NLP Model\Task Evaluation: Accuracy of different models on Language Detection/Identification

By Ayush Shenvi Pissurlenkar & Yousuf Khan

Introduction

Through natural language processing, methods exist to classify texts, be it by sentiment analysis, language detection, or document processing. For our final project, we implemented and analyzed different natural language models to classify texts into the respective language in which they were written. The ability to classify languages has many uses throughout the world, especially within science, history and language classes themselves. For example, groups of scholars are trying to use it to record all of the world's languages to compile written records of our world history. In contrast, others have used it to demonstrate the differences between cultural and cross-cultural patterns that give regions their native language/dialect [1]. While the concept seems simple, language classification has essential uses that make it necessary to consider when discussing the world of natural language processing. Through our project, we analyzed the performance of three models: Multinomial Naive Bayes (MNB), Support Vector Machines (SVM) and Recurrent Neural Networks (RNN) and their language classification ability with the Europarl language dataset.

Related Work

Given that language classification has been a portion of the natural language field for some time now, it has many previous works and methods. While the methods we selected are famous, others such as n-gram, graph-based-n-gram, majority voting, and prediction partial matching are all techniques that have achieved reasonably high accuracy results across different studies. Furthermore, researchers have spent efforts applying these models to many problems, from supporting low-resource languages within the field to building agnostic engines[2][3]. While many of these solutions achieve relatively high accuracy scores in the lower 90s, their approach requires the generated n-grams out of tokenized words. However, this approach does fail when it comes to non-boundary word languages such as those from Eastern Asia, i.e. Japanese. Solutions to this problem came from byte-level n-grams of entire strings instead of word-level n-grams[4]. Using the information from related work, we selected our models accordingly. We chose to use both n-gram methods at the word level and the overall string level.

Dataset

Before we discuss the models, it is essential to understand the data we were using as it directly impacts the ability of our models and research. The European Parliament Proceedings Parallel Corpus, or Europarl for short, is a dataset containing 21 languages and roughly 60 million words per language [5]. The dataset is comprised by extracting the proceedings from the European Parliament in the 21 European languages: Romanic (French, Italian, Spanish, Portuguese, Romanian), Germanic (English, Dutch, German, Danish, Swedish), Slavik (Bulgarian, Czech, Polish, Slovak, Slovene), Finno-Ugric (Finnish, Hungarian, Estonian), Baltic (Latvian, Lithuanian), and Greek. While individual parallel corpus between specific languages exists within Europarl, we did not use them but instead chose to use the source release containing all 21 languages.

Thus allowing us to test the models across a broader range of languages instead of creating a more straightforward binary or tertiary analysis. The full breakdown of the languages with their respective sentences and words can be seen in Figure 1.

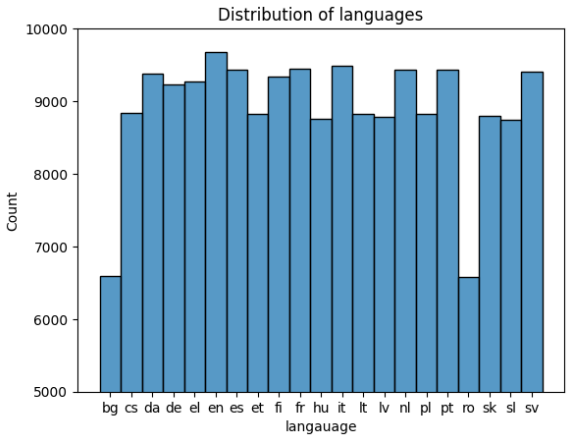


Fig. 1: Distribution of Languages

Data Preprocessing

With an understanding of our data, we needed to process the overall dataset into something accessible to our models. The final desired dataset was a pandas dataframe containing the processed text in one column and the 2-letter language identifier in the next column. We first started tackling the text containing XML portions to describe speakers, locations, and topics, all of which needed to be removed as it was irrelevant to our analysis. Given that all XML lines begin with "<", we could specify that character and remove those portions from our data. Next, we compiled all the sentences in every given text into a single string and added that as one entry to our data frame with its respective language identifier. We then drop any empty rows where the respective text contains no usable string data. This was done to create a test dataset composed of 4200 string-language pairs and a complete dataset of 187071 string-language pairs. An image of the pandas data frame can be seen in Figure 2.

