Language Identification Using Traditional Machine Learning and Deep Learning Methods

Christoforos Spyretos

Linköping University
January 2023
Text Mining Project
Course Code: 732A81

Abstract

In natural language processing, language identification is the problem of determining which natural language given content is in. Computational approaches to this problem view it as a special case of text classification, solved with various statistical methods. *Google Translate*, *DeepL Translator* and *Reverso* are some of the services to identify the language of the text and translate it. In this project, two traditional machine learning models and three neural networks were implemented to classify to which language the text belongs. The models' efficiency and performance were interpreted based on their accuracy, complexity and fitting time. The outcomes showed that the traditional approaches performed better, where a grid search cross-validation was conducted on these models to tune their hyperparameters and improve their performance. The deep learning models also achieved high accuracies, but their predictive results were approximately 10% lower in some cases compared to the traditional methods; thus, they did not reach the expected performance considering their structural complexity and training time.

1 Introduction

Language identification of text has become increasingly important as large quantities of text are processed or filtered automatically for multiple tasks in the industry and commercial sectors. Language detection is the first step towards achieving various tasks, such as detecting the source language for machine translation, improving the search relevancy by personalizing the search results according to the query language and delivering a uniform search box for a multilingual dictionary. Such tasks benefit people who spend much time travelling to multiple countries where they need to become more familiar with foreign languages. It is also essential to many web services, social networking and social media. Thus it is convenient and helpful to use applications to detect the language and translate words and texts. Language identification has three distinct modules to detect language from texts, images, and audio files. This project focuses on text data and creating algorithms that could detect in which language the content of the texts belongs. The models are trained on a dataset containing 17 languages. The rest of the report will overview the related work on language detection. First, the theory section will present a theoretical background of the machine learning and the deep learning models that were used. Followed by the data section, which will give an insight into the data structure, what information contains and the pre-processing of the texts. Then, the method section explains the methodology and how the project was approached. Next, the results section describes the results obtained. Finally, an interpretation of the results and the project's conclusion will be described in the discussion and the conclusion section, respectively.

2 Theory

2.1 Traditional Machine Learning Models

2.1.1 Multinomial Naive Bayes Classifier (MNB Classifier)

Multinomial Naive Bayes (MNB) [1] is a probabilistic learning method widely used in text classification. Given a set of labelled data, MNB often uses a parameter learning method called Frequency Estimate (FE), which estimates word probabilities by computing appropriate frequencies from data. The significant advantage of FE is that it is simple to implement, often provides reasonable prediction performance, and is efficient. The equation below shows the Multinomial Naive Bayes (MNB) model:

$$P(c|d) = \frac{P(c)\prod P(w_i|c)^{f_i}}{P(d)}$$

where f_i is the number of occurrences of a word w_i in a document d, $P(w_i|c)$ is the conditional probability that a word w_i may happen in a document d given the class value c, and n is the number of

unique words in the document d. $P(c)$ is the prior probability that a document with class label c may happen in the document collections.

2.1.2 Linear Support Vector Machine (Linear SVM)

Support Vector Machines (SVMs) [5] could be the ideal model for this problem. Language classifying has a high dimensional input space where SVMs are effective. In this project a SVM with a linear kernel is used: $K(x_i, x_j) = x_i^T x_j$. Other common kernels are the polynomial, rbf and sigmoid.

In this project, the data includes multiple classes; thus a possible and reasonable approach is the *One-Against-All Approach*. Consider an M-class problem, where we have N training samples: $x_1, y_1, ..., x_N, y_N$. Here $x_i \in R^m$ is a m-dimensional feature vector and $y_i \in 1, 2, ..., M$ is the corresponding class label. One-against-all approach constructs M binary SVM classifiers, each of which separates one class from all the rest. The ith SVM is trained with all the training examples of the ith class with positive labels, and all the others with negative labels. Mathematically the ith SVM solves the following problem that yields the ith decision function :$f_i(x) = w^{T_i}\phi(x) + b_i$, minimises: $L(w, \xi_i^j) = \frac{1}{2}\|w_i\|^2 + C\sum_{l=1}^{N}\xi_i^j$, subject to: $\tilde{y_j}(w_i^T\phi(x_j) + b_i) \leq 1 - \xi_i^j, \xi_i^j \leq 0$, where $\tilde{y_j} = 1$ if $y_j = i$ and $\tilde{y_j} = -1$ otherwise. At the classification phase, a sample x is classified as in class i^* whose f_{i^*} produces the largest value $i^* = argmax_{i=1,...,M} f_i(x) = argmax_{i=1,...,M}(w^{T_i}\phi(x) + b_i)$

2.2 Deep Learning Models

2.2.1 Text Convolutional Neural Network (TextCNN)

Convolutional neural networks (CNNs) learn local features and assume that these features are not restricted by their absolute positions. In the field of NLP, they are applied in problems such as Part–Of–Speech Tagging (POS) and Named Entity Recognition (NER). The figure 1 represents a two layer CNN.

Neurons in CNN are locally connected with neurons in previous layer and the weights of the same filter are shared across the same layer. The mathematical details for the h nodes are given by the formula: $f(\sum_i(w_i x_i + b)$, where x_i are the inputs, their corresponding weights w_i, a bias b and the activation function f applied to the weighted sum of the inputs.

