Universiteit Leiden

Master Computer Science

When can BERT beat SVM?
Replication of four text classification papers and fine-tuning RobBERT on Dutch language datasets

Name:	Zhenyu Guo
Student ID:	s2594773
Date:	27/08/2021
Specialisation:	Data Science: Computer Science
1st supervisor:	Suzan Verberne
2nd supervisor:	Peter van der Putten

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

With respect to the recent researches in the NLP domain, BERT (Bidirectional Encoder Representations from Transformers) is obviously a milestone [8]. The excellent performance makes BERT popular in the current NLP (Natural Language Processing) tasks. However, BERT is not perfect in every downstream task. In our project, we investigate two topics: The first topic is to replicate the results of four text classification papers and draw a conclusion on the replication. The second topic is to figure out when BERT can beat SVM . We replicate SVM methods from the text classification papers and explain the reasons if we can not reach the results from the original papers. We fine-tune RobBERT, a pre-trained model based on RoBERTa, on four Dutch language datasets and compare the prediction results with the replication results from some text classification papers. We discuss the experiments and draw a conclusion that the small datasets with quite imbalanced labels are probably more suitable for SVM methods rather than BERT fine-tuning. In addition, the replication results of text classification papers show that it is hard to completely replicate the results of original papers.

Contents

1 Introduction

Nowadays, the fast developing AI technologies provide people with much assistance in daily life. Natural Language Processing (NLP), as a branch of Artificial Intelligence, is composed of computer science and linguistics. It helps computers understand human languages to automatically process the texts with various types of languages. With the explosive growth of data, supervised and unsupervised learning methods are applied more frequently in NLP than ever before to help people not lost in the huge quantity of texts.
From the 1960s to the 2010s, traditional word-featured based learning methods such as SVM and Naive Bayes methods have been widely applied to solve the text classification problems [15]. Since 2010s, deep learning methods develop fast and are extensively deployed in the various industries.
In 2018, a model named Bidirectional Encoder Representations from Transformers (BERT) [8] became the state of the art among all methods on various types of downstream tasks in NLP domain. As the name suggests, BERT is based on Transformers [25], which is an entirely attention-based model used for NLP tasks. BERT is the milestone of transfer learning [17] in the NLP domain. With BERT models, we can directly apply the pre-trained model and fine-tune the model architecture to solve the detailed tasks. Better NLP experiment results can usually be obtained by merely adding a classifier layer on the pre-trained model when fine-tuning BERT models.
However, despite that BERT outperforms other traditional methods in many tasks, it is not perfect in every case. Some researches find that SVM tends to give better results than BERT model [1][10] for small samples. We are curious about the conditions when BERT can beat the baselines such as SVM with word features for downstream tasks. The problem to be discussed are:

- **With what success can we replicate the earlier experiment results reported for Dutch text classification tasks?**
- **In which cases can BERT beat SVM in Dutch text classification tasks?**

Some studies are inspirations for us to investigate BERT and SVM performance on Dutch text classification tasks.
Delobelle et al. [6] use the Dutch Book Review dataset (DBRD) scraped from `Hebban.nl` for evaluating their own pre-trained model RobBERT. RobBERT is a RoBERTa-based pre-trained model customized for the performance enhancement of Dutch NLP tasks. They pre-train the model on a large Dutch corpus (39GB) and then fine-tune the model on DBRD. They compare the binary classification task with other methods including ULMFiT [12], BERTje [5] and BERT-NL [3]. The results indicate that RobBERT model outperforms other methods on this dataset. Inspired by Delobelle et al. [6], we investigate the performance of RobBERT model in diverse domains comparing with SVM concerning Dutch datasets. Delobelle et al. [6] investigate the performance of binary text classification with RobBERT model on the DBRD. However, there are actually 5 rankings (from 1 to 5) of labels for Dutch Book Review dataset. Thus, besides the replication of binary text classification task, we aim to apply RobBERT model on the ordered-class text classification task and contrast the performance with traditional learning method SVM.
Verberne et al. [26] obtain and process the cancer forum data from `Kanker.nl`, which is a Dutch cancer online community. Then they explore their methods on predicting the labels of

cancer forum posts. There are 5 labels in total and they finally choose linear SVC (Support Vector Classifier) as their method due to the best results of label prediction. In this paper, Verberne et al. [26] provide a Dutch cancer forum dataset and the baseline for the multi-label text classification task. What we aim to achieve is to replicate the original experiment and fine-tune the RobBERT model on this dataset to compare the prediction with the results of baseline (linear SVC).
We also choose a dataset [4] in archeology domain, which is processed and created for text classification tasks. This dataset contains over 65,000 excavation files as the train set with two types of main categories, period labels and subject labels. In addition, the test set is manually created and it is similar as the distribution of the training set. The test set contains 100 archeological excavation reports and corresponding labels for the period types and the subject types. We preprocess the raw data and respectively replicate the experiment from [2] and fine-tune RobBERT on these two types of labels. Then we can contrast the performance of RobBERT model with SVM.
Another research from Verberne et al. [27] indicates their effort on topic classification in political domain. They investigate the Dutch political manifestos for year 1986, 1994 and 1998. They digitize all the labels and apply linear SVM for the multi-label classification task. In our project, we plan to investigate the performance of RobBERT model on Dutch political manifestos dataset and contrast the prediction results with the linear SVM.
In general, based on the above 4 Dutch datasets provided by the researchers, we attempt to replicate experiments from original papers and fine-tune RobBERT model on these 4 datasets among completely different domains (book reviews, health, archeology and politics). Then we will compare the prediction results with the baseline Linear Support Vector Machine (SVM) with word features. We attempt to give some explanations behind the experiment results and draw a conclusion on the probable factors on BERT model performance.
The remainder of the paper will be divided into several parts: Section 2 will give introduction concerning the previous relevant research including their problems, methods and experiment results. Section 3 illustrates the details of datasets applied in this project. The size, background and the usage of datasets will be introduced in this part. Section 4 will explicitly describe the models applied in this project. Section 5 will indicates the progress of experiments including data preprocessing, model construction, tuning hyperparameters and the final results display and corresponding discussion. The last section 7 will draw a conclusion on this project and we will attempt to give an answer to the problem mentioned. Moreover, the future works will be discussed in this part.

2 Related Work

Some studies have investigated the performance differences between BERT models and other traditional methods in NLP tasks. González-Carvajal et al. [11] compare the prediction results on IMDB[1]dataset after fine-tuning BERT and training other models such as logistic regression, linear SVC and multinomial NB. BERT outperforms other models at a nearly 94% accuracy. Garcia-Silva et al. [10] concentrate their research on scientific paper classification. They complete expeiments on comparing various types of BERT models and linear classifiers. To their surprise, when it comes to the prediction results of classes with an extremely small amount of examples, the linear classifiers perform better than BERT models. Barbouch et al.[1] explore their model performance on tweet classification. They also mention that SVM performs better

