Lessons from a letter based language classifier

Akash Singh[1], Mahshid Mohammadalitajrishi[2], Vyom bhardwaj[3]

***Abstract*—This report analyses different classification methods to predict the categories of language from corpus letter distribution. We evaluate three different machine learning techniques to compare and contrast their suitability for the task of letter based language classification. We performed this classification operation using Centroid-based text classification and term frequency-inverse conversation frequency (tf-tdf) approach. The report explains the feature engineering methodologies used and their outputs. We experiment and evaluate the accuracy and error for four different classifiers: Naïve Bayes, Logistic Regression, k-Nearest Neighbors and Support Vector Machines(SVM). We produce our findings and lessons learned with respect to each classification above. Finally, we explain and discuss the reasoning behind our design choices and the corresponding results. Source code for all the simulations can be found in the given link: `https://bitbucket.org/akashsingh93/ml_nlp_proj2/src/8df1f6cc17134b294e885f2395adaefd8a7c1a98?at=master`**

I. Introduction

Classification approaches have been used for identifying the language of conversations, including Bayesian classification [1], relative entropy-based classification [2], centroid-based classification [3], Markov models [4], decision trees [5], neural networks [6], support vector machines (SVM) (Zhai et al., 2006), and multiple linear regression [7]. As a result of plenty of works, language identification is considered as a solved problem. However, while the methods proposed in these studies were analyzed in text classification based but in this report we use the similar classification approach to perform language based text classification but over a distribution of letters.

II. Methodology

A. Dataset

The dataset provided was 3 csv files. One of them was the training-set which had 276517 conversations from language corpus. The 2nd one was the category of each of these conversation in the training-set. A common id column was maintained in both conversations to map the category in the category csv to the conversation in the former one. The categories were one of the following 5: 0:Slovak, 1: French, 2: Spanish, 3: German, and 4: Polish. A 3rd testset was also provided which includes another 118508 letter distribution from corpus conversations. The task at hand was to predict the category of each of these letter distribution in the 3rd csv file based on a model constructed using the abstracts in the training-set and their categories in the 2nd file. An additional random predictor csv file was provided, but it has not been used in performing any of the simulations mentioned in this report. However, this test set is a distribution of letters obtained from corpus of different languages. The interesting point to note is that the training sample consists of sentences from various corpus whereas the test is a distribution of letters from various corpus of different languages. However, the predictions were to be submitted to the on-line learning platform kaggle. [8] to validate its accuracy and error. Hence, we divided the given training set into two portions of 80% and 20% except for Logistic regression where the split was 75% and 25%. The 80% of the samples were used to predict the categories of the remaining 20%. Then we used our predictions on this 20% to validate the accuracy of our model. Once we had reached a level of satisfactory accuracy (after testing with different parameters for the model), we used the complete set of 276517 training data to classify the original test-set. We submitted the results of this predictions to the on-line learning platform to validate how good our predictions were. We experimented the same with four different classifiers and selected the best two for the final submission to be the predictions from the classifier for which the best rate of accuracy was reached.

B. Data cleaning

Before going into feature extraction phase, we cleaned all the abstracts to remove every unnecessary details. We used *nlp* libraries of npm for data cleaning. This is because I found this library close-hauled, covering all the languages required for the classification problem. First both the datasets: test and training dataset were cleaned with the following criteria:

1) Change all the letters to lower case
2) All the emoiji's were removed
3) Urls in the conversations were removed
4) All the numerical values in the datasets were removed
5) removed all punctuation marks and symbols

Since, we were not sure of the test dataset, therefore we created two separate training datasets. We further performed the following operations to obtain the first training dataset using the **nlp** library of **npm**[1]. Dataset1:

1) removed the redundant *stop-words*[2]
2) *stemmed* all the remaining words to reduce them into their root form. Stemming reduces different patterns of the same word to its original form (by removing suffixes), hence allowing us to treat them all as one (ex: message, messages, messaging and messaged will all be treated as one word of the root form.)
3) *lemmatized* the words to transform them to its lemma3 form. It is different from stemming in that lemmatizing

[1] akash.singh@mail.mcgill.ca \ 260728046
[2] Mahshid.Mohammadalitajrishi@mail.mcgill.ca \ 260818751
[3] vyom.bhardwaj@mail.mcgill.ca \ 260632961

[1]https://www.npmjs.com/browse/keyword/nlp

[2]words that are very common in the English language and those which occur in every conversation. (ex: the, is, a, are and etc. in English)

does a morphological analysis in converting the words whilst the former just does a crude reduction.