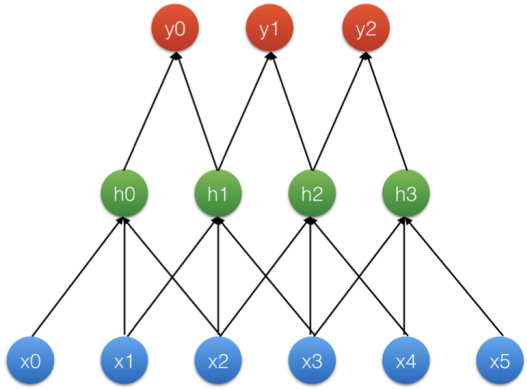


Figure 1: Two Layer CNN Architecture

2.2.2 BiDirectional Long Short-Term Memory (BiLSTM)

In 1997, Hochreiter and Schmidhuber proposed an LSTM [4] network in order to overcome the shortcomings of vanishing and exploding gradient in the RNN model. The key concept behind an LSTM model is to regulate the cell states by using three gates, namely, input, forget and output gates, as depicted in the below figure.

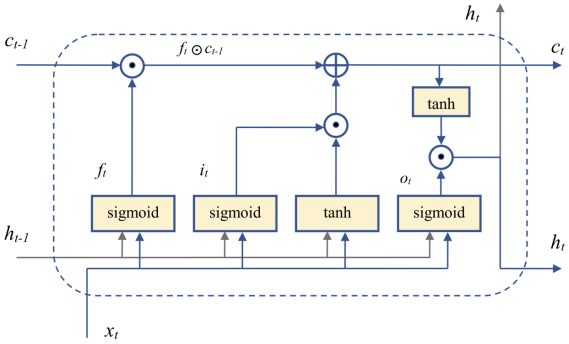


Figure 2: LSTM Architecture

The forget gate, f_t, determines whether to forget or keep the information of previous state, c_{t-1} by looking at the values of input, x_t, and hidden state, h_{t-1}, and its output value maybe a 0 or 1. Similarly, the input gate, i_t decides how much information of the input text, x_t and h_{t-1} should pass in order to update the cell state, and its output maybe a 0 or 1. The value of c_t represents the generated cell state as a result of mathematical operations on c_{t-1}, f_t and i_t . The output gate, o_t, controls the flow of information from the current cell state to the hidden state, and its value maybe a 0 or 1. The mathematical details for these gates are given in

the below equations:

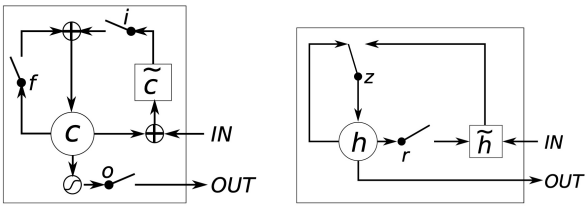
$$f_t = sigmoid(W_{fx}x_t + W_{fh}h_{t-1} + b_f)$$
$$i_t = sigmoid(W_{ix}x_t + W_{ih}h_{t-1} + b_i)$$
$$c_t = c_{t-1} \odot f_t + i_t \odot tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_c)$$
$$o_t = sigmoid(W_{ox}x_t + W_{oh}h_{t-1} + b_o)$$
$$h_t = o_t \odot tanh(c_t)$$

Where $x_t \in R_n$ is the input vector, $W \in R^{\ni * n}$, $b \in R^v$ and the superscripts n and v depict the dimension of the input vector and the number of words in the dataset or vocabulary, respectively. At any time, t, the inputs to LSTM are input vector x_t, previously hidden state h_{t-1}, and previous cell state c_{t-1}, whereas the outputs are current hidden state h_t and current cell state c_t.

In the LSTM network, information propagates entirely in a forward direction which indicates that the state of time t only depends on the information before t. However, to characterize the whole semantic of an input review, subsequent information is equally effective as the previous ones. Thus, for a better representation of contextual information, Bidirectional LSTM (BiLSTM) model was employed. The BiLSTM model is composed of two LSTM networks and is capable of reading input reviews in both directions, forward and backwards. The forward LSTM processes information from left to right and its hidden state can be shown as $\overrightarrow{h_t} = LSTM(x_t, \overrightarrow{h_{t-1}})$ whereas the backward LSTM processes information from right to left and its hidden state can be expressed as $\overleftarrow{h_t} = LSTM(x_t, \overleftarrow{h_{t+1}})$. Finally, the output of BiLSTM can be summarized by concatenating the forward and backward states as $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$.

2.2.3 Gated Recurrent Unit (GRU)

Gated recurrent units (GRUs) [3] [10] are a gating mechanism in recurrent neural networks, introduced in 2014 by Kyunghyun Cho et al. The GRU is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate. GRU's performance on certain tasks of music modelling, speech signal modelling and natural language processing was found to be similar to that of LSTM. GRU has been shown to exhibit better performance on certain smaller and less frequent datasets. A comparison between the LSTM and GRU architectures is illustrated in the below figure.



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Figure 3: Illustration of (a) LSTM and (b) Gated Recurrent Unit. In LSTM, i,f and o are the input, forget and output gates, respectively. c and $\tilde{c}$ denote the memory cell and the new memory cell content. In GRU, r and z are the reset and update gates, and h and $\tilde{h}$ are the activation and the candidate activation.

3 Data

The data came from Kaggle, which can be found on this web page. This dataset came in a CSV file with texts and a label with the corresponding language. Overall, there were 10267 unique texts covering 17 different languages, mainly European, Indic and Middle East languages. The languages are English, Malayalam, Hindi, Tamil, Kannada, French, Spanish, Portuguese, Italian, Russian, Swedish, Dutch, Arabic, Turkish, German, Danish and Greek.

3.1 Raw Data

The languages that consist of the data are illustrated in the below graph. English is the language with the most observations, while Hindi is the least observed, and the rest of the languages have between 400 and 800 observations.

