

Final Project

Low Resource Language Text Classification

CS 505 Natural Language Processing

Team members:

Sichao Yu

Nicholas Zotalis

Keith Tyser

Aditya Agrawal

Minh Le Nguyen

Adam Clay



Link to Github:

<https://github.com/nicholaszotalis/505Project>



Link to GoogleSlides:

 CS 505 NLP Presentation



Table of contents

Introduction and Motivation	1
Project Description	1
Approach	1
Model Descriptions and Results	1
Logistic Regression	2
Combine models	2
Neural Network	2
Linear SVM	2
Rocchio	2
Neural Network	2
SGD	2
MultiNB	2
Random Forest	2
SGD	2
Bagging	2
KNN	2
Boosting	2
Fine-tune Roberta	2
Discussion and Interesting Insights	3
Conclusion	4
Future Tasks	4
References	4

Introduction and Motivation

“Natural Language Processing (NLP) research predominantly focuses on developing methods that work well for English despite the many positive benefits of working on other languages.” ^[1]

Algorithms for text classification have come a long way, but classifying long texts and working with low resource languages still faces a lot of difficulties. In CS 505 Natural Language Processing class at Boston University, we have the chance to participate in a low resource text classification challenge of the Chichewa language on Zindi.africa ^[2].

Project Description

The project consists of only 1437 news titles in Chichewa language as training instances, with 20 class labels, including: Politics, Social, Religion, Law/Order, Social Issues, Health, Economy, Farming, Sports, Educations, Relationships, WildLife Environment, Opinion/Essay, Local Chiefs, Culture, Witch Craft, Music, Transport, Arts and Crafts, and Flooding.

Our goal is to successfully classify 620 news titles in Chichewa language into their correct class among the 20 classes.

Approach

We have learned a lot of preprocessing methods and text classification techniques. However, we do not know what might work the best with Chichewa language. Thus, we decided to try all the techniques we know, with different combinations of preprocessing methods.











































































































Considering the size of our training data, our classification solutions start with classic classifiers including Logistic Regression, Linear SVM, Rocchio, SGD, KNN, and Multinomial Naive Bayes. Then we implemented ensemble models including Gradient Boosting, Bagging and Random Forest. We also tried deep learning approaches including neural networks and multi-language pre-trained Roberta. Our preprocessing methods include Word2Vec, Countvectorizer, TF-IDF Transformer, NLTK Wordnet Lemmatizer, removing stopwords, removing punctuations, Random Over Sampling, and Smote.

We found some more unlabeled data from scraping the web for additional news articles in Chichewa, primarily from The Nation Malawi Online.^[4] These articles were useful for preprocessing, mainly in training a Word2Vec model to bolster its vocabulary and word associations, but were not used in training in any capacity, as the nature of this problem was to train a classification model with a very small amount of data. There were additional closed-source datasets in Chichewa online, but we did not use these for this project for the same reason. We obtained a list of pseudo-stopwords by identifying the 1000 most common words in the Chichewa language.^[3] This is clearly a flawed method for obtaining these stopwords but this is part of the challenge of working with a low-resource language.

Model Descriptions and Results

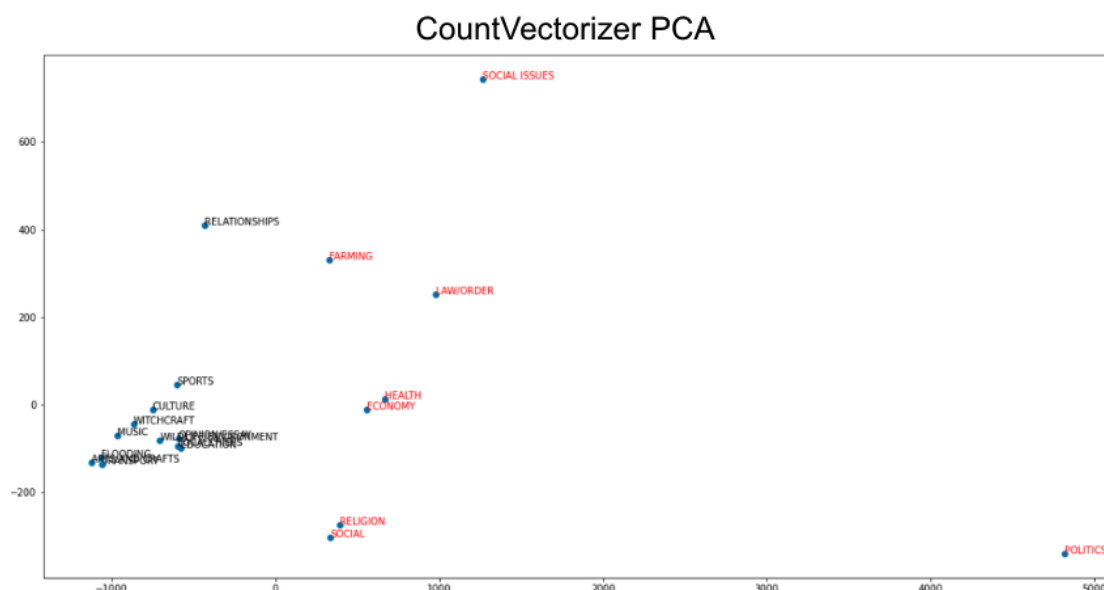
Since there is a limit on the number of pages we can write for this report, we will only include the combinations that perform well on the testing data.