Dataset 2:
In the second dataset we just used the cleaned dataset for model accuracy prediction, without performing any of the above operations.
Dataset 1 was not giving good results as the accuracy via Naïve bayes and logistic regression varied between 0.30 and 0.40. Therefore, we have used dataset 2 for all the predictions mentioned in this report.

Lesson: For classification of languages based on letter distribution, directly using lemmatization or stemming does not produce effective dataset for classification. Instead using a cleaned data produces a more effective results.

C. Training methods

1) Naïve Bayes: Naïve Bayes is an algorithm based on probability depends on Bayes' theorem which is used for classification. In Naïve Bayes algorithm method we need to calculate probabilities to make predictions. At first, we need to separate data by class, then Calculate the mean and the standard deviation for each attribute. At the end, summarize the attributes by each class. Gaussian Probability Density Function is used for prediction and estimate the accuracy.

This classifier was used to enhance the accuracy by taking the account of probabilistic approach. The authors identified this algorithm because of it's flexibility and having the property which makes each feature mutually independent. The algorithm was built using basic statistical logics of mean, standard deviation and Gaussian distribution. However the implementation and predictive approach were directly dependent on feature extraction using techniques like n-gram, which is the part of stemming and lemmatization, the process also included the tokenization to convert the text into individual elements. It's really critical to have feature vector before applying the naïve bayes algorithm. Naïve Bayes is an eager learning algorithm, which implies it learns a model from a training dataset as soon as the data becomes available. Once the model is learned, the training data does not have to be re-evaluated in order to make a new prediction.

Naïve Bayes Math representation:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$P(c|x)$ =Posterior Probability
$P(x|c)$ = Conditional Probability
$P(c)$ = Prior Probability
$P(x)$ = Evidence

2) Logistic Regression: We used one vs all method for classification using Logistic Regression. More precisely, we train 4 models to classify French/non-French, Slovak/non-Slovak, Spanish/non-Spanish, German/non-German, and Polish/non-Polish pairs. We then choose the prediction with highest probability among the five models for reporting the final outcome. For example, if the probability reported by the four models are 0.3 (Slovak), 0.4(French), 0.1 (Spanish), 0.5 (German) and 0.6 (Polish), then we choose our final outcome as 'Polish' since it has the highest probability amongst the five. We start with cleaning our data using $npm\ nlp$ package. Both training and prediction data are fed through same pipeline.

3) k-Nearest Neighbors (kNN) Classification: kNN search through the training dataset for the k-most similar instances based on distance metrics: *Minkowski Distance*, *Manhattan distance*and *Euclidean Distance* that are used in this algorithm. Then the top k train-set entries with the maximum distance/nearness values are filtered out. From the filtered set of k train-set entries we determine how many instances of each class has fallen within this top list. We then pick the class with maximum number of occurrences in the top-k list and assign it to the test-entry under consideration. It is noteworthy that the meaning of distance is a metric of closeness between the test-entry and each of the train samples. Thus, maximum distance above does not necessarily mean how far apart they are but the value of a score indicating their closeness. For a classification problem where this distance function is Euclidean distance (ED), then the nearest neighbors will be decided based on the minimum values of the ED.

Approach 1: For the case of corpus letter classification it was required of us to decide on a correct measure for the distance/nearness function. The nearness metric (between two different letter distribution of corpus) we considered in this project frequency of number matching letters in both the conversations. Frequency of each letter was a primary input for centroid based observation.

Approach 2: For every matching pair of letters between a test-conversation and a train-conversation, we calculate the tf-idf of those two letters, to be the nearness measure contributing to the score of that training-sample. We then sum all such individual letter scores per matching letter and take the total as the final score of that specific training-set against the given test. For every new letter distribution to be predicted, we do this calculation against each and every sample conversation in the training set. Thus, we have a score metric indication of nearness of each sample in the training set to the new-letter distribution. We then filter out the training samples with the top k scores calculated in this fashion. Finally, we determine the abstract-category (Slovak, French, German, Spanish, Polish) with the most frequency in the filtered top k results and assign it as the category of the new-abstract. In our experiments we have kept k to be 5. We don't do any further experiments for different value of k given that we are able to achieve good accuracy for predictions with k = 5 (This k-value has nothing to do with the cardinality of the classification which happens to be 5 coincidently). In the occasional instance of more than one category being present in the top k set with equal frequency (i.e: French:4, Spanish:4, Slocak:3, German:3, and Polish: 2); we calculate the total score-per-category by adding the individual scores of train samples in the top k for a specific category. We then assign the category with the highest

cumulative total score to the test letter distribution.