	langauage	text
0	bg	Състав на Парламента: вж. протоколи
1	bg	Одобряване на протокола от предишното заседани...
2	bg	Състав на Парламента: вж. протоколи
3	bg	Проверка на пълномощията: вж. протоколи
4	bg	Внасяне на документи: вж. протоколи
...	...	...
187066	sv	Återupptagande av sessionen Jag förklarar Euro...
187067	sv	Återupptagande av sammanträdet Av tekniska skä...
187068	sv	Meddelande från talmannen Jag kan meddela att ...
187069	sv	Justering av protokollet från föregående samma...
187070	sv	Justering av protokollet från föregående samma...

187071 rows × 2 columns

Fig. 2: Dataset Dataframe

For MNB and SVM, the data was split into a standard 80/20, where 80% of the data is used for training and 20% for testing. In addition, we tokenized all the data before feeding it into our models, as that will allow them to be more effective in training and predicting. Finally, the test dataset was used to validate the MNB and SVM models, while their final models were trained on the full dataset.

For RNN, due to hardware constraints, it was validated and trained on the test dataset. Therefore, the data from the data frame was first tokenized, limiting the tokens using the 1370000 most essential words, thereby allowing us to include all of them. This number was chosen by getting the number of unique words within the training dataset, 1360372. Once tokenized, the sequences were then padded to ensure that each string-language pair was at max 50000 characters long, as we needed to standardize this before feeding into the neural network. With this, the data was again divided using the aforementioned 80/20 split.

Model 1: Multinomial Naive Bayes

It is a model for assigning text to labels based mainly on a statistical dataset analysis. It assigns labels to each text file by determining the probability that this text belongs to the class.

In our project, we have given our MNB model a range of n-grams as inputs to see how it reacts to different input parameters and affects its accuracy scores.

Model 2: Support Vector Machine

The Support Vector Machines are supervised machine learning models that are optimal margin classifiers and use different kernels to find an optimal decision boundary between all the possible output labels to divide the data perfectly. It generally finds a hyperplane that maximizes the separation of data points to their output labels in an n-dimensional space. It uses the concept of support vectors which are the data points with the minimum distance to the hyperplane, to create an optimal boundary.

In this project, we aim to use the Radial Basis function kernel and the Polynomial kernel within our SVM model to train it. The Polynomial kernel represents the similarity of vectors in the dataset in a feature space over polynomials of the original dataset.
The RBF kernel adds a radial basis method to improve the transformation of the data over the complete dataset.

Model 3: Recurrent Neural Network

The recurrent neural network, or RNN, that we designed is a single hidden layer RNN with gated recurrent units. This was chosen as gated recurrent units are good at avoiding the issue of growing/vanishing gradients during training due to long input sequences. Furthermore, as we knew our chosen dataset contained thousands of long strings, we needed to ensure the RNN could process the information without crashing or being unable to compile.
Our RNN consists of an embedding layer whose size is determined by the total vocabulary and has a batch size of 128. First, the tokenized and padded data is passed into the embedding layer to turn it into dense vectors of a fixed size and mapped, respectively. The data is then passed from the embedding layer to the hidden layer, consisting of gated recurrent units (GRUs) of size 25. The GRU layer is similar to an LSTM layer, but instead, it also includes a forget gate in its logic, giving it better performance in applications such as language classification.
After the hidden layer, the data is passed to a Dense layer which acts as our output. The dense layer has a size of 21 because there are 21 languages to classify from our dataset, and it uses a softmax function since we have more than two output possibilities. Had this instead been used for machine translation between two languages, the dense layer would instead have a size of 2 and use a sigmoid function.
With the model layers defined, the entire model is then compiled. For this step, we specify our loss function as categorical cross entropy as this is the better choice for models that have a large number of possible outputs. Along with this, we use an adam optimizer, as this is standard, and accuracy is our metric, so we can compare it to our other two models. Again, due to hardware constraints, we had to train our RNN model on the training dataset instead of the total dataset.