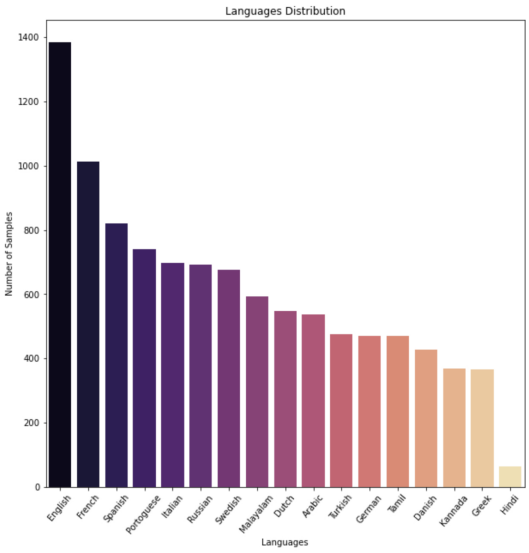


Figure 4: Distribution of the Unprocessed Data

The below table represents a few examples of how the unprocessed texts look. In the texts, punctuation marks such as commas, quotation marks and brackets, numbers, stop words and words coming from the same lemma could be spotted. In the next section, the cleaning process of the data will be explained, which is important for the classification part.

	Text	Language
0	Nature, in the broadest sense, is the natural...	English
1	"Nature" can refer to the phenomena of the phy...	English
2	The study of nature is a large, if not the onl...	English
3	Although humans are part of nature, human acti...	English
4	[1] The word nature is borrowed from the Old F...	English
...	...	...
10332	ನಿಮ್ಮ ತಪ್ಪು ಏನು ಬಂದಿದೆಯೆಂದರೆ ಆ ದಿನದಿಂದ ನಿಮಗೆ ಒ...	Kannada
10333	ನಾರ್ಸಿಸಾ ತಾನು ಮೊದಲಿಗೆ ಹೆಣಗಾಡುತ್ತಿದ್ದ ಮಾರ್ಗಗಳನ್...	Kannada
10334	ಹೇಗೆ ' ನಾರ್ಸಿಸಿಸಮ್ ಈಗ ಮರಿಯನ್ ಅವರಿಗೆ ಸಂಭವಿಸಿದ ಎ...	Kannada
10335	ಅವಳು ಈಗ ಹೆಚ್ಚು ಚಿನ್ನದ ಬ್ರೆಡ್ ಬಯಸುವುದಿಲ್ಲ ಎಂದು ...	Kannada
10336	ಟೆರ್ರಿ ನೀವು ನಿಜವಾಗಿಯೂ ಆ ದೇವದೂತನಂತೆ ಸ್ವಲ್ಪ ಕಾಣು...	Kannada

Figure 5: Table of the Unprocessed Data

3.2 Data Pre-processing and Preparation

To work with texts, it is essential to transform them into a consistent format using pre-processing techniques to remove the undesirable noisy text from the data. Most of these techniques are offered by spaCy [12], which is an open-source software library for advanced natural language processing. More specifically, removing the stop words and punctuation marks and lemmatising the texts is one of the essential parts of text data preparation. Then it is important to tokenise the text into word tokens to identify the patterns and boundaries between every word in the text corpora. Therefore, all the pre-process actions will help to normalise the data into a more efficient format for sentiment analysis.

In this project, 65% of the data is used for training the models and 35% of the data for testing. The figure 7 illustrates the training data distribution.

It is evident that the training dataset is unbalanced as some class labels have a very high number of observations compared to others that have a very low number of observations. To overcome this obstacle, oversampling [11] is an ideal method, which is a technique used to adjust the class distribution of a data set. In this project, random oversampling is performed, which involves supplementing the training data with multiple copies of some of the minority classes. A benefit of random oversampling is that instead of duplicating every sample in the

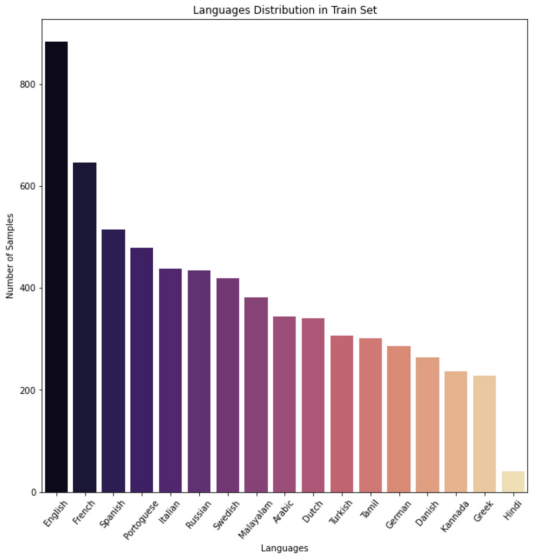


Figure 6: Distribution of the Training Data

minority class, some of them might be randomly chosen with replacement. In the below figure, the balanced training data is represented.

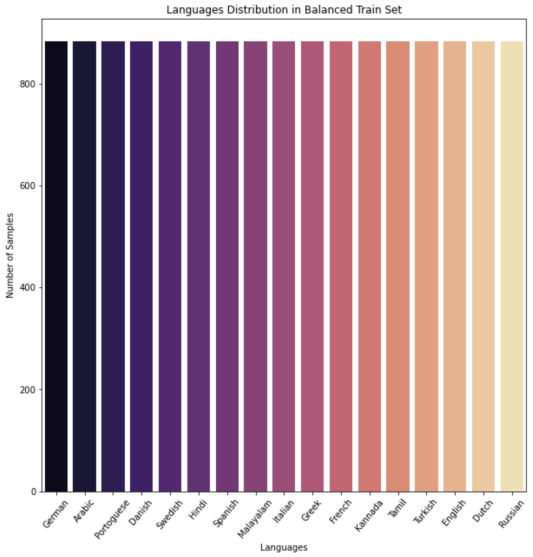


Figure 7: Distribution of the Balanced Training Data

4 Method

4.1 Bag-of-Words (CountVectorizer)

The bag-of-words [7] model simplifies the representation used in natural language processing and information retrieval. In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. The bag-of-words model is commonly used in methods of document classification where the (frequency of)

occurrence of each word is used as a feature for training a classifier.