than BERT for the classes with a few tweets. These researches motivate us to investigate the prediction performance of BERT model and SVM on datasets with a strong imbalance.
Saad et al. [19] investigate the performance of linear classifiers on an ordered-class text classification task. They apply Support Vector Regression (SVR), random forest, decision tree and logistic regression on a tweet dataset with 6 level of sentiment labels. Their experiment results indicates that SVR performs best among the chosen models. We are inspired from their research and further explore which method performs better between BERT and SVM on fine-grained sentiment analysis.
Delobelle et al. [6] pre-train the RobBERT model based on RoBERTa architecture with a large Dutch corpus and fine-tune their pre-trained model on sentiment analysis for Dutch book review dataset. Then they compare the results with ULMFiT [12], BERTje[5] and BERT-NL[3]. It indicates that RobBERT performs better when predicting the labels from the Dutch book review dataset (accuracy over 95%). We hope this pre-trained RobBERTa-based model also works well on other Dutch datasets. Thus, in this project, we choose RobBERT model as our primary experiment model to investigate the performance on datasets within various domains.
Van der Burgh et al. [23] investigate the performance of the ULMFiT [12] in contrast to linear SVC on a small Dutch dataset named Dutch Book Reviews where the data are scraped from `Hebban.nl`. Their goal is to study the performance of two models in sentiment polarity classification for a small dataset in Dutch language. The experiment results show that ULMFiT performs better than linear SVM on the same test set with various training sizes in this dataset.
Van Hee et al.[24] investigate the prediction on cyberbullying in English and Dutch language with SVM method. They pre-process the social media text dataset and categorize the data into 7 types of cyberbullying and investigate the best features and hyperparameters on SVM. The results indicate that SVM with appropriate features as input and tuned hyperparameters can be a good choice when facing the Dutch cyberbullying detection problem. Winters et al. [28] investigate the performance of RoBERT, LSTM, CNN and Naive Bayes on Dutch joke detection over three datasets. The results illustrate that the RobBERT model performs best among these methods on detecting the joke texts. Desmet et al. [7] study the automatic text classification on Dutch suicide post detection. They choose SVM as their text classification method and select the best extracted features as the input. Then they train the model and predict if there is suicide tendency in Dutch posts. It turns out that SVM performs well in suicide detection in Dutch posts. These researches focus on Dutch text classification in various domains. Some of them choose SVM as their primary method while some choose neural networks models in text classification. They all prove that both word-featured based method and deep learning method could produce good prediction results on Dutch text classification.

3 Dataset Descriptions

In this section, we will give some descriptions about the four dataset we use in this project. The basic information of the datasets are indicated in Table 1.

[1]IMDB is a movie review dataset containing 50,000 rows of reviews with different labels (positive and negative). This dataset is used for sentiment analysis.

Dataset Name	Instance Count	Classes	Classification Task Type
Book Reviews	22252	4	Binary and ordered-class
Patient forum posts	5332	5	multi-label
Archeological excavation reports	65006	19	multi-label
Political manifestos	2569	335	multi-label

Table 1: The information of the Dutch language datasets

3.1 Dutch Book Reviews

The Dutch book review dataset contains the negative and positive reviews towards Dutch books in `Hebban.nl`. There are 118,516 reviews in total but only 22,252 reviews are balanced in a training set(90%) and a testing set (10%) with equal amount of positive and negative labels. The labels from 1 to 5 represent the sentiment from negative to positive, among which label 3 represents the neutral attitude towards the book. This dataset is initially applied in binary sentiment text classification for testing ULMFiT model [23]. In this project, we will use this dataset for BERT model fine-tuning in ordered multi-class text classification. The RobBERT [6] model based on Roberta model will be applied and fine-tuned in the tasks. There are some differences between an ordered multi-class and a categorical class text classification. Firstly, as the ordered multi-class text classification is regarded as a regression problem, the output class number of it is only one which is different from the case when regarding this task as classification problem. Secondly, the loss function will be converted to mean squared error instead of cross-entropy loss.

3.2 Patient forum posts

The forum post dataset is derived from Verberne et al [26]. This dataset describes the forum posts for peer patients who are affected by cancer. It contains 5332 posts with various types of labels. As the description in Verberne et al. [26], the labels of the forum posts are divided into $Narrative$, $Question$, $Informational\ support$, $Emotional\ support$ and $External\ source$. The label $Narrative$ represents the posts tell their stories. The label $Question$ denotes the posts about the questions for their confusion or difficulties. The label $Informational\ support$ represents forum posts about suggestions or information about the symptoms of various diseases. The label $Emotionalsupport$ represents encouraging posts to other users, The label $Externalsource$ represents the posts that display external links or websites information. The distribution of the labels are indicated in Figure 1. Figure 1 indicates that the posts with $narrative$ label is the majority (3482 posts) while only 753 posts have the label $external\ source$. The amount of posts with other three types of labels $question$, $information\ support$ and $emotional\ support$ are 1318, 1521 and 855 respectively. Each post can be labelled as multiple types and the average label amount for each post is approximately 1.57. We will apply this dataset on fine-tuning a Bert model for the multi-label text classification task.

3.3 Dutch Archaeological Excavation Dataset

This dataset originated from Brandsen et al.[4] contains Dutch archeological excavation report files, in which the amount of raw excavation reports is 65,006. After the removal of the unreadable files, there are 54,871 files remained. Each file has multiple types of time and subject

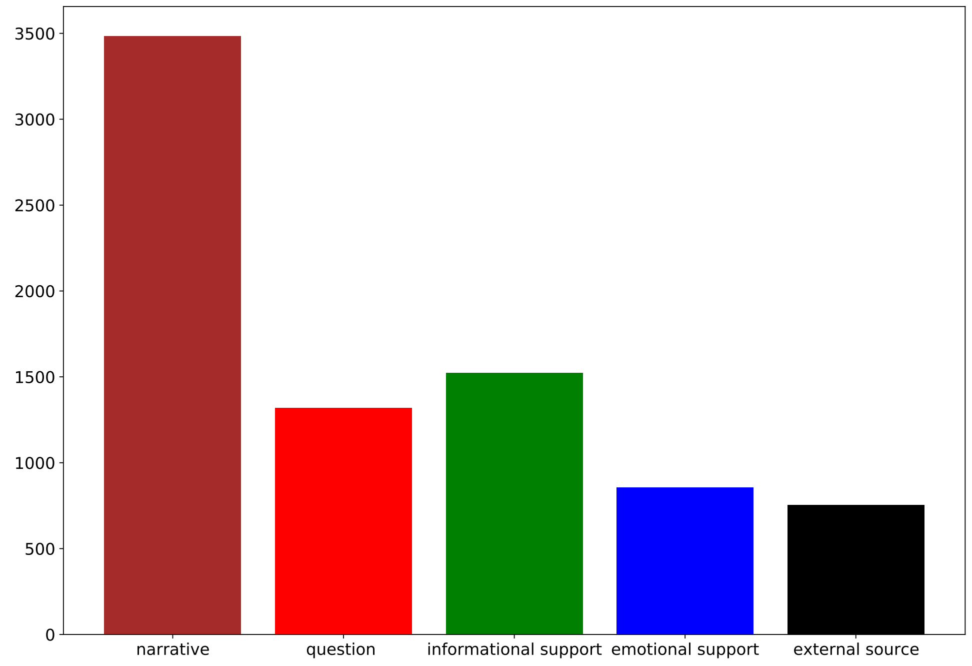

Figure 1: The cancer forum label distribution

labels which represent various time periods and sites of the remains, respectively. These excavation report files are labeled by following the criterion of *Archeologisch Basisregister*(ABR), which is the Dutch archaeological basic register. The ABR records the code for different types of attributes about the archaeological remains. In this raw dataset, there are 11 main subject categories and 8 main time categories in total. Each main category has several sub categories. In our project, we focus our research on the main category prediction. The meaning of subject and time main labels are indicated in Table 2 and Table 3. The distribution of raw training set for time and subject labels are indicated in Figure 2 and Figure 3.
It is manifest that the labels distribution are imbalanced from Figure 2 and Figure 3. So in the data processing section, we will apply some methods to investigate the influence of imbalanced dataset on model performance.
In the raw training set, the labels are not guaranteed to be totally correct. We are not convinced whether the directly separation from the raw dataset will generate the test set with completely correct labels. Thus, the test set is the manually created dataset which contains 100 archeological excavation reports with time or subject labels. The labels are manually annotated by following ABR. The test set has similar distribution with the raw training dataset.
In this project, we plan to investigate the performances of BERT model and SVC on the Dutch archeological excavation report dataset. In addition, we also attempt to study the effect of data augmentation method for balancing data on the prediction performance.