Results & Analysis

MNB Data

For our MNB model we decided to first vectorize our inputs and then train the model using n-grams from 1-gram to 7-gram. As you can see from the results below in Figure 3, the accuracy scores kept increasing as we increased our n-gram number. From 1-gram accuracy of 91.76 to 7-gram accuracy of 98.46%. As we reached 7-gram the computational time taken to run the model was too expensive and the increase in accuracy was stagnating so we did not proceed forward. From the bigram classification report for MNB, which can be seen in Appendix B.1, we can see the model had a 97% f1-score accuracy. We can also see the bigram heatmap in Appendix A.1.

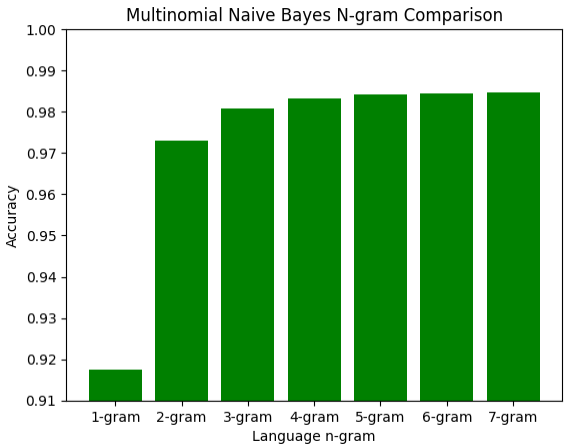


Fig. 3: MNB n-gram Comparison

SVM Data

For our SVM models we decided that we wish to hyper-parametrize the kernels and also test different n-grams to see how they would affect the accuracy metrics of the model. As you can see from Fig. 4 below we used the Polynomial and RBF kernel and trained our data over 1-gram to 3-gram. To improve the training performance we also used TFIDF-vectorization to vectorize our inputs and increase the efficiency of the model. As we reached 3-gram the computational time taken to run the model was too expensive and the increase in accuracy was stagnating, and in one case even dropping showing overfitting in the training set, so we did not proceed forward. See Appendix B.2 for the classification report of the RBF bigram model and Appendix A.2 for its heatmap.

RNN Data

The RNN model was trained for a total of 10 epochs which took roughly 30 hours to fully train. We first trained 2 epochs so that we could save a temporary model to use in validation, which resulted in a 70.55%. After another 8 epochs, for the final total of 10, the RNN model had an accuracy of 99.39%. The accuracy of the RNN over each epoch can be seen in Figure 5 below. As can be seen from the classification report and heatmap, the RNN performed exceptionally well in classifying all languages, with 12 of the 21 having full f1-scoring. The model struggled the most with classifying Polish as this was its lowest f1-score of 95%. See Appendix B.3 for the classification report of the model and Appendix A.3 for its heatmap.