4.2 Term Frequency-Inverse Document Frequency (TfidfVectorizer)

In information retrieval, the term frequency-inverse document frequency (tf–idf) [13] is a numerical statistic intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches for information retrieval, text mining, and user modelling. The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

4.3 Training the Models

In the traditional machine learning methods, the classifiers are built using built-in classes in scikit-learn [6]. Scikit-learn provided a simple way to interact with CountVectorizer, which converts a collection of text documents to a matrix of token counts, and TfidfVectorizer, which converts a collection of raw documents to a matrix of TF-IDF features. Then pipelines are created combining CountVectorizer or TfidfVectorizer and classifiers. Pipelines sequentially apply a list of transforms and a final estimator, which is a convenient method for parameter tuning in the following steps. Using GridSearch class in scikit-learn, which is an exhaustive search over specified parameter values for an estimator, was used to maximise the use of the classifiers.

In the deep learning methods, the tokeniser from keras [2] was used, which allows vectorising a text corpus by turning each text into either a sequence of integers. Then the sequences are padded, which transforms a list of sequences into a 2D Numpy array. The problem is a categorical classification with 17 targets since there are 17 languages to identify. Thus, the labels for the train and test sets need to be one-hot encoded. Finally, the deep learning models are created and trained using the appropriate layers imported from keras.

4.4 Evaluation Metrics

The evaluation metrics are used to measure the quality of predictions from the classification algorithms. The main classification metrics that the models were evaluated are f1-score on a per-class basis, accuracy and confusion matrix.

5 Results

5.1 Traditional Machine Learning Methods

5.1.1 Multinomial Naive Bayes Classifier

Multinomial Naive Bayes with CountVectorizer and TfidfVectorizer are the first experiments on language identification. More specifically, a 5-fold cross-validation for each model in parameter tuning was implemented. The best parameters for the MNB with CountVectorizer were (1,2) for ngram range, binary set to True and 0.1 for the additive (Laplace/Lidstone) smoothing parameter. The same parameters were used for the MNB with TfidfVectorizer. The evaluation metrics were similar for both models; they achieved a total accuracy of 94%. However, in both experiments, the f1-score for English was 75%, while for the rest of the languages was between 88% and 99%. Thus, there was not any significant performance difference between the MNB models. The tables with the evaluation metrics and confusion matrices are illustrated below.

	precision	recall	f1-score	support
English	0.60	1.00	0.75	185
Portoguese	0.89	0.90	0.90	141
Dutch	0.94	0.91	0.93	183
Arabic	0.96	0.96	0.96	476
Turkish	0.99	0.95	0.97	347
French	0.98	0.84	0.91	154
Swedish	1.00	0.93	0.96	122
Italian	1.00	0.95	0.98	22
Malayalam	0.96	0.90	0.93	235
Kannada	1.00	0.95	0.97	128
Greek	1.00	0.99	0.99	206
Russian	0.97	0.99	0.98	258
Tamil	1.00	0.94	0.97	234
Spanish	0.96	0.91	0.94	277
German	0.99	0.94	0.97	226
Danish	1.00	0.96	0.98	162
Hindi	1.00	0.97	0.98	165
accuracy			0.94	3521
macro avg	0.96	0.94	0.94	3521
weighted avg	0.96	0.94	0.95	3521

Figure 8: Evaluation metrics of the Multinomial Naive Bayes classifier with CountVectorizer of the test data.

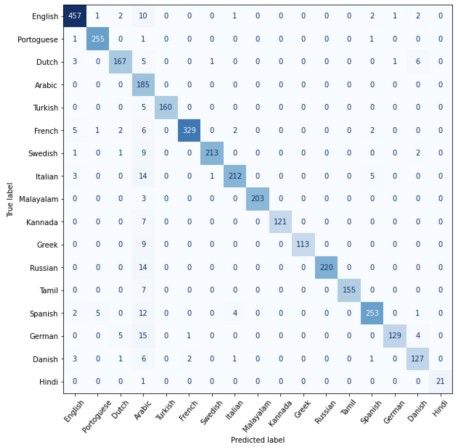


Figure 9: Confusion matrix of the Multinomial Naive Bayes classifier with CountVectorizer of the test data.

	precision	recall	f1-score	support
English	0.60	1.00	0.75	185
Portoguese	0.89	0.88	0.89	141
Dutch	0.94	0.90	0.92	183
Arabic	0.96	0.96	0.96	476
Turkish	0.99	0.95	0.97	347
French	0.98	0.85	0.91	154
Swedish	1.00	0.93	0.96	122
Italian	1.00	0.95	0.98	22
Malayalam	0.97	0.90	0.93	235
Kannada	1.00	0.95	0.97	128
Greek	1.00	0.99	0.99	206
Russian	0.97	0.98	0.98	258
Tamil	1.00	0.94	0.97	234
Spanish	0.96	0.92	0.94	277
German	0.98	0.94	0.96	226
Danish	1.00	0.96	0.98	162
Hindi	1.00	0.97	0.98	165
accuracy			0.94	3521
macro avg	0.96	0.94	0.94	3521
weighted avg	0.96	0.94	0.95	3521

Figure 10: Evaluation metrics of the Multinomial Naive Bayes classifier with TfidfVectorizer of the test data.