3.4 Political Election Manifestos

This dataset contains three XML files of the Dutch political manifestos for year 1986, 1994 and 1998 created by Verberne et al.[27] Each XML file consists of various types of manifestos with multiple labels. There are over 800 texts and 200 themes on average for each XML file. The manifestos are sorted by the parties so it is clear for classification and convenient to analyze the preference of parties on manifesto themes. The Table 4 indicates the amount of labels

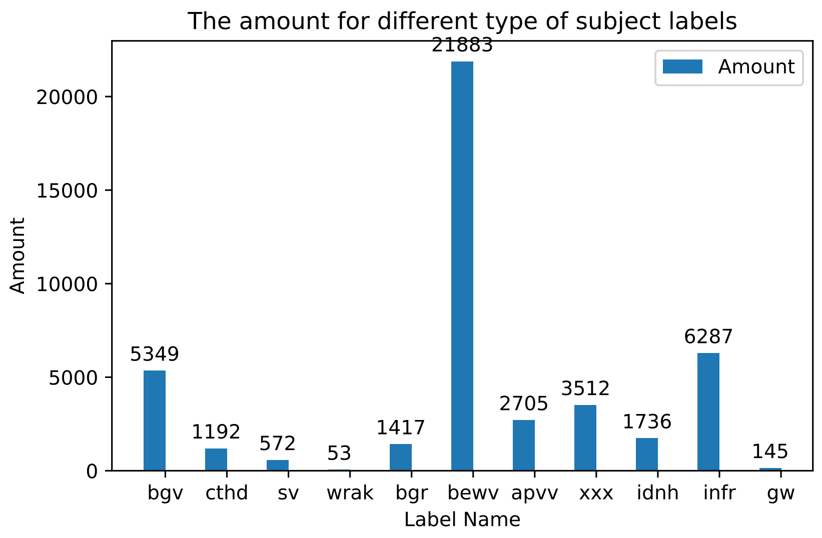

Figure 2: The distribution of subject labels in the raw training set

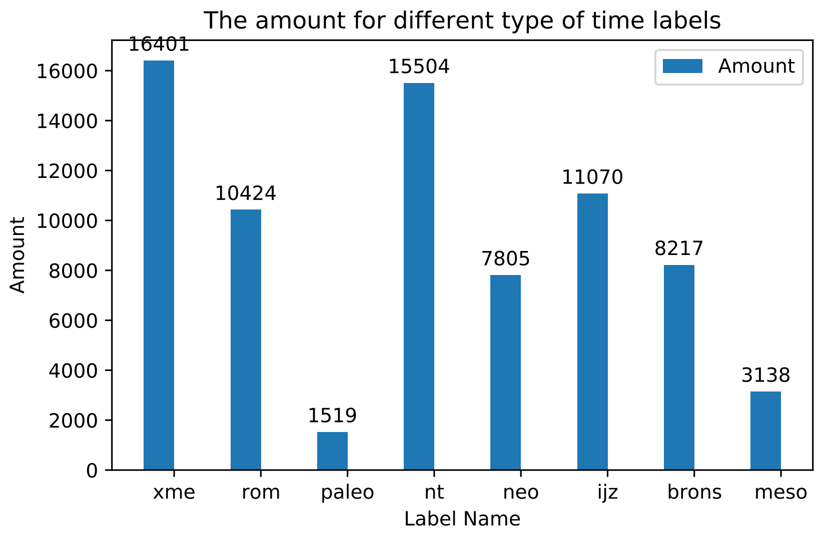

Figure 3: The distribution of time labels in the raw training set

Subject label	Meaning
bgv	burial
cthd	cult or shrine
sv	shipping
wrak	wreck
bgr	grave field
bewv	habitation (including defence)
apvv	agricultural production and food supply
xxx	unkown
idnh	industry and crafts
infr	infrastructure
gw	Raw material extraction

Table 2: The meaning of the main subject labels

Time label	Meaning
xme	Middle Ages
rom	Roman Ages
paleo	Paleolithic
nt	New Time
neo	Neolithic
ijz	Iron Age
brons	Bronze Age
meso	Mesolithic

Table 3: The meaning of the main time labels

Year	Amount of texts	Amount of labels	Avg text length	Avg label amount
1986	797	214	369	10.5
1994	946	217	289	6.9
1998	826	218	301	8.3

Table 4: The basic information of Dutch Political Manifesto dataset

and words for each file from our own statistics. In the third column, Table 4 also indicates average text length and label amount of each year manifestos. Because the manifestos are small paragraphs concerning several topics, the average length of each text is short while the average label amount is relatively high. The Table 5 indicates the data information froom the original paper. From Table 5, we can find that the amount of texts and labels of year 1994 are different with my own statistics. It can probably lead to some differences in the experiment results. In this project, we plan to predict the themes of manifestos for year 1998 by training the data from previous two XML files by using RobBERT model and compare it with SVM method. We will try to implement the replication of LinearSVC from [4] and fine-tune RobBERT model on this dataset to investigate the performance of both methods.

Year	Amount of texts	Amount of labels	Avg text length	Avg label amount
1986	797	214	373	10.5
1994	951	210	284	6.9
1998	826	218	302	8.3

Table 5: The basic information of Dutch Political Manifesto dataset from original paper

4 Methods

4.1 Support Vector Machines

In this project, we will apply SVC and SVR. Both of them belong to the SVM class of models.

- **Linear Support Vector Classifier (Linear SVC) and SVC**

  The SVC will try to determine a max margin hyperplane to separate two classes. The reason for maximizing the margin is to enhance the generalization ability (prevent overfitting). We will apply LinearSVC and SVC (with linear kernel) function from Scikit-learn [18] package. We extract TF-IDF unigram features from the data and convert them to the vectors as the inputs. Hereby we use $TfidfVectorizer$ function as the feature extraction method. We select the max vocabulary size per dataset for the parameter $max\_features$. We use the default parameter settings such as $max\_df = 1$, $min\_df = 1$ and $stop\_words = None$ in the $TfidfVectorizer$ if the parameter settings are not explicitly mentioned in the experiments. We apply GridSearchCV (default 5-fold cross-validation) to search the best parameter C in training the classifier. The parameter C in LinearSVC is for the trade-off between the small and the large margin hyperplane selection. If C is large, then the smaller margin hyperplane will be chosen. In the contrast, the larger margin hyperplane will be selected when C is quite small and some labels can probably be misclassified. After we search out the best C value, we will train the classifier again with the optimized C.

- **Multi-label Classification and Multi-class Classification** Multi-label classification is to categorize the texts with multiple labels. Multi-class classification is to classify texts with single label, but the total categories are more than 2. The One-Versus-All strategy [20] is suitable for multi-class and multi-label classification tasks. This strategy will construct n binary classifiers if there are n classes in total. The chosen class will be compared with other classes. The class with largest prediction probability will be selected as the final predicted class. To solve the multi-label classification tasks, we can also use the Multi Output Classifier. It will train a classifier per target.

- **Support Vector Regression** With respect to the Support Vector Regression, this method is to search a hyperplane which is with the shortest distance to all the data points [9]. This method can be applied when we face ordered-class classification tasks. After training the model, the final outputs will be floats and the metrics is the mean squared error.