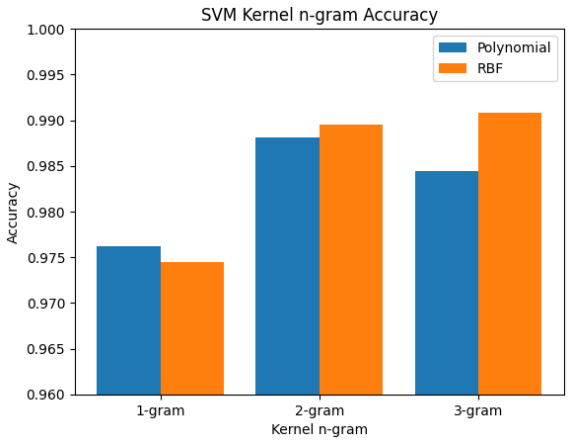


Fig. 4: SVM Kernel n-gram Comparison

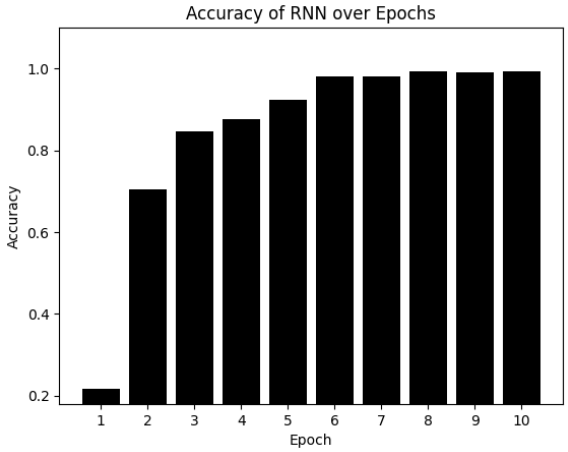


Fig 5: RNN Accuracy over epochs

Between our three models, the RNN performed the overall best, which can be seen in Appendix C.

The graph shows the three models f1-scores in a cluster bar graph comparison. This can also be seen in Figure 6 where the average f1-score for all the languages for each model is depicted. MNB had an average f1-score of 97.29%, SVM had 98.9%, and RNN had 99.14%.

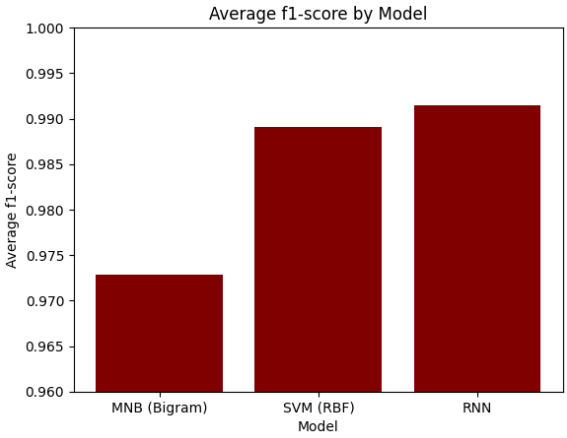


Fig 6: Average f1-score by model

For 12 out of 21 languages, the RNN model had a 1.0 f1-score, whereas SVM only had 2 languages with a 1.0, and MNB had none. The SVM RBF model was the most consistent of all three, with its score ranging from 0.98 to 1.0. The RNN and MNB models tied in terms of consistency, as the RNN scores range from 0.95 to 1.0 and MNB with its scores of 0.94 to 0.99. So both had a 0.5 score range overall.

From Figure 7, we can see that the language with the average worst score is Polish with 96.66%, with Latvian being second worst with 97%, and English third worst with 97.66%. The language with the best score is Bulgarian with 99.33%. Tied for second is six languages: Greek, Spanish, Finnish, French, Italian, and Portuguese, all with a score of 98.99%.