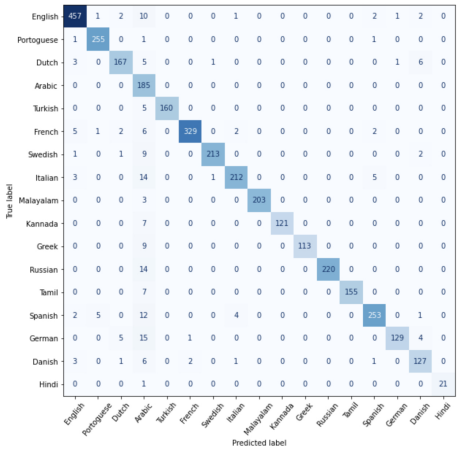


Figure 11: Confusion matrix of the Multinomial Naive Bayes classifier with CountVectorizer of the test data.

5.1.2 Linear Support Vector Machine (Linear SVM)

The Linear SVM models provided satisfactory results. In this case, the model with the TfidfVectorizer performed better than the model with CountVectorizer. More specifically, the best parameters for the Linear SVM model with CountVectorizer were (1,1) for ngram range, binary set to False and the regularisation parameter equals 1. Whereas the best parameters for the model with TfidfVectorizer were (1,2) for ngram range, binary set to True and regularisation parameter equals 5. From the below figures, it is evident that the TfidfVectorizer model performed better with 92% accuracy compared to the 87% accuracy of the CountVectorizer model. The most interesting result of these models was that the f1-score of Kannada was very low in both cases. From the confusion matrices, it is apparent that most mispredictions are in English. The tables with the evaluation metrics and confusion matrices are illustrated below.

	precision	recall	f1-score	support
English	1.00	0.90	0.95	185
Portoguese	0.81	0.77	0.79	141
Dutch	0.95	0.83	0.88	183
Arabic	0.92	0.87	0.90	476
Turkish	0.97	0.88	0.92	347
French	0.84	0.77	0.80	154
Swedish	1.00	0.83	0.91	122
Italian	1.00	0.82	0.90	22
Malayalam	0.97	0.85	0.91	235
Kannada	0.29	1.00	0.45	128
Greek	1.00	0.97	0.99	206
Russian	0.98	0.95	0.97	258
Tamil	1.00	0.87	0.93	234
Spanish	0.91	0.84	0.88	277
German	0.98	0.82	0.89	226
Danish	1.00	0.90	0.95	162
Hindi	0.99	0.89	0.94	165
accuracy			0.87	3521
macro avg	0.92	0.87	0.88	3521
weighted avg	0.93	0.87	0.89	3521

Figure 12: Evaluation metrics of the Linear Support Vector Machine classifier with CountVectorizer of the test data.

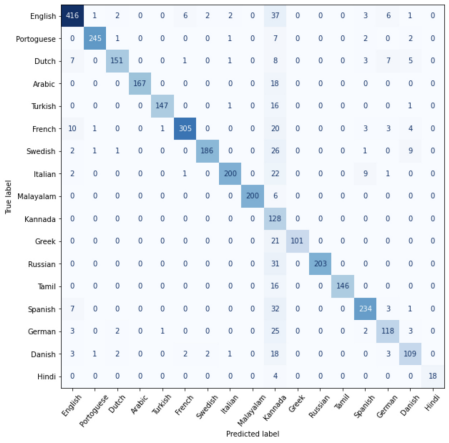


Figure 13: Confusion matrix of the Linear Support Vector Machine classifier with CountVectorizer of the test data.

	precision	recall	f1-score	support
English	1.00	0.94	0.97	185
Portoguese	0.85	0.83	0.84	141
Dutch	0.96	0.86	0.90	183
Arabic	0.94	0.94	0.94	476
Turkish	0.98	0.95	0.96	347
French	0.89	0.82	0.86	154
Swedish	1.00	0.86	0.93	122
Italian	1.00	0.95	0.98	22
Malayalam	0.97	0.89	0.93	235
Kannada	0.41	1.00	0.58	128
Greek	1.00	0.99	0.99	206
Russian	1.00	0.97	0.98	258
Tamil	1.00	0.92	0.96	234
Spanish	0.96	0.88	0.92	277
German	0.98	0.90	0.94	226
Danish	1.00	0.94	0.97	162
Hindi	1.00	0.96	0.98	165
accuracy			0.92	3521
macro avg	0.94	0.92	0.92	3521
weighted avg	0.95	0.92	0.93	3521

Figure 14: Evaluation metrics of the Linear Support Vector Machine classifier with TfidfVectorizer of the test data.

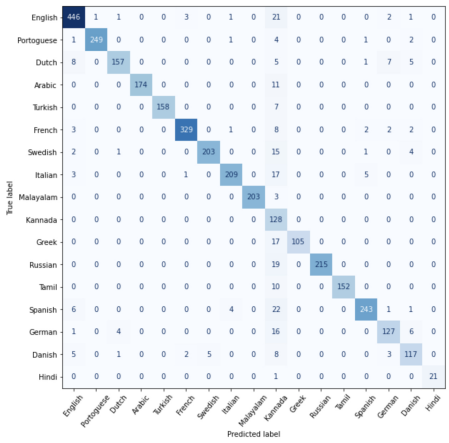


Figure 15: Confusion matrix of the Linear Support Vector Machine classifier with TfidfVectorizer of the test data.

5.2 Deep Learning Models

5.2.1 Text Convolutional Neural Network (TextCNN)

A simple Convolutional Neural Network was attempted in the first deep learning experiment. The model contains embedding layer, two convolutional layers with a relu activation function, one dropout layer with a 0.2 rate, a global average pooling layer and a dense layer with a softmax activation function. The summary of the model is shown in the below figure.