4.2 Fine-tuning the RobBERT model

In RobBERT model fine-tuning, firstly, we customize a RobBERT data loader for the model training. We utilize the pre-trained RobBERT tokenizer from Deobelle et al. [6] to encode

the data into four tensor types: ids (inputs ids), mask (attention masks), token type ids and targets (the label vectors). We will use the data with this format as the inputs.
Next, we customize RobBERT model class to build the training model. The initial step is to apply tez [22] module in PyTorch to build the customized model class. In the initialization function, we use the pre-trained RobBERT model structure as the first 12 attention layers. We add a dropout to prevent from overfitting [21]. After that, we add a simple untrained linear classifier from PyTorch (torch.nn.Linear). The feature amount of this linear function is the dimension number of inputs features. The reason why we add a fully connected layer on the top of pre-trained RobBERT model is that the pre-trained model already consists of enough information and we just need a few adjustment for the output features to fit in our own classification task [16].
For the optimizer function, we apply Adaptive Moment Estimation (Adam) [13] optimization algorithm to optimize the model with setting default learning rate= e^{-5}. Many studies prove that Adam optimizer gives solid performance when training the neural networks model.
For the loss function, it depends on the type of classification task. If it is a binary classification task, $BCELoss$ (binary cross entropy loss) function can be applied. If it is a multi-label classification task, we can use $BCEWithLogitsLoss$ function as the loss fucntion. $BCEWithLogitsLoss$ is a combination of sigmoid function and $BCELoss$ function in one layer.
In the evaluation function, we often choose Precision, recall and F1 score as the metrics. We apply the metrics functions from sciki-learn package. This evaluation function can be tuned if we attempt to use other metrics.

4.3 Evaluation Metrics

To clarify the metrics, we will give some assumptions as: there are n predicted labels. TP represents True Positive labels. FP represents False Positive labels. TN represents True Negative labels. FN represents False Negative labels. TP_i represents the i^{th} True Positive category. $Prec_i$ represents the precision of i^{th} label. Rec_i represents the recall of i^{th} label. $F1_i$ represents the F1 score of i^{th} label. The following metrics are used to evaluate the model performance.
Macro Averaging scores Macro averaging scores are used to observe the overall performance of the model and it treats all categories equally.

- **Precision:**
$$Precision_{Macro} = \frac{\sum_{i=1}^{n} Prec_i}{n} \tag{1}$$
- **Recall:**
$$Recall_{Macro} = \frac{\sum_{i=1}^{n} Rec_i}{n} \tag{2}$$
- **F1 Score:**
$$F1_{Macro} = \frac{2(Precision_{Macro} * Recall_{Macro})}{(Precision_{Macro} + Recall_{Macro})} \tag{3}$$

Micro Averaging scores
Micro averaging scores are applied when we aim to treat each instance equally [14].

- **Precision:**
$$Precision_{Micro} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + FP_i} \tag{4}$$

- **Recall:**

$$Recall_{Micro} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + FN_i} \tag{5}$$

- **F1 Score:**

$$F1_{Micro} = \frac{2(Precision_{Micro} * Recall_{Micro})}{(Precision_{Micro} + Recall_{Micro})} \tag{6}$$

Mean squared error: Mean square error are applied in the regression problem. It describes the mean of squared distance between predicted values and true values. Assuming there are n points. $Pred_i$ represents the i^th predicted values and $Test_i$ represents the i^th test set values.

$$MSE = \frac{\sum_{i=1}^{n}(Pred_i - Test_i)^2}{n} \tag{7}$$

5 Experiments

In this section, we will try to replicate the experiment results from Delobelle et al. [6], Verberne et al. [26] [27] and Brandsen et al. [4]. Based on the replication results, we will fine-tune the RobBERT model [6] on the four datasets: Dutch Book Review dataset, Cancer Patient Forum dataset, Dutch Archeological Excavation Report dataset and Dutch political Manifesto dataset. After predicting on the test set, we will compare the performance of RobBERT model and baseline (SVM).

5.1 Experiments on Dutch book review dataset

In this experiment, we will investigate both binary and ordered class text classification on the Dutch Book Review Dataset.
We initially convert the labels to binary categories and employ RobBERT model and SVC (baseline) for the prediction of binary class labels (0 for negative, 1 for positive). Then we fine-tune the RobBERT model and train the model on the original dataset (ranking from 1 to 5) to predict the ratings of the book reviews. As it is a task for ordered class text classification, we regard the task as a regression problem. With respect to the baseline, we use the Support Vector Regression algorithm and employ the Grid Search method to find the best parameters of the SVR algorithm. At last, we will list the results and draw a conclusion on the method comparisons.

5.1.1 Baseline for binary text classification

For the binary text classification, the first step for the experiment is to process the dataset. The original dataset are the reviews with ratings ranging from 1 to 5, among which no reviews are rating 3. In other words, there are no neutral reviews in this dataset. As the ratings are ranging from 1 to 5, we convert the label 1 and 2 to label 0 representing the negative labels. Then we change the label 4 and 5 to label 1 representing the positive labels.
For data processing, we tokenize the texts at first and then lowercase the texts and remove the punctuation and numbers from the texts. After we obtain the processed texts, we apply the texts into the $TfidfVectorizer$ with the default settings ($max\_df = 1$, $min\_df = 1$ and $stop\_words = None$) for the vectorization. We use the max vocabulary size 74,308 of this dataset as the setting of $max\_features$. The vectors and normalized labels are the

input content and labels for the model training. In the model training step, we apply the Grid Search method from $Sklearn$ package to find the best parameters for the SVC method. A set of model settings are indicated as: 'C': [0.1, 1, 10,100,1000], 'gamma': [1,0.1,0.01, 0.001, 0.0001], 'kernel': ['rbf','linear']. As we use 5-fold cross validation method on the train set, we fit 250 candidates and then choose the parameter setting with highest score. It takes 30 minutes to train the model and select the best parameters among the candidates. We choose the best parameters and then predict the test set labels. The metrics are accuracy and F1 score. The prediction results on test set are shown in Table 6.

5.1.2 Fine-tuning RobBERT model for Binary text classification

In this task, we will employ the pre-trained RobBERT [6] model and fine-tune this model for predicting binary labels of test set, thereby aiming to replicate the results reported in Deobelle et al.[6] In terms of label processing, we apply the same method as described in the baseline. Briefly, we normalized the class 1 and 2 to class 0 for representing negative ratings while class 4 and 5 to class 1 for denoting positive ratings. For the subsequent steps, we create a class named $BERTDataset$ to customize the pattern of tokenization and transform encoded data to tensor type.
With respect to tokenization, we simply apply the pre-trained tokenizer from version 2 of RobBERT model. Then we create $getitem$ function to get the items from the encoded text, such as input IDs, token IDs, attention masks and targets.
For the model construction, we apply version 2 of RobBERT model as our pre-trained model. We firstly inherit the module from tez [22] instead of "$torch.nn$" module. It is a simple and clean module to train the PyTorch model. We create $init$ function and $forward$ function which is the same as $torch.nn$ module. Besides the two default functions, we also create a metric monitor and loss functions in the model class. The metrics are accuracy and F1 score. And because the labels are binary, we apply the binary cross entropy loss as the loss function. We use the Adam optimizer to optimize the model. We apply some hyperparameters as given in Delobelle et al.[6]: the learning rate with the level of 10^{-5}, the weight decay at 0.001 and the a small dropout of 0.1. Due to the memory limitation on Google Colab GPUs, we apply smaller batch size with 16 and the epoch amount is 2.
The run time of training model takes around 11 minutes for each epoch. Each epoch contains 1158 batches. The validation loss of the model is around 0.22 and the accuracy and F1 score of final prediction on the test dataset (2224 reviews) are indicated in Table 6. The confusion matrix is indicated in Figure 4. Delobelle et al. [6] also train version 2 of RobBERT model and the accuracy and F1 score both reach 0.951 in their result. In contrast with prediction results from Delobelle et al., the accuracy and F1 score of our prediction are slightly lower. Because BERT models are not deterministic and each run gives slightly different outputs, the results only reflect the performance for one run.