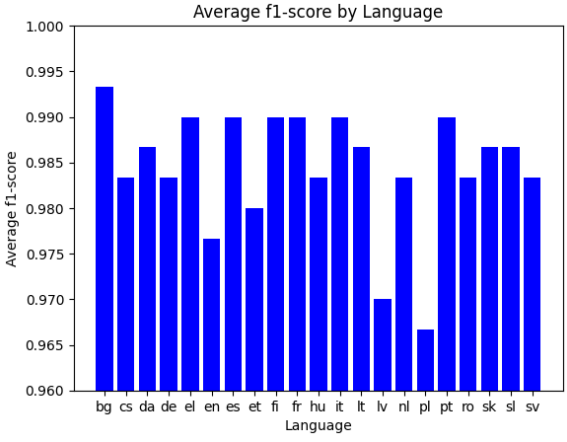


Fig 7: Average f1-score by Language

Conclusion

Overall, our data is consistent with what we had expected. Our initial hypothesis was that English was going to receive the lowest f1-score as it shares the most characters with all the other languages, so scoring in the bottom three was to our expectations. We were surprised to see the Polish score so low, 96.67%, as the language does have unique characteristics, such as having six more letters than English and also nine letters that contain a diacritical mark[6]. This was even more peculiar given that the other Slavic languages in the dataset, Slovenian and Slovak, both had a much higher average score of 98.66% and 98.33%, respectively.

In regards to our models, all of our expectations were met with each of the models' individual performances and how they stacked up to one another. We hypothesized that the RNN would have the highest overall accuracy, although we did not expect it to take so long to train and that greatly impacted our project plan towards the end as we were training and testing models. There was also one unique instance within the RNN training in epoch 8 where the model had a better accuracy than it would in the following epoch, which can again be seen in Figure 5. While we expected the SVM to be the most consistent model, we were still surprised by the fact that its accuracy only differed by 0.2 maximum between languages. With this consistency, it can be argued that SVM is actually the better model over RNN since RNN has a range of 0.5 between its language f1-scores.

Another expectation we had from our models was that if we kept increasing our n-gram sizes we would come upon overfitting, where the training accuracy of the model would be amazing and keep increasing but the test accuracy would slowly start to decrease. In our MNB models, we were not able to train high enough n-grams to see this as the 7-gram MNB

took over 3 hours and takes up about 20 GB of storage to save the model. But we can still see a trend of our accuracy score slowly stagnating which is what happens before it starts decreasing. While on the other hand in our SVM model we could start seeing the accuracy score decrease at 3-gram which clearly shows that we're overfitting the data. It also takes up a lot more computational time to run models with higher n-grams which is what we expected since there will be a lot more features to work with.

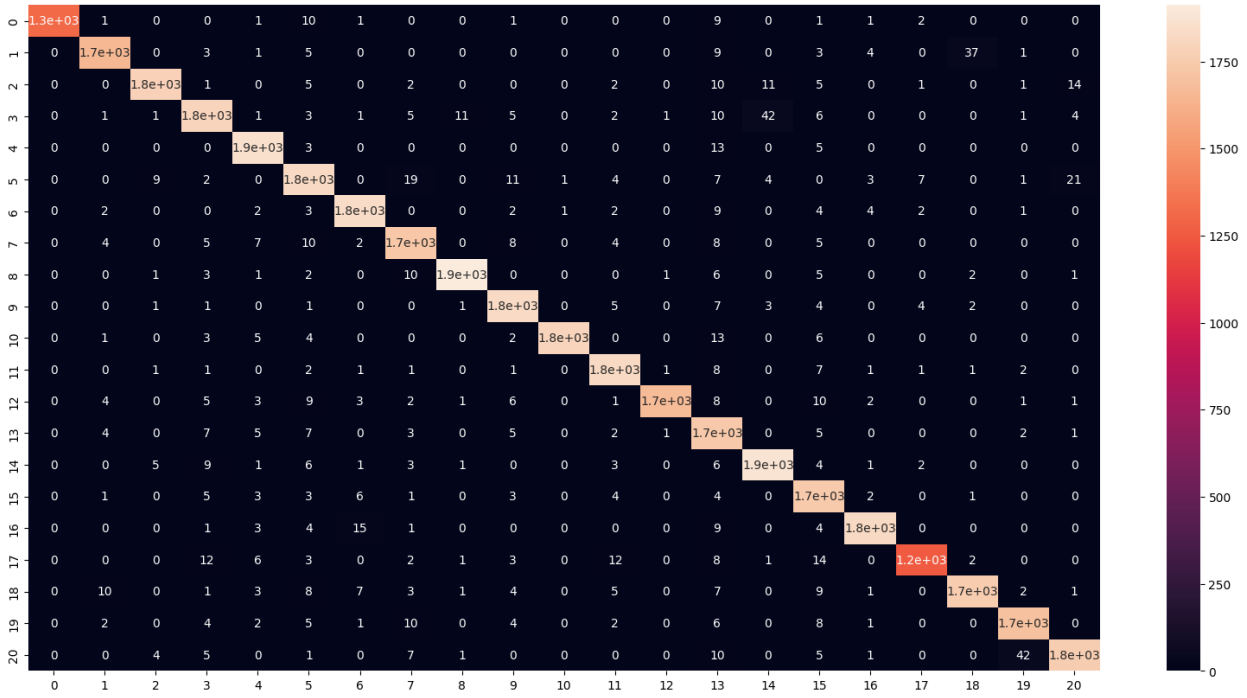
With the research we performed for our models, we determined potential future work within this space of natural language processing. To further validate our work, training the RNN model on the full dataset would be top priority as this would allow for a proper one to one comparison with the other models. Along with this, further tuning the RNN to determine its capabilities such as changing batch size, number of training epochs, and also the composition and number of layers within the model itself. The same goes for SVM and MNB, as future work could further test the boundaries of the models by using a larger number of n-grams in both cases and testing other kernels specifically for SVM. Beyond the models we used, we believe future work should also include other models such as logistic regression, convolutional neural networks, and even LSTM or long short-term memory neural networks. All of these models have different advantages and disadvantages to them, but we believe that although our models have achieved high 90% accuracy, there is still more to find as no model is yet perfect.

