meaney, and F. Rudzicz, “A survey of word embeddings for clinical text,” *Journal of Biomedical Informatics: X*, p. 100057, 2019.

[59] S. Gupta, T. Kanchinadam, D. Conathan, and G. Fung, “Task-optimized word embeddings for text classification representations,” *Frontiers in Applied Mathematics and Statistics*, vol. 5, p. 67, 2019.

[60] D. Araci, “Finbert: Financial sentiment analysis with pre-trained language models,” *arXiv preprint arXiv:1908.10063*, 2019.

[61] T. Schuster, O. Ram, R. Barzilay, and A. Globerson, “Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing,” *arXiv preprint arXiv:1902.09492*, 2019.

[62] W. Elazmeh, W. Matwin, D. O’Sullivan, W. Michalowski, and W. Farion, “Insights from predicting pediatric asthma exacerbations from retrospective clinical data,” in *Evaluation Methods for Machine Learning II–Papers from 2007 AAAI Workshop*, pp. 10–15, 2007.