III. Feature Extraction

A. Centroid [3]

One of the feature which we use for extract feature is centroid-based approach. The centroid concept indicate the central value of a set. By this way, each centroid represent a class of dataset. Each conversation is dedicated by a vector which create by central value. For language identification we first calculate the frequency of characters mentioned in the dataset and then scale it based on the central value of the dataset. Algorithm 1 explains the centroid approach for feature extraction. The centroid values were calculated for the characters shown in :

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'ß', 'à', 'á', 'â', 'ä', 'æ',
'ç', 'è', 'é', 'ê', 'ì', 'í', 'î', 'ï', 'ñ', 'ò', 'ó', 'ô', 'ö', 'ù', 'ú', 'û',
'ü', 'ý', 'ÿ', 'č', 'ď', 'ě', 'ľ', 'ť', 'ň', 'œ', 'ř', 'ř', 'š', 'ť', 'ů', 'ž',
'!', '?', '¿', '¡']
```

Fig. 1. For each character a centroid value is calculated with respect the entire dataset. Each conversation is represented as a feature vector of the following character-set.

B. TF-IDF

We extracted the distinct set of letters/tokens that appears in any of the training-set by using the *TfidfVectorizer()* class of the scikit-learn toolkit with the following parameters as input: *decode_error='strict', analyzer='char',min_df=0)*.

IV. Feature creation per algorithm

[9]

A. Logistic regression

From the cleaned data set, we choose the first 'n' most commonly used letters as our feature set. The choice of n was kept as hyperparameter. With increasing 'n', the trade off was between accuracy vs over fitting and execution time. We tested using n = 10, 20 and 67. The first big jump in accuracy was noticed when increasing number of features from 10 to 20. Similarly, increasing features from 20 to 67 also resulted in stellar increase in accuracy rates. Therefore, we included all the characters mentioned in section 3A. This is reflected in figure 2.

Regularization parameter for Logistic Regression: For the choice of regularization parameter, we split our training data into 75-25 ratio and used L2 regularization. We trained our model using 75 percent of the data and varied lambda between 1e-1 and 1e4. For, all values of lambda, our training and cross validation accuracy remained almost same. This most likely suggests that the weights are already very small to warrant any significant change when using regularization. This is reflected in figure 3.

Lesson: This likely suggests that for such tasks the weights are already very small to warrant any significant change when using regularization.

Algorithm 1: Training and Testing Algorithms for ICF($sf = 10$ and $\epsilon = 0.001$)

input : Training Set
output: Centroid vector $\vec{c_i}$ for each class c_i
Local Variable:
$\text{freq}_{i,k}$: total frequency of letter l_k in class c_i
$\text{freq}_{i,k}$: total frequency of letter l_k in the corpus
begin
end
for *each class* c_i **do**
  for *each conversation* d_i *in class* c_i **do**
    for *each letter* l_k *in conversation* d_i **do**
      $\text{freq}_{i,k} + 1$;
      ; // Increment class frequency of letter l_k
      freq_{k+1};
      ; // Increment corpus frequency of letter l_k
    end
  end
end
for *each class* c_i **do**
  for *each letter* l_k **do**
    $c_{ik} = \log(((\text{freq}_{i,k} + 0.001)/\text{freq}_k) * 10)$;
    ; // calculate centroid value $c_{ik}(sf = 10)$
  end
end
input : Test conversation d
Centroid vector $\vec{c_i}$ for each class c_i
output: Class index of conversation d
Local variable:
$\vec{f}$: frequency vector of letters in the corpus
begin
end
for *each letter* l_k *in conversation* d **do**
  $f_k = f_k + 1$;
  ; // Increment conversation frequency of letter l_k
end
for *each class* c_i **do**
  compute $\text{sim}(\vec{f}, \vec{c_i})$;
end
return $\arg\max_i \text{sim}(\vec{f}, \vec{c_i})$;

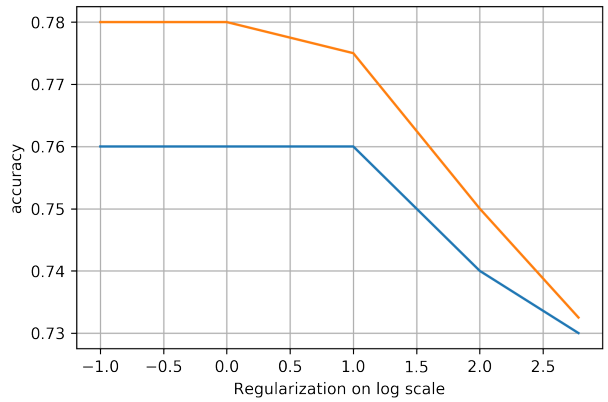

Fig. 2. Complexity Vs Accuracy

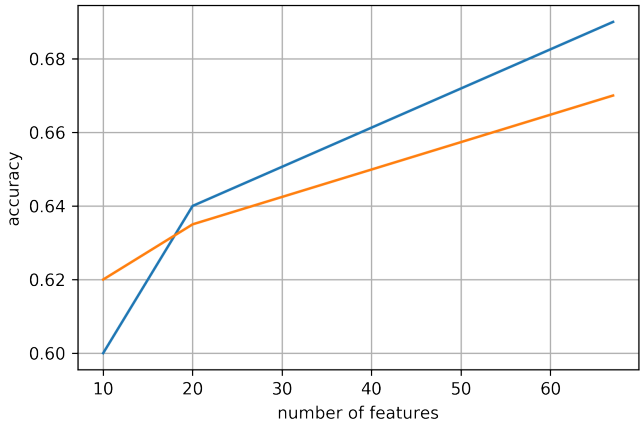

Fig. 3. Regularization Vs Accuracy

V. Results

A. Logistic Regression