Work Cited

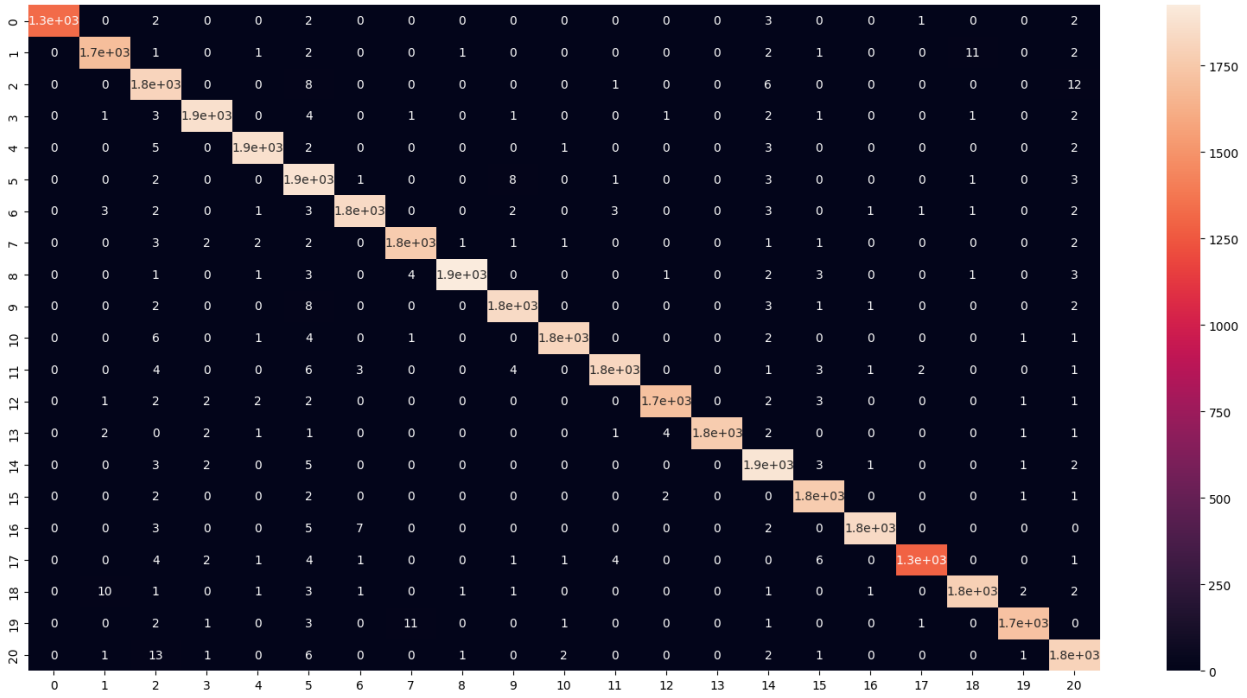
[1] http://anthropology.iresearchnet.com/classification-of-language/#:~:text=The%20first%20purpose%20is%20to,have%20previously%20been%20classified%20linguistically.
[2] https://aclanthology.org/W12-2108/
[3] https://www.win.tue.nl/~mpechen/publications/pubs/TrompPechenizkiy_LIGA_Benelearn11.pdf
[4] https://cs229.stanford.edu/proj2015/324_report.pdf
[5] https://www.statmt.org/europarl/
[6] https://www.careersinpoland.com/article/customs/did-you-know-that-7-essential-facts-about-the-polish-language

