Word Etymology Predictor English

Stan Goodwin: 20449042
December 2023
Link to git repo here
Link to google drive datasets here

Introduction

In the following report I will outline the steps taken to build a word etymology predictor for the English language. It will contain the details of the task and the dataset as well as the details of the baseline heuristic experiments, supervised machine learning experiments, BERT fine-tuner and the ChatGPT prompting results. The report will analyse the differences between these methods, the strengths and weaknesses of the models as well as the suitability of these approaches for this task. This report also serves as documentation to be read in conjunction with the jupyter notebooks that accompany it.

1. Description of task and dataset

The task presented in this report is to build, or test the proficiency of, a predictor that can correctly classify the etymology or a word in the English language. “Etymology is the investigation of word histories'” (Durkin). For example, “friar” was taken from the Old French word “frere” which means brother which came from the latin word “frater”(Durkin). In this report we will attempt to build a model that is capable of predicting the language from which an english word was derived from. For example, if we give the model the word “democracy” we would want the model to tell us that it comes from Greek. We will do this by designing hand-crafted, heuristic rules, building supervised machine learning models, fine-tuning a BERT model as well as analysing the effectiveness of ChatGPT in solving this problem.

For this project I will be using the Etymological Wordnet dataset from Hasso Plattner Institute. The data here is mostly mined form wiktionary with some data also manually added. The dataset consists of three columns. Two columns contain a word as well as the language that word is in. This is in the format “language:word”. The language information is in the form of ISO-639-3 language codes. The other column contains the type of etymological relationship between the two words.

	0	1	2
0	aaq: Pawanobskewi	rel:etymological_origin_of	eng: Penobscot
1	aaq: senabe	rel:etymological_origin_of	eng: sannup
2	abe: waniigan	rel:etymological_origin_of	eng: wangan
3	abe: waniigan	rel:etymological_origin_of	eng: wannigan
4	abs: beta	rel:etymological_origin_of	zsm: beta

Figure 1.1: Etymology Dataset

When cleaned, filtered and processed, I was left with the etymological origin for each word in modern English. However, I discovered that the origins of English words consisted of 238 different languages, meaning if I were to build a machine learning model, there would be 238 different classes to predict. As well as this, I discovered that outside the top ten languages represented in that group, no language had more than 400 words represented in the dataset. If I continued with this I would be dealing with an extremely unbalanced dataset that is heavily skewed towards languages such as Latin or French. As a result of this I decided to only take the top ten languages represented in the dataset as this would cover the majority of English words while having a more balanced dataset. This lead to the following languages that would represent my classes:

```
Top ten languages:
Latin
Middle English
French
Ancient Greek
Old French
Old English
Middle French
Japanese
Italian
Spanish
```