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, 200, 128)	3365888
conv1d_4 (Conv1D)	(None, 191, 128)	163968
dropout_1 (Dropout)	(None, 191, 128)	0
conv1d_5 (Conv1D)	(None, 182, 256)	327936
global_average_pooling1d_2 (GlobalAveragePooling1D)	(None, 256)	0
dense_2 (Dense)	(None, 17)	4369

Total params: 3,862,161
Trainable params: 3,862,161
Non-trainable params: 0

Figure 16: Summary of the TextCNN model.

The model was trained with a 128 batch size and 20 epochs. The graphs below represent the progress of the accuracy and the loss of the train and test data.

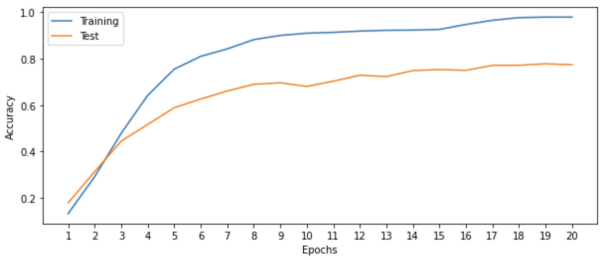


Figure 17: Train and Test accuracy.

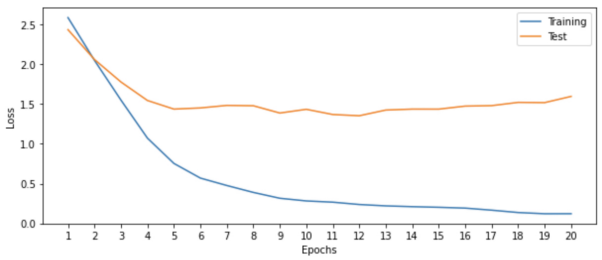


Figure 18: Train and Test loss.

From the above plots, it is evident that until the 17th epoch the test accuracy increases and after,

it remains plateau. However, the test loss after the 13th epoch keeps increasing while the training loss keeps decreasing, which means that the model overfits after the 13th epoch.

The below figures show the accuracy and f1-score on a per-class basis and the confusion matrix of the test data. The accuracy is 77% and some f1-scores and relatively low. The f1-score for Swedish is 48%, and it was mainly mispredicted with Tamil.

	precision	recall	f1-score	support
English	1.00	0.78	0.88	185
Portoguese	0.73	0.63	0.68	141
Dutch	0.70	0.75	0.73	183
Arabic	0.84	0.76	0.80	476
Turkish	0.71	0.82	0.76	347
French	0.57	0.66	0.61	154
Swedish	0.32	0.98	0.48	122
Italian	0.89	0.77	0.83	22
Malayalam	0.88	0.66	0.76	235
Kannada	0.89	0.79	0.84	128
Greek	0.98	0.77	0.86	206
Russian	0.99	0.88	0.93	258
Tamil	0.78	0.79	0.79	234
Spanish	0.93	0.71	0.80	277
German	0.76	0.78	0.77	226
Danish	0.90	0.80	0.85	162
Hindi	0.90	0.80	0.85	165
accuracy			0.77	3521
macro avg	0.81	0.77	0.78	3521
weighted avg	0.82	0.77	0.79	3521

Figure 19: Evaluation metrics of the TextCNN of the test data.

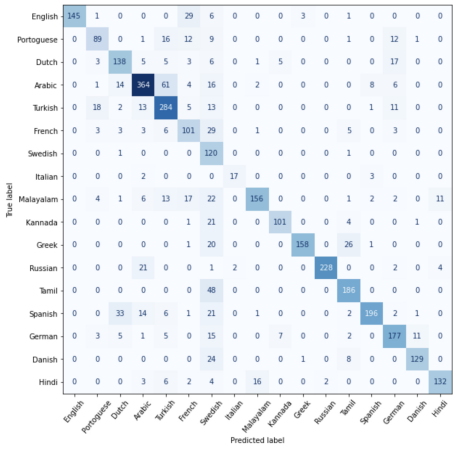


Figure 20: Confusion matrix of the TextCNN of the test data.

5.2.2 BiDirectional Long Short-Term Memory (BiLSTM)

The second deep learning is more ”advanced” compared to the TextCNN. The BiLSTM model structure is one embedding layer, one bidirectional LSTM layer and one dense layer with a softmax activation function. Below is the summary of the model.

Layer (type)	Output Shape	Param #
embedding_6 (Embedding)	(None, 200, 128)	3365888
bidirectional (Bidirectiona l)	(None, 64)	41216
dense_6 (Dense)	(None, 17)	1105

Total params: 3,408,209
Trainable params: 3,408,209
Non-trainable params: 0

Figure 21: Summary of the BiLSTM model.

The model was trained with a 128 batch size and 7 epochs. The graphs below represent the progress of the accuracy and the loss of the train and test data.

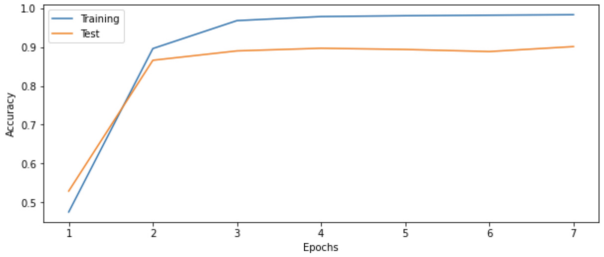


Figure 22: Train and test accuracy of the BiLSTM model.