Based on our 589 feature Logistic Regression model, score on final prediction set as reported by Kaggle is 0.78. Also, computationally cheaper, 67 feature model scored 0.63 on Kaggle. Please note that this is neither our final model for submission nor our final score. The scores mentioned in this section are for reference only.

TABLE I
Accuracy obtained on the training set using centroid based approach

	Centroid	Execution Time	Num feature
SVM	0.76422	≈2hrs	67
KNN	0.75056	≈30min	67
Naïve Bayes	0.73262	≈2min	67
Logistic Regression	0.73118	≈2min	67

B. kNN Results

We opted to carry-out an initial prediction on a split-set of the given training data. The predictions on the above split resulted in an accuracy of 89%. Thus, the entire 100% of the given training set was used to predict the categories of the original test-set. These final predictions were submitted to the online kaggle system. The accuracy-scores of submissions to kaggle is 0.78 as shown in table7.

Lesson: An important lesson is that the prediction score increases with increase in k for tasks as such. Higher the number of closely related samples taken into account for prediction, the better the results were. One important point in terms of increasing k was that, this reduced the chances of a conversation being closely related to equal number of training samples for more than one specific category. A higher value for k always ensured that one specific category out of the five occurred as majority among the filtered k samples. Hence, it made the final classification step straight forward.

TABLE II
SVM performance metric

Category	Precision	Recall	f1-score
Slovak	0.85	0.86	0.85
French	0.91	0.91	0.91
Spanish	0.82	0.87	0.84
German	0.88	0.88	0.88
Polish	0.93	0.94	0.93

C. Support Vector Machine

Below are the results obtained for the various features and kernels used for SVM classification.

TABLE III
SVM performance metric

Category	Precision	Recall	f1-score
Slovak	0.85	0.86	0.85
French	0.91	0.91	0.91
Spanish	0.82	0.87	0.84
German	0.88	0.88	0.88
Polish	0.93	0.94	0.93

TABLE IV
SVM Scores for centroid and tf-idf approach

Kernel	Feature	Score(tf-idf)	Score(centroid)
linear	200	0.7521	0.6912
	400	0.8728	NA
	589	0.8968	NA
rbf	200	0.7481	0.6678
	400	0.8163	NA
	589	0.8623	NA

Lesson: We get a better score for non-liner classifiers. This showcases that letter based language classification is a non-linear problem.

TABLE V
RESULTS

	tf-idf	Execution Time	Num feature
SVM	0.85708	≈8hrs	589
KNN	0.89147	≈4hrs	589
Naïve Bayes	0.82671	≈2min	589
Logistic Regression	0.87666	≈5min	589

Lesson: It was noticed that there was no significant increase or change in the score by increasing the no of features/letter centroids taken into consideration for the model. The only advantage in terms of tuning the feature count was that we were able to achieve better execution times for the predictions. By reducing the number of features the predictions over the entire test set could be completed at a much lesser duration.

TABLE VI
RESULTS OF CLASSIFICATION

	Centroid	tf-idf
SVM	0.69422	0.75708
KNN	0.67056	0.78147
Naïve Bayes	0.63262	0.66671
Logistic Regression	0.63118	0.77666

VI. CONCLUSION: BAGGING FOR IMPROVEMENT

The final output after bagging of predictions with 2 algorithms: kNN and logistic regression improved our score by 0.00191. Even though this may seem too small an improvement we were able to go up by two positions in the leaderboard on Kaggle.

TABLE VII
ALGORITHM KAGGLE SCORES

kNN	0.78147
SVM	0.75708
Naïve Bayes	0.66671
Logistic Regression	0.77666
Neural Networks	0.51198
Ensembled prediction	0.78338

Lesson: Neural networks do not produce effective predictors when using a feature vector with multiple collinear features. For instance in case of centroid approach multiple features were collinear, therefore we failed to obtain good results.

VII. DISCUSSION