5.1.3 Baseline for Ordered class text classification

With regard to the ordered class text classification, the number of output classes are are essential. In this task, because of the ordinal ratings ranging from 1 to 5, we regard it as the regression problem. As the original dataset contain reviews without rating 3, we normalize the classes to integers from 0 to 3.
For the baseline in this task, we process the texts in the similar way as the baseline in binary text classification. We tokenize the text and then remove punctuation and numbers from the

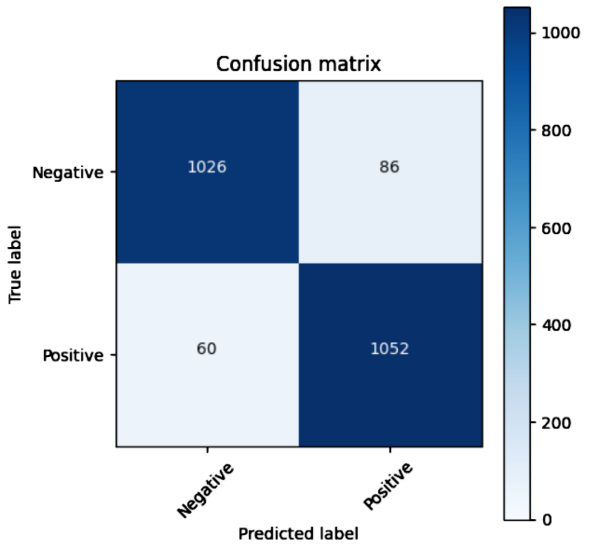

Figure 4: Binary text classification confusion matrix

texts. Then we use $TfidfVectorizer$ with the default settings to vectorize the texts for the inputs of the model training. we attempt to apply Support Vector Regression algorithm for training. In order to obtain more satisfying prediction result, we tune the model parameters by Grid Search method from $Sklearn$. The set of parameters are given as: 'C': [0.1, 1, 10, 100, 1000], 'gamma': [1, 0.1, 0.01, 0.001, 0.0001], 'kernel': ['rbf','linear']. Because we use 5-fold cross validation, there are 250 fits in total. It takes 34 minutes to train the model and the metrics is mean squared error. The best parameters are shown below: Regularization parameter C is 10, parameter gamma is 1 and the kernel is RBF kernel. The results for predicting test labels are shown in Table 6.

5.1.4 Fine-tuning RobBERT model for ordered class text classification

In this task, the first step is to process the labels. We apply the same method to normalize the labels as the baseline mentions. Then we set the number of output features to be 1 because it is a regression problem. We also apply the mean square error as out loss function and metrics. Moreover, we set a boundary rule in the interval [0,3] to alleviate the influence of the output values which are far away from the interval [0,3]. Except for the above transformations, the model construction procedure is the same as the binary text classification.
We set the hyperparameters as: the learning rate is 10^{-5}, the weight decay is 0.001 and the dropout is 0.3. Due to the GPU memory limitation, we can only use the batch size of 8. Epoch number is 4 and 1125 batches are trained for each epoch. It takes 7 and a half minutes to finish one epoch. The mean square error of the prediction on the test dataset is indicated in Table 6.
Table 6 illustrates the results for binary and ordered class text classification on Dutch book review dataset. For the binary class text classification task, we simply regard it as a classification problem. Thus, the accuracy and F1 core are the metrics. RobBERT performs better in both accuracy and F1 score against SVC . Both accuracy and F1 score of RobBERT model are 7 percent higher than SVC. The reason why the replication results are slightly lower than Delobelle et al. is that the batch size setting 16 is smaller than original experiment batch size of 128. The model does not converge to the global optima due to the limitation of GPU

Tasks	Binary Class			Ordered Class	
Methods / Metrics	SVC	RobBERT (ours)	RobBERT[6]	SVR	RobBERT
Accuracy	0.876	0.934	**0.951**	-	-
F1 Score	0.876	0.935	**0.951**	-	-
MSE	-	-	-	0.421	**0.295**

Table 6: Result table for DBRD text classification tasks

	Precision		F1	
Label	LinearSVC [26]	Replication	LinearSVC [26]	Replication
Narrative	89.2	**94.0**	91.1	**94.8**
Question	**87.2**	83.2	**72.7**	71.3
Informational support	75.0	**82.1**	61.4	**72.3**
Emotional support	83.6	**88.6**	73.6	**75.4**
External source	**93.0**	86.8	69.8	**75.9**
Macro Average	85.6	**86.9**	73.7	**77.9**

Table 7: The Precision and F1 on prediction by Linear SVC method from the original paper and the replication

memory on Google Colab.
Concerning ordered class text classification task, we use mean squared error as the metrics since we regard it as a regression problem. RobBERT also performs better in this task. It intuitively indicates that RobBERT model outperforms SVM methods no matter we regard the task as a classification problem or a regression problem on this dataset.

5.2 Experiments on cancer patient forum dataset

In this section, we investigate the replication from Verberne et al. [26] and fine-tuning RobBERT model on cancer patient forum dataset. Our goal is to succeed in replicating the results from the original paper and compare the results with the experiment results that we obtianed from fine-tuning RobBERT model.

5.2.1 The replication of the original paper

To replicate the results from Verberne et al. [26], we firstly follow the data preprocessing steps from the original paper. We lowercase the words and remove the punctuation and remove all the data with empty labels in the dataset. Then we split the dataset into 75% training data and 25% test data. We apply LinearSVC from scikit-learn package and optimize the only hyperparameter C from the interval $[10^{-3}, 10^{3}]$ in the steps of $\times 10$. We use GridSearchCV function to optimize C parameter by a 5-fold cross-validation training. The final optimized C=1.0 which is the same as what described in the original paper. Next, we randomly split the dataset into 80% training data and 20% test data. We choose $TfidfVectorizer$ with the default settings to extract the features from the dataset as our input for the classifier. The parameter $max\_features$ is 27571, which is the same as the max vocabulary size of this dataset. We use LinearSVC function and apply the parameter C=1.0 to train the classifier. The final results are indicated in Table 7.

	Precision		F1	
Label	LinearSVC[26]	RobBERT	LinearSVC [26]	RobBERT
Narrative	89.2	**94.5**	91.1	**93.1**
Question	**87.2**	84.6	72.7	**80.5**
Informational support	75.0	**76.5**	**61.4**	46.1
Emotional support	83.6	**88.9**	**73.6**	73.5
External source	**93.0**	80.6	69.8	**76.9**
Macro Average	**85.6**	85.0	**73.7**	74.0

Table 8: The Precision and F1 on prediction by Linear SVC and RobBERT model

From the Table 7 we can see that the replication results are overall higher than the results reported in the original paper. It is not clear about the reasons why the replication results are higher although we choose the same preprocessing steps and training steps.

5.2.2 Fine-tuning RobBERT model on Cancer patient forum dataset

In this subsection, we will fine-tune RobBERT model to compare the results with those reported in the original paper [26]. For the reference results, we use the results from Verberne et al. [26] as our baseline results. They apply a linear SVC and then use the best parameter setting C=1.0 to obtain the best training results. The prediction results on test set are indicated in Table 8. Then we fine tune the RobBERT model and test it to compare the performance with Linear SVC method.
In this dataset, each post contains one or more labels. So we can primarily process the text labels to a 5 dimension vector for the representation of 5 labels. Then we modify the model based on the model from the multi-class text classification. We convert the loss function to Binary cross entropy with logits loss function. The hyperparameter settings are: the dropout of model is 0.3. We apply AdamW optimizer with learning rate of 10^{-5} as the default settings from the precious experiments.
After training the model for 4 epochs and prediction on the test set, the overall precision and F1 score are 0.879 and 0.792. The results are indicated in Table 8.
From the result Table 8, the macro average precision and F1 score by linear SVC method are slightly higher than RobBERT model. For the prediction in labels, the precision of RobBERT model on predicting label $Narrative$ and $Informational\ support$ are higher than linear SVC. The sample amount is probably a factor because the posts with label $Narrative$ occupy nearly 63 percent and posts with label $Informational\ support$ occupy approximately 27.5 percent in the second place. We can conclude that when the training data for a class is larger, the prediction results of RobBERT model will be better when comparing with linear SVC.