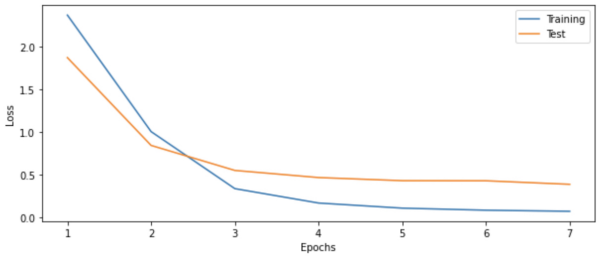


Figure 23: Train and test loss of the BiLSTM model.

From the line graphs, it is apparent that test accuracy keeps increasing until the 3rd epoch and after it remains steady, while the test loss keeps dropping until the 5th epoch. The 4th epoch is ideal because it provides high accuracy and a low loss.

The test data’s accuracy is higher than the previous model, 90%, while only the French language has a low f1-score, 0.55. More specifically, multiple French texts are mispredicted in all the other languages except Italian.

	precision	recall	f1-score	support
English	0.99	0.96	0.98	185
Portoguese	0.84	0.79	0.81	141
Dutch	0.94	0.83	0.88	183
Arabic	0.96	0.91	0.94	476
Turkish	0.98	0.93	0.95	347
French	0.39	0.93	0.55	154
Swedish	0.93	0.89	0.91	122
Italian	1.00	0.95	0.98	22
Malayalam	0.94	0.86	0.90	235
Kannada	0.97	0.88	0.92	128
Greek	0.95	0.92	0.94	206
Russian	1.00	0.97	0.99	258
Tamil	0.98	0.90	0.94	234
Spanish	0.94	0.91	0.92	277
German	0.94	0.88	0.91	226
Danish	1.00	0.83	0.91	162
Hindi	0.99	0.94	0.96	165
accuracy			0.90	3521
macro avg	0.93	0.90	0.90	3521
weighted avg	0.94	0.90	0.91	3521

Figure 24: Evaluation metrics of the BiLSTM of the test data.

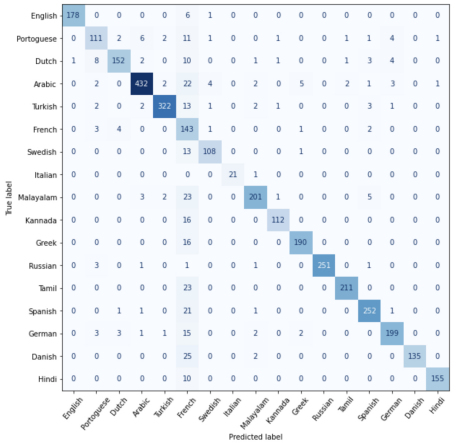


Figure 25: Confusion matrix of the BiLSTM of the test data.

5.2.3 Gated Recurrent Unit (GRU)

The last deep learning model is the GRU model. The model is consisted of one embedding layer, one GRU layer and one dense layer with a softmax activation function. The summary of the model is illustrated below.

Layer (type)	Output Shape	Param #
embedding_5 (Embedding)	(None, 200, 128)	3365888
gru_2 (GRU)	(None, 64)	37248
dense_5 (Dense)	(None, 17)	1105

Total params: 3,404,241
Trainable params: 3,404,241
Non-trainable params: 0

Figure 26: Summary of the GRU model.

The model was trained with a 128 batch size and 7 epochs. The graphs below represent the progress of the accuracy and the loss of the train and test data.

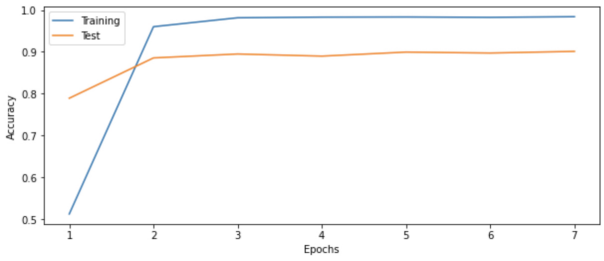


Figure 27: Train and Test accuracy.

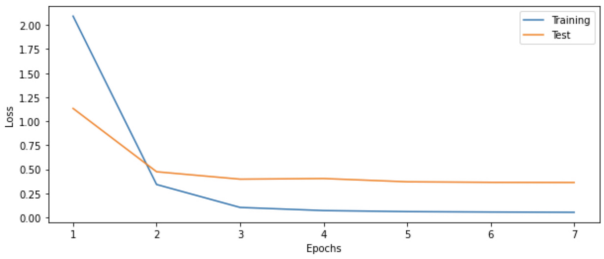


Figure 28: Train and Test loss.

From the above plots, it is clear that after the 2nd epoch, the train and test accuracies and losses remain plateau.

The accuracy of the test data is and the results of the f1 scores are very similar to the ones from the BiLSTM model. More specifically the test accuracy is 90%, and most of the f1-scores are higher than 0.9; only the f1-score of French is low, 0.55. The tables with the evaluation metrics and confusion matrices are illustrated below.

	precision	recall	f1-score	support
English	1.00	0.95	0.97	185
Portoguese	0.83	0.79	0.81	141
Dutch	0.94	0.84	0.89	183
Arabic	0.93	0.93	0.93	476
Turkish	0.98	0.93	0.96	347
French	0.39	0.91	0.55	154
Swedish	0.99	0.87	0.93	122
Italian	1.00	1.00	1.00	22
Malayalam	0.96	0.84	0.90	235
Kannada	0.96	0.87	0.91	128
Greek	0.96	0.92	0.94	206
Russian	1.00	0.97	0.98	258
Tamil	1.00	0.88	0.94	234
Spanish	0.93	0.89	0.91	277
German	0.93	0.88	0.91	226
Danish	0.99	0.84	0.91	162
Hindi	0.98	0.96	0.97	165
accuracy			0.90	3521
macro avg	0.93	0.90	0.91	3521
weighted avg	0.93	0.90	0.91	3521

Figure 29: Evaluation metrics of the GRU of the test data.