Appendix A - Heatmaps of Models

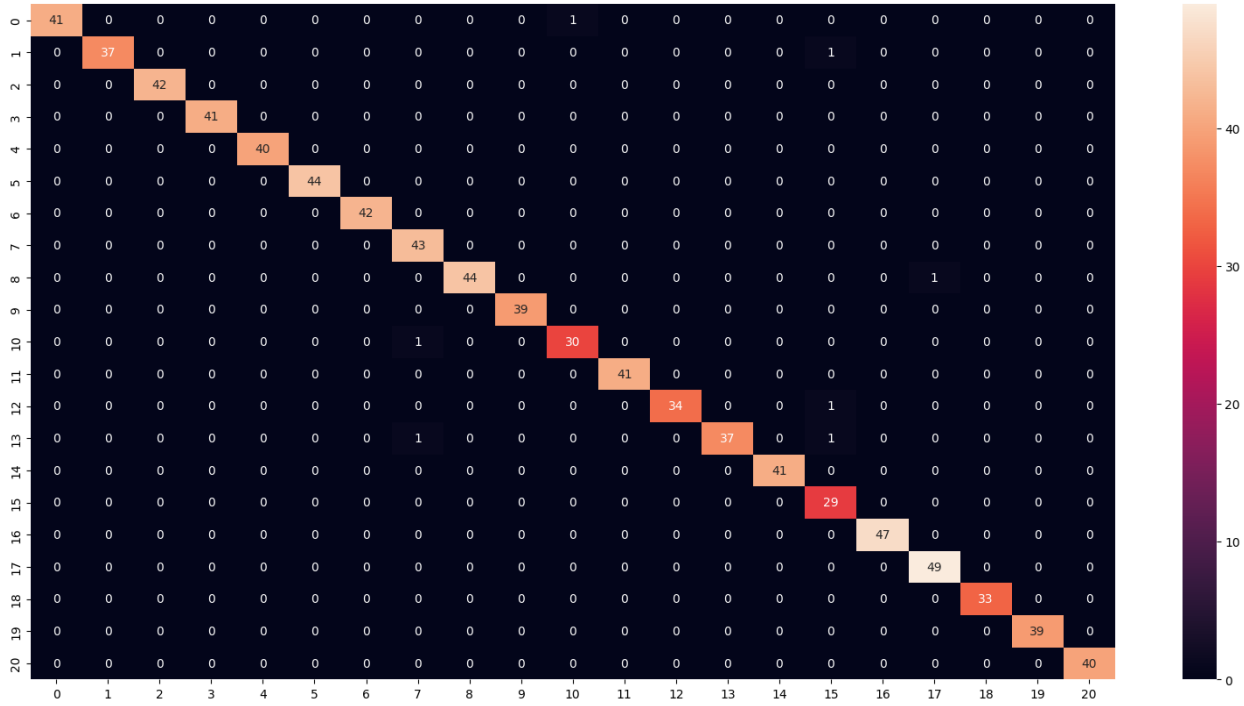
A.1 - Heatmap of Multinomial Naive Bayes Bigram Model