5.3 Experiments on the Dutch Archeological excavation report dataset

In this section, we aim to investigate the performance of the RobBERT model and SVC (baseline) to find out which model performs better. In addition, we also attempt to figure out the factors that have impact on the prediction.

Methods / Metrics	Subject Labels		Time Labels	
	LinearSVC [4]	Replication	LinearSVC [4]	Replication
F1 Score	0.408	**0.490**	0.703	**0.788**
Precision	0.633	**0.637**	0.848	**0.881**
Recall	0.355	**0.432**	0.621	**0.720**

Table 9: The prediction results (Macro-Averaging) for LinearSVC from Brandsen et al. [4] and the replication on the test set of Dutch Archeological Excavation Report Dataset

5.3.1 The replication of LinearSVC method from the original paper

In this subsection, we will try to replicate the SVM method from Brandsen et al. [4]. Therefore, the first step is to clean the raw dataset. As there is some noise within the raw dataset, we clean the data as the following steps which is the same as Brandsen et al. [4]: Firstly, because some files in the raw dataset are are the OCRed files, some files are not converted to text files correctly. We remove all files that are not in text format. Secondly, we remove the files which contains less than 1000 characters. The excavation report within 1000 characters often include few information or the content of this file is lost when scanning the texts. Thirdly, we exclude the files with specific words in their names. The words are $notulen$ (minutes), $bijlage$ (appendix) and $meta$ (metadata). The files with these words in their names contains some irrelevant texts instead of the report content. The processed data will be used as the training data. The test data are manually created by Brandsen et al. [2]. They created 100 excavation reports with their labels to prevent the situation when no labels are annotated in the original dataset or the misannotated data.

Next, we process the texts for feature extraction in the SVC method. We remove the punctuation and convert the texts to lowercase. Then we tokenize the texts and apply the $TfidfVectorizer$ with default settings to extract the features from the data. The parameter $max_features$ in $TfidfVectorizer$ of the subject label classifier and the time label classifier are 559269 and 611643, respectively. We use the max vocabulary size for each label as the $max_features$. We convert the text labels into a one-hot vector for training and predicting. In the training phase, we apply the $LinearSVC$ function in package $sklearn$. We also use $GridSearchCV$ to investigate the best parameters in SVC function which is not mentioned in the original paper.

We choose a set of C values as 0.1, 1, 10 and 100. Since it is a 5-fold cross validation training, there are 20 fits for training the Linear SVC. In our experiment, we apply Linear SVC to train the model on each main category. Therefore, we will train 11 times for subject labels as well as 8 times for time labels. Then we predict test set labels separately with those trained models. The metrics are macro-averaging F1 score, Precision and Recall. The prediction results and the results reported in the original paper are indicated in Table 9.

From the Table 9, we can see that all prediction results for the replication perform better than those reported in Brandsen et al. [4]. The reason for this is that we use GridSearchCV method to find out the best parameter C whilst C is not optimized in Brandsen et al. [4].

With respect to fine-tuning RobBERT model on this dataset, we apply tez [22] in the PyTorch module as the base of our code. We use the RobBERT version 2 for the pre-trained tokenizer and the pre-trained model. Then we set the output class number as the label amount of subject labels (11 categories) and time labels (8 categories), respectively. In order to conduct

	Subject Labels		Time Labels	
Methods / Metrics	LinearSVC	RobBERT	LinearSVC	RobBERT
Macro F1 Score	**0.490**	0.104	**0.788**	0.552
Macro Precision	**0.637**	0.102	**0.881**	0.514
Macro Recall	**0.432**	0.109	**0.720**	0.684
Micro F1 Score	**0.720**	0.515	**0.811**	0.630
Micro Precision	**0.849**	0.410	**0.877**	0.605
Micro Recall	0.625	**0.694**	**0.754**	0.657

Table 10: The prediction results for Linear SVC and fine-tuned RobBERT model on test set of Dutch Archeological Excavation Report Dataset

multi-label classification, we employ $BCEWithLogitsLoss$ (a combination of sigmoid layer and BCELoss in one layer) as our loss function. For the customized model class, we choose $AdamW$ as optimizer with parameter learning rate at 10^{-5} because it will be overfitting when the learning rate is set to be the same as the original paper ($3e-5$). The metrics are set as aforementioned macro-averaging F1 score, Precision and Recall. In the training phase, due to the GPU memory limitation on Colab, we set the batch size to 8 and epochs to 3 to use the GPU memory as much as possible. As we do not have to preprocess the content of texts, we directly use the texts and corresponding transformed labels (one-hot vector) as the inputs. After training, we apply the trained model to predict test set labels. The prediction results are shown in Table 10.
Table 10 implies that LinearSVC performs much better than RobBERT model on archeological excavation report dataset. The probable reason for the performance difference is that these two methods is that training RobBERT on imbalanced dataset can leads to a circumstance that RobBERT model almost only predicts the labels with highest frequencies rather than predicts labels with small scale. In contrast, $OneVsRestClassifier$ for LinearSVC is actually a binary classification method to train one classifier for each class. It will not be influenced that much by the imbalanced label distribution.
The results in Table 10 also show that the micro averaged scores are generally higher than the macro averaged scores. The reason for the performance difference of these two types of metrics is that the macro averaged precision and recall regard classes equally and calculate the mean of precision or recall directly whilist micro scores will regard every instance equally. Thus, when the dataset is imbalanced, the micro scores tend to be higher than macro scores. From the experiments on this dataset, we can conclude that when dealing with highly imbalanced dataset, LinearSVC performs better than RobBERT.

5.4 Experiments on the Dutch Political Manifesto dataset

In this section, we first attempt to replicate the experiment results from Verberne et al. [27]. Next, we will fine-tune the RobBERT model on this dataset and compare the performance of RobBERT model with the baseline in the original paper. This is a multi-label text classification task in the politics domain.