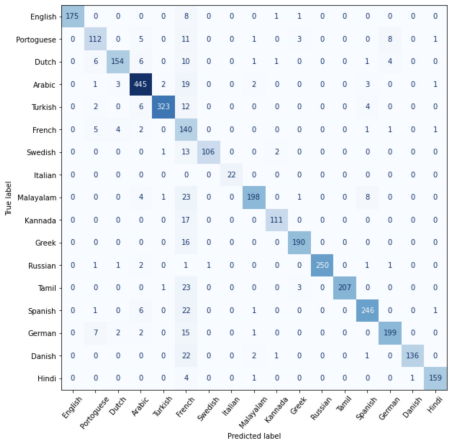


Figure 30: Confusion matrix of the GRU of the test data.

6 Discussion

First, a reference would be made to the result of traditional machine learning methods. The linear SVM and the MNB had the best accuracies among all classifiers, where both algorithms have their natural advantages. The MNB algorithm is a probabilistic learning method primarily used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem and predicts the tag of a text. It computes the probability of each tag for a given sample and then gives the tag with the highest probability as output. The SVMs are well known for their effectiveness in high-dimensional spaces. The MNB with CountVectrorizer or TfidfVectorizer provided the same results, so there is no clear preference between the models. An interesting result is that the f1-scores for English are relatively low, because of poor precisions, compared to other languages. Referring to the previous comment, it could be seen in the confusion matrix that English is only mispredicted with languages that share the same characters as English. The Linear SVM performed better with TdifVectorizer than with CountVectorizer. In the evaluation metrics of the Linear SVM with TfidfVectorizer, only the f1-score in Kannada is very low; multiple texts were mispredicted with all of the languages. The only mispredictions that can be explained are the ones with Arabic, Turkish, Greek and the rest of the Indic languages, which are languages that have special characters.

The second part will focus on deep learning methods. In general, the three classifiers could have performed better on the dataset. The BiLSTM and GRU models were the two classifiers that gave relatively high accuracies, although someone could expect higher numbers because of their complex structure and training time. The TextCNN gave a 77% accuracy on the test data and multiple f1-scores around 60% and 70%. Moreover, in the confusion matrix, the most mispredictions are in Swedish. The BilSTM model had a 90% accuracy on the test data, and multiple f1-scores were around 90%. The only low f1-score it could be spotted for the French language. Watching the mispredictions in the confusion matrix, the ones that make sense are those whose language is based in Latin, such as Italian, Portuguese, and Spanish. The last deep learning model is the GRU, which performed similarly to the BiLSTM, with a 90% accuracy and a low f1-score for French.

7 Conclusion

In order to work with text data, it is essential to transform the raw text into a form that can be understood and used by machine learning algorithms; this is called text pre-processing. The text pre-processing for the project's data had its complications and constraints. For example, the data contains multiple languages, which might only be known to some. Hence, people who speak and understand these languages could offer their assistance in this project. Another tribulation was that libraries for some languages took much time to be found and evaluated. For the evaluation part, there is still some doubt if these libraries were practical. For instance, more certainty is needed if the libraries referring to Indic languages contain all the stopwords. In the future, it would be interesting to include more languages, such as languages from Asian countries, which are missing from this dataset.

This project's traditional machine learning models outperformed deep learning techniques in language classification. LinearSVM and MNB might be the best candidates for text classification due to their advantages. Furthermore, other traditional machine learning could be considered and applied to this project, such as Decision Tree and Random Forest. Deep learning methods still have their potential, but obtaining a better performance might require more effort for their layers construction compared to traditional methods, with more time and computational resources required. In the future, it could be tried to implement more advanced methods, such as BERT [8], which is transformer-based

machine learning technique for natural language processing pre-training developed by Google. Another interesting model is Fasttext[9], a library created by Facebook's AI Research lab for learning word embeddings and text classification.

8 Code

The Github repository for this project could be found here.

References

[1] Muhammad Abbas, K Ali Memon, A Aleem Jamali, Saleemullah Memon, and Anees Ahmed. Multinomial naive bayes classification model for sentiment analysis. *IJCSNS Int. J. Comput. Sci. Netw. Secur*, 19(3):62, 2019.

[2] François Chollet et al. Keras. https://keras.io, 2015.

[3] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[4] Zabit Hameed and Begonya Garcia-Zapirain. Sentiment classification using a single-layered bilstm model. *IEEE Access*, 8:73992–74001, 2020.

[5] Yi Liu and Yuan F. Zheng. One-against-all multi-class svm classification using reliability measures. volume 2, pages 849 – 854 vol. 2, 01 2005.

[6] 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.

[7] Wikipedia contributors. Bag-of-words model — Wikipedia, the free encyclopedia, 2022. [Online; accessed 29-December-2022].

[8] Wikipedia contributors. Bert (language model) — Wikipedia, the free encyclopedia, 2022. [Online; accessed 2-January-2023].

[9] Wikipedia contributors. Fasttext — Wikipedia, the free encyclopedia, 2022. [Online; accessed 2-January-2023].

[10] Wikipedia contributors. Gated recurrent unit — Wikipedia, the free encyclopedia, 2022. [Online; accessed 28-December-2022].

[11] Wikipedia contributors. Oversampling and undersampling in data analysis — Wikipedia, the free encyclopedia, 2022. [Online; accessed 2-January-2023].

[12] Wikipedia contributors. Spacy — Wikipedia, the free encyclopedia, 2022. [Online; accessed 6-January-2023].

[13] Wikipedia contributors. Tf–idf — Wikipedia, the free encyclopedia, 2022. [Online; accessed 29-December-2022].