The classification task at hand was experimented with four different classification algorithms Logistic Regression, k-Nearest Neighbors, Support Vector Machines and Naïve Bayes. Predictions with these algorithms were submitted to the kaggle platform. And the predictions from Logistic regression and Naïve bayes was used in for an ensemble prediction. However, the best prediction scores (for the algorithms which were coded) in terms of original training set and the original test set was observed for the k-Nearest Neighbors classifier. It was observed that the prediction score was increasing with k. The higher the number of closely related samples taken into account for prediction, the better the results were. One important point in terms of increasing k was that, this reduced the chances of a conversation being closely related to equal number of training samples for more than one specific category. A higher value for k always ensured that one specific category out of the five occurred as majority among the filtered k samples. Hence, it made the final classification step straight forward. Another important metric in terms of the classification problem was the number of features to be considered. In this problem the features were letters in a given set of conversations from different corpus. More specifically it was the tf-idf values of these letters. It was noticed that there was no significant increase or change in the score by increasing the no of features/letters taken into consideration for the model. The only advantage in terms of tuning the feature count was that we were able to achieve better execution times for the predictions. By reducing the number of features the predictions over the entire test set could be completed at a much lesser duration. However, it was observed that by having a higher and a richer set of features/letters taken into account we could assure that for every matching pair of letters between the test-sample and the train-sample, that this letter was present in the feature set. Reducing the feature-letters meant that for some of the matching letters between the test and train set, it was possible that this letter is not present in the feature set. Hence, this affected how this train sample was scored for the given test letter distribution.

VIII. CONTRIBUTIONS

We hereby state that all the work presented in this report is that of the authors. Each author played a role in feature selection. Akash did the data cleaning and feature extraction. Mahshid implemented the kNN model and also contributes in tf-idf feature extraction whilst Vyome implemented Naïve Bayes algorithm for language classification. Statistical tests were performed by Akash. However, all authors contributed to the approach, design of the feature sets and the written report.

REFERENCES

[1] S. Armstrong-Warwick, H. S. Thompson, D. McKelvie, and D. Petitpierre, "Data in your language: the eci multilingual corpus 1," in *Proceedings of the International Workshop on Sharable Natural Language Resources*, 1994, pp. 97–106.

[2] D. Hawking and P. Thistlewaite, "Methods for information server selection," *ACM Transactions on Information Systems (TOIS)*, vol. 17, no. 1, pp. 40–76, 1999.

[3] H. Takçı and T. Güngör, "A high performance centroid-based classification approach for language identification," *Pattern Recognition Letters*, vol. 33, no. 16, pp. 2077–2084, 2012.

[4] C. Kruengkrai, P. Srichaivattana, V. Sornlertlamvanich, and H. Isahara, "Language identification based on string kernels," in *Communications and Information Technology, 2005. ISCIT 2005. IEEE International Symposium on*, vol. 2. IEEE, 2005, pp. 926–929.

[5] E. Leopold and J. Kindermann, "Text categorization with support vector machines. how to represent texts in input space?" *Machine Learning*, vol. 46, no. 1-3, pp. 423–444, 2002.

[6] J. Tian and J. Suontausta, "Scalable neural network based language identification from written text," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 1. IEEE, 2003, pp. I–48.

[7] P. K. Ghosh, R. J. Majithiya, M. L. Umrethia, and R. S. Murthy, "Design and development of microemulsion drug delivery system of acyclovir for improvement of oral bioavailability," *AAPS pharmscitech*, vol. 7, no. 3, pp. E172–E177, 2006.

[8] A. Narayanan, E. Shi, and B. I. Rubinstein, "Link prediction by de-anonymization: How we won the kaggle social network challenge," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 1825–1834.

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.