5.4.1 The replication of the original paper

With regard to the replication, we should follow the experiment steps as described in the paper. Therefore, the first essential step is to process the raw data from the XML files. We apply $ElemntTree$ package to extract the corresponding manifesto content and labels from the XML files. After the data extraction, we process the data by converting to the lowercase, removing the punctuation and numbers. We use the $CountVectorizer$ to vectorize the words and set the $ngram$ to (1,1), which represents unigrams. The parameters in the vectorizer function is $min\_df = 2$ and $stop_words = None$. The $max\_features$ is set to the max vocabulary size 33573. As described in Verberne et al. [27], we choose the political manifestos of year 1986 and year 1994 as our training set while we select the data of year 1998 as the test set. The subsequent step is to convert the word labels into one-hot labels. We apply the $MultiLabelBinarizer$ function from $sklearn$ package to turn the labels into one-hot vectors. We simply fit and transform the theme set of year 1986 and 1994 for the training set labels. With respect to the test set labels, We transform them by using the theme set of year 1986 and 1994 political manifestos. Since the topics of political manifestos vary each year, a part of labels in year 1998 cannot be labeled correctly. We calculate the max recall of test set (year 1998 data) to determine the maximum value of the prediction. The max recall is the intersection between length of training set and test set labels divided by the length of test set labels, which is 79.4% in our results. It implies that we should not obtain the predicted recall larger than 79.4%. The maxrecall is lower than 83.1% reported in the original paper [27].
In the training phase, we employ SVC with linear kernel and use $OneVsRestClassifier$ to train the data with multiple labels. We choose $GridSeachCV$ to find out the best parameter C, which is the penalty parameter to trade-off the problem of overfitting and misclassification. The set of parameters C are 0.0001,0.001,0.01,0.1,1,2,10,100,1000 and 10000. There are 10 settings in the parameter set and we apply the 5-fold cross validation in $GridSearchCV$, so 50 fits will be carried out in total. After training the classifier, We find out the best parameter $C = 0.1$, which is not the same as the optimized C=2.0 in the original paper. Then we use the trained classifier to predict the classes of test set.
It is worth mentioning that the descriptions about the prediction function is not clear from Verberne et al. [27]. They used a software named LCS (Linguistic Classification System) to implement SVC function. There are three specific parameters in their experiment, C, maxranks and threshold. C is the penalty parameter and the maxranks is the maximum number of classes assigned to a text. The threshold is described as the classification score, which is not clear. In addition, the LCS (Linguistic Classification System) they used is not an open-source software. The information we obtain from the paper is that the threshold value is ranging from -2 to 2. It is probably the scaled interval for the probability array concerning predicted classes. So we apply the $predict\_proba$ function to predict the probability of the labels for the test set and then scale the array values from [0,1] to [-2,2]. We use the optimal threshold value 0.6 and maxranks 14 from the original paper.
However, the prediction results given in the first row of Table 11 are not as good as the original experiment. Then we optimize the threshold and maxranks to obtain better results. The final optimal results with optimized $threshold$ and maxranks are indicated in the second row of Table 11. Our experiment results are much lower than those of original paper.
We choose another feature extraction method to see if there are some differences on the prediction. We extract TF-IDF features (with ngram=1) from pre-processed texts and then train the SVC classifier as above settings. Then we employ $predict\_proba$ to obtain the probability

Method	Precision(%)	Recall (%)	F1 Score (%)
SVC (threshold=0.6, maxranks = 14)	19.8	60.6	29.8
SVC (threshold=0.2, maxranks = 14)	36.6	37.8	37.2
SVC (original paper [27])	**49.3**	**44.7**	**46.9**
RobBERT model (threshold = 0.17)	10.9	14.6	12.5

Table 11: The predicition results for Dutch Political Manifesto dataset

Table 12: Prediction results based on different settings in SVC method. Feature Extraction stands for the feature extraction functions in processing data. ngrams stands for continuous n items in the texts. For instances, (1,1) represents unigrams in the texts and (1,2) represents unigrams and bigrams in the texts.

Feature Extraction	ngrams	threshold	maxranks	Precision	Recall	F1 Score
CountVectorizer	(1,1)	0.6	14	19.8%	60.6%	29.8%
	(1,1)	0.2	14	36.6%	37.8%	37.2%
	(1,2)	0.6	14	20.8%	60.4%	30.9%
	(1,2)	0.21	14	**36.7%**	40.5%	**38.5%**
TfidfVectorizer	(1,1)	0.6	14	19.9%	**61.3%**	30.0%
	(1,1)	0.22	14	35.3%	39.6%	37.4%
	(1,2)	0.6	14	19.6%	60.7%	29.7%
	(1,2)	0.22	14	35.2%	40.0%	37.4%

matrix for the test set. With the probability matrix, we can scale the values to [-2,2] along with setting the threshold and maxranks to 0.6 and 14, respectively. Again, the results can be optimized by adjusting the threshold value and maxranks. For the ngrams selection, we also choose unigrams combining with bigrams as the features in the $CountVectorizer$ and $TfidfVectorizer$. The prediction results with various settings are indicated in Table 12.
In the Table 12, the settings and results in the first and second row are the same as what indicated in the first and second row of Table 11. For the first, third and fifth and seventh row, we use the same settings on the threshold and maxranks as the original paper. And we change the ngrams and feature extraction function to observe the differences in the prediction results. With respect to the second, fourth and sixth row, we list the prediction results for the threshold with best prediction scores. We regard F1 score as our primary standard on the prediction performance. The settings with the best prediction scores in this table are in the fourth row. However, the F1 score at 38.5% and precision at 36.7% are still way lower than the optimal results in the original paper showed in the third row of Table 11.
The reasons for the replication failure on this dataset can be listed in 2 points. First, the data preprocessing can induce some errors on the texts. In our project, there are approximately 1% to 2% deviation on the amount of vocabulary and total words between our own processing and reported in the original paper [27]. In addition, the text and theme set amounts of political manifestos in 1994 are a bit different from what Verberne et al. [27] record. Second, since we cannot use the software described in the original paper, we simply use SVC classifier in the scikit-learn package. The case of different prediction results usually occurs when using different methods. And we are not sure if we use the same parameters as the original paper.

5.4.2 Fine-tuning RobBERT model on Dutch Political Manifesto Dataset

In this part, our goal is to assess the performance of fine-tuning RobBERT model on Dutch Political Manifesto Dataset. Firstly, We process the data with pre-trained BERT tokenizer and $encode\_plus$ function encapsulated in a class named BERTDataset. The inputs are processed to the torch tensors. For the model customization, we select RobBERT model v2 as our pre-trained model. We add a dropout layer to prevent overfitting and then a single linear layer to classify the labels. With respect to the optimizer, we apply $Adam$ and set a learning rate at 10^5. As this is a multi-label text classification task, we choose $BCEwithLogits$ as out loss function. For the evaluation metrics, we simply use micro F1 score, precision and recall to measure the performance. When training, we use the split the dataset into 80% training set and 20% valid set and apply . we set the training batch size as 16 and epochs as 4. As the training set is not large, it takes around 12 minutes to train the model. We predict the test set labels and the results are indicated in the fourth row of Table 11.
From the given results, we can obviously see the gap between the RobBERT model and SVC classifier. The performance of SVC classifier is much better than RobBERT model. This is predictable because the dataset is not large and there are hundreds of labels in total. This case is like the situation when we investigate the RobBERT performance on Dutch Archeological Excavation Report Dataset. It seems like word featured model performing better than BERT model when dealing with a small and imbalanced dataset with multiple labels.

6 Discussion

In this project, we firstly investigate the replications from the related papers for Dutch language text classification tasks on various domains (book, cancer, archeology and politics). What we learn is that we should know the details about the implementation from the original papers to obtain the similar results. Then we fine-tune RobBERT models on each dataset and compare the performance with the SVM methods to speculate the reasons for the performance gap.
In the first experiment, we replicate the RobBERT model fine-tuning on Dutch Book Review Dataset as a binary text classification. The prediction results we obtain from the replication is slightly lower than what described in Delobelle et al.[6]. The probable reasons for this are that we did not use the same training batch size due to the lack of GPU memory in $Colab$ and the randomization when splitting dataset into training and test sets. We also investigate the SVC performance on this dataset. Table 6 suggests that prediction score of SVC are relatively lower than RobBERT model. Next, we use the whole labels with 5 ratings and regard this task as a regression problem. We use SVR as the baseline and change the output feature dimension to 1 in the RobBERT model fine-tuning in order to predict the ratings instead of classes. In contrast with SVR, fine-tuning RobBERT model on this dataset performs much better. In general, the RobBERT model fine-tuning performs better than SVM methods on this dataset. The balanced dataset and high quality of data content are the probable reasons for the outperforming of RobBERT model fine-tuning.
For the second experiment, we replicate training the LinearSVC classifier on Dutch Hospital Cancer Forum Dataset, which contains 5 types of labels about posts of hospital cancer forum. The replication results are generally higher than reported in Verberne et al.[26]. We also fine-tune a RobBERT model on this dataset and obtain solid results. The overall scores of prediction are slightly lower than what reported in Verberne et al.[26] but the prediction of labels with large proportion performs better than LinearSVC.