A.2 - Heatmap of Support Vector Machine RBF Kernel Model



A.3 - Heatmap of Recurrent Neural Network



Appendix B - Classification Report of Models

B.1 - Classification Report of Multinomial Naive Bayes Bigram Model

	precision	recall	f1-score	support
bg	1.00	0.98	0.99	1339
cs	0.98	0.96	0.97	1720
da	0.99	0.97	0.98	1837
de	0.96	0.95	0.96	1887
el	0.98	0.99	0.98	1880
en	0.95	0.95	0.95	1904
es	0.98	0.98	0.98	1863
et	0.96	0.97	0.97	1779
fi	0.99	0.98	0.99	1944
fr	0.97	0.98	0.98	1852
hu	1.00	0.98	0.99	1833
it	0.97	0.98	0.98	1845
lt	1.00	0.97	0.98	1724
lv	0.91	0.98	0.94	1775
nl	0.97	0.98	0.97	1910
pl	0.94	0.98	0.96	1774
pt	0.99	0.98	0.98	1852
ro	0.99	0.95	0.97	1313
sk	0.97	0.97	0.97	1811
sl	0.97	0.97	0.97	1741
sv	0.98	0.96	0.97	1832
accuracy			0.97	37415
macro avg	0.97	0.97	0.97	37415
weighted avg	0.97	0.97	0.97	37415

B.2 - Classification Report of Support Vector Machine RBF Kernel Model

	precision	recall	f1-score	support
0	1.00	0.99	1.00	1339
1	0.99	0.99	0.99	1720
2	0.97	0.99	0.98	1837
3	0.99	0.99	0.99	1887
4	0.99	0.99	0.99	1880
5	0.96	0.99	0.98	1904
6	0.99	0.99	0.99	1863
7	0.99	0.99	0.99	1779
8	1.00	0.99	0.99	1944
9	0.99	0.99	0.99	1852
10	1.00	0.99	0.99	1833
11	0.99	0.99	0.99	1845
12	1.00	0.99	0.99	1724
13	1.00	0.99	1.00	1775
14	0.98	0.99	0.98	1910
15	0.99	1.00	0.99	1774
16	1.00	0.99	0.99	1852
17	1.00	0.98	0.99	1313
18	0.99	0.99	0.99	1811
19	1.00	0.99	0.99	1741
20	0.98	0.98	0.98	1832
accuracy			0.99	37415
macro avg	0.99	0.99	0.99	37415
weighted avg	0.99	0.99	0.99	37415

B.3 - Classification Report of Recurrent Neural Network

	precision	recall	f1-score	support
0	1.00	0.98	0.99	42
1	1.00	0.97	0.99	38
2	1.00	1.00	1.00	42
3	1.00	1.00	1.00	41
4	1.00	1.00	1.00	40
5	1.00	1.00	1.00	44
6	1.00	1.00	1.00	42
7	0.96	1.00	0.98	43
8	1.00	0.98	0.99	45
9	1.00	1.00	1.00	39
10	0.97	0.97	0.97	31
11	1.00	1.00	1.00	41
12	1.00	0.97	0.99	35
13	1.00	0.95	0.97	39
14	1.00	1.00	1.00	41
15	0.91	1.00	0.95	29
16	1.00	1.00	1.00	47
17	0.98	1.00	0.99	49
18	1.00	1.00	1.00	33
19	1.00	1.00	1.00	39
20	1.00	1.00	1.00	40
accuracy			0.99	840
macro avg	0.99	0.99	0.99	840
weighted avg	0.99	0.99	0.99	840

Appendix C - Cluster Bar Graph of Model Accuracies to Languages