The best participant of the challenge achieved 71% accuracy. Therefore, we consider any model that gets > 60% accuracy on the website a good performer. The processing pipeline for each model is given below.

 : Included in preprocessing pipeline  : Not included in preprocessing pipeline									
	Word2Vec	Count Vectorizer	NLTK Wordnet Lemmatizer	Smote	TF-IDF Transformer	Remove Stopwords	Remove punctuation	Over Sampling	Results (on website's test data)
Logistic Regression									66.77%
Combine models	Most votes on each predicted entry								65.1%
Neural Network									65.1%
Linear SVM									63.87%
Rocchio									64.19%
Neural Network									65.1%
SGD									63.22%
MultiNB									62.25%
Random Forest									61.93%
SGD									61.61%
Bagging									60.65%
KNN									57.7%
Boosting									58.71%
Fine-tune Roberta									53.54%

The best result we achieved was 66.77% accuracy, with Logistic Regression, graded by the website on the given test set. We made the top 10 best performers, out of 784 participants.

Discussion and Interesting Insights



We used CountVectorizer on all of our training data to obtain a vector representation for each class in the dataset, then used PCA to reduce the representations to two dimensions to obtain the graph above. A primary issue in this project was the imbalance of data for each class. The classes with a lot of data points are in red, and those with fewer points are in black. For each of the high-representation classes, we were able to differentiate between them well, with politics being by far the most represented class and the most clearly isolated vector. The classes with lower representation are clumped together because there simply was not enough data present to differentiate between them. This disparity was apparent when training and testing our models, as they were able to predict text belonging to the highly-represented classes with far greater accuracy than the classes with much less data. We tried to mitigate this effect with oversampling and SMOTE, which marginally helped to predict classes with lower-representation.

Saulos Chilima (1)	Current Vice President of Malawi
Nawo Pamando (2)	Chairman
Chipanicho (3)	The party
Mneneli (6)	Speaker
Ulamuliro Wa (7)	Ruler
Chipanichi (8)	Party
Wachiwiri Kwa (9)	Second to
Msonkhano Waukulu (10)	General Assembly

Above are the 10 closest tokens to “pulezidenti,” the word in Chichewa for president, in our best Word2Vec model, and their English translations on the right. For classes that had a lot of data, our

word2vec model was able to learn very good word associations, as is the case for politics as demonstrated above. However, the model was not able to learn as good associations for more general words, or words specific to one of the classes with less representation, so while this model was able to learn some interesting associations, it was not very helpful as a preprocessing step for our models. Additionally, the Chichewa language has a dissimilar syntax to Indo-European languages where each token generally has a stand-alone meaning (or is at least easily separable, such as the German language), where word2vec does very well in learning word associations: in the syntax of the Chichewa language what a tokenizer would see as a token is translated as a whole phrase, where a single character could change the meaning of the whole phrase. For this reason a lemmatizer of the Chichewa language would be very useful for training word embeddings, but that is likely a difficult linguistics task in its own right.

Conclusion

Due to the extremely small amount of training data available this was a very difficult task. Since Chichewa is such a low-resource language, it was challenging to find any additional training data online to supplement what we already had. Additionally, the large class imbalance made it tough to accurately predict labels for which we only had a few samples. As a result, due to the limited training data we found that using simpler models and preprocessing methods like logistic regression and TF-IDF achieved the best performance. We also found techniques like SMOTE and oversampling helped improve performance of those models because of the class imbalance. If more training data is able to be accessed in the future, we would likely see our deep learning models improve in performance. In addition, a better list of Chichewa stopwords and a Chichewa lemmatizer could improve the performance of many of these models. Low-resource language also means limited preprocessing tools. Our list of stopwords was a very crude analogue.

Future Tasks

For future next steps, we would recommend trying to obtain more training data from the internet. We were able to find a closed access dataset on the internet that had an additional 3400 articles. We did not use this dataset for the project, but using it would likely improve the performance of all of our models. We were also able to scrape some news articles in Chichewa from the internet to train our Word2Vec embeddings but they did not have any labels. Scraping more of these articles and manually labeling them could be another method to obtain more data. Finally, we would recommend experimenting with more sampling techniques due to the class imbalance and fine-tuning the ones we already tried like SMOTE and oversampling. These methods had a significant impact on increasing the performance of our models, so perhaps there are other sampling methods that would increase performance more.

References

- 1) Sebastian Ruder. (2020, August 3). *Why you should do NLP beyond English*. Sebastian Ruder. Retrieved May 3, 2022, from <https://ruder.io/nlp-beyond-english/>

- 2) *AI4D Malawi News Classification Challenge*. Zindi. (n.d.). Retrieved May 3, 2022, from <https://zindi.africa/competitions/ai4d-malawi-news-classification-challenge>
- 3) Words. (2022, February 26). *1000 most common Chichewa Words - 100% best list of words*. 1000 Most Common Words. Retrieved May 4, 2022, from <https://1000mostcommonwords.com/1000-most-common-chichewa-words/>
- 4) *Chichewa*. The Nation Online. (n.d.). Retrieved May 4, 2022, from <https://www.mwnation.com/section/chichewa/>