Concerning the third experiment, we firstly train a LinearSVC classifier with TF-IDF feature extraction on the Dutch Archeological Excavation Report Dataset. Then we fine-tune the RobBERT model on this dataset. The prediction results of the LinearSVC classifier indicated in Table 10 are much better than RobBERT model. The reason for this case is probably the imbalanced label distribution. The macro scores regard each category equally so when the dataset is extremely imbalanced the macro score will be lower than micro scores. As in LinearSVC, we use multiple binary classifiers to predict the multi-label test set. Comparing with a direct multi-label text classification for fine-tuning the RobBERT model, the imbalanced dataset has smaller influence on LinearSVC due to the different strategy. We choose one-vs-rest strategy to solve the problem that LinearSVC cannot be used in multi-label classification directly. The one-vs-rest strategy will train a classifier per class while fine-tuning RobBERT is to directly classify the multiple classes. From the micro prediction scores, we can still find that LinearSVC achieves better results. We can speculate that in the case of multi-label prediction, SVM methods seem to perform better than the BERT model.
In the fourth experiment, we replicate the experiment from the paper [27] as usual. However, we cannot reach the prediction results without using the same tools. The code used in the original paper are not open-source. We also try various feature extraction method and parameter settings. None of the prediction results can reach those of the original paper. Then we fine-tune RobBERT modle on this political manifesto dataset to observe the performance of this method. The prediction on test set shows that it is not a good method for this dataset. It is worth mentioning that in the third experiment, fine-tuning RobBERT model also obtains much lower prediction results than SVM methods. It proves the speculation that SVM methods performs better than BERT model on the multi-label text classification especially with class imbalance.

7 Conclusion

After the discussion about the four experiments, the answer to the research questions are indicated below.
With what success can we replicate the earlier experiment results reported for Dutch text classification tasks? (Research Question 1)
In this project, we obtain better results than the original papers in two experiments. For other two experiments, we can not achieve the results due to the computation limitation and information missing in the original papers. Thus, if the requirement of the successful replication is not to completely replicate the results from the original papers, for instance, F1 score and precision for all the labels are equal, we reckon that we succeed in replicating the results in two experiments.
There are some thoughts for the replication: To replicate the paper results, we should follow the steps described in the original papers. The pre-processing, parameter settings and training methods should be the same as the original papers. The replication results can be lower sometimes because of the inevitable error such as computational limitation and the information missing concerning data cleaning methods, key parameters and the feature extraction methods from the original papers. Another factor that has an impact on the model prediction is the data quality. If we use the raw data with much noise, it will be hard to replicate the results from the original papers. If the labels are more related to the content and the noise in the data is easy to remove, the features can be extracted more accurately. The results will be close if

we completely follow the paper we replicate.

When can BERT beat SVM in Dutch text classification tasks? (Research Question 2)

From the experiments on Dutch language datasets, we find that when we are facing binary text classification and the dataset is not small or extremely imbalanced, the BERT model can often achieve better results than SVM methods. SVM often performs better than BERT when dealing with multi-label text classification tasks. The instance amount per class in the dataset has more influence on BERT than SVM. BERT model needs more instances to enhance the performance.

In this project, we investigate the feasibility of replications from the original papers concerning the Dutch language datasets and it indicates that we can reach the original results only if the experiment details are completely the same. Then we study the conditions that the BERT model can beat SVM on Dutch language datasets within four domains. The results show that a small amount of labels and a balanced dataset with high data quality are the key points.

Limitations and Future work

In the experiments, we only use the RobBERT model representing BERT models to investigate performance differences with SVM. We can use more BERT-like models to give a more comprehensive perspective on the conditions that BERT models perform better than the SVM methods.

We do not completely replicate the steps from the original papers. This can lead to the deviation of the prediction results. In addition, the computational resource is limited in this project so we should tune some parameters to prevent out of memory on GPU, for instance, the training batch size.

In the future, we suggest others to use other BERT models to investigate the conditions under which BERT models can beat SVM methods. We can also try to balance the data and investigate the performance of BERT models and SVM methods. In addition, we can use more datasets and find out the key attributes in the datasets that have the largest impact on the SVC training and BERT model fine-tuning performance.

8 Bibliography

[1] Mohamed Barbouch, Frank W. Takes, and Suzan Verberne. Combining language models and network features for relevance-based tweet classification. In Samin Aref, Kalina Bontcheva, Marco Braghieri, Frank Dignum, Fosca Giannotti, Francesco Grisolia, and Dino Pedreschi, editors, *Social Informatics*, pages 15–27, Cham, 2020. Springer International Publishing.

[2] Alex Brandsen. alexbrandsen/archaeo-document-classification- dataset: Second version, October 2020.

[3] Alex Brandsen, Anne. Dirkson, Suzan. Verberne, Maya Sappelli, Dung Manh Chu, and Kimberly Stoutjesdijk. Bert-nl a set of language models pre-trained on the dutch sonar corpus. 2019.

[4] Alex Brandsen and Koole Martin. Labelling the past: data set creation and multi-label classification of dutch archaeological excavation reports. *Language Resources and Evaluation*, 2021.

[5] Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. Bertje: A dutch bert model, 2019.

[6] Pieter Delobelle, Thomas Winters, and Bettina Berendt. RobBERT: a Dutch RoBERTa-based Language Model. In *Dutch-Belgian Information Retrieval Conference*, pages 3255–3265, Online, November 2020. Association for Computational Linguistics.

[7] Bart Desmet and Véronique Hoste. Online suicide prevention through optimised text classification. *Information Sciences*, 439-440:61–78, 2018.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[9] Harris Drucker, Chris, Burges* L. Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems 9*, volume 9, pages 155–161, 1997.

[10] Andres Garcia-Silva and Jose Manuel Gomez-Perez. Classifying Scientific Publications with BERT – Is Self-Attention a Feature Selection Method? *arXiv e-prints*, page arXiv:2101.08114, January 2021.

[11] Santiago González-Carvajal and Eduardo C. Garrido-Merchán. Comparing BERT against traditional machine learning text classification. *CoRR*, abs/2005.13012, 2020.

[12] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[14] Ajitesh Kumar. Micro-average & macro-average scoring metrics - python.

[15] Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. A survey on text classification: From shallow to deep learning, 2020.

[16] Chris McCormick and Nick Ryan. Bert fine-tuning tutorial with pytorch.

[17] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] Shihab Elbagir Saad and Jing Yang. Twitter sentiment analysis based on ordinal regression. *IEEE Access*, 7:163677–163685, 2019.

[20] scikit-learn developers. Multiclass and multioutput algorithms.

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[22] Abhishek Thakur. tez module.

[23] Benjamin van der Burgh and Suzan Verberne. The merits of universal language model fine-tuning for small datasets - a case with dutch book reviews. *CoRR*, abs/1910.00896, 2019.

[24] Cynthia Van Hee, Gilles Jacobs, Chris Emmery, Bart Desmet, Els Lefever, Ben Verhoeven, Guy De Pauw, Walter Daelemans, and Véronique Hoste. Automatic detection of cyberbullying in social media text. *PLOS ONE*, 13(10):1–22, 10 2018.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[26] Suzan Verberne, Anika Batenburg, Remco Sanders, Mies van Eenbergen, Enny Das, and Mattijs S Lambooij. Analyzing empowerment processes among cancer patients in an online community: A text mining approach. *JMIR Cancer*, 5(1):e9887, Apr 2019.

[27] Suzan Verberne, Eva D'hondt, Antal Van den Bosch, and Maarten Marx. Automatic thematic classification of election manifestos. *Information Processing & Management*, 50:554–567, 07 2014.

[28] Thomas Winters and Pieter Delobelle. Dutch humor detection by generating negative examples. *CoRR*, abs/2010.13652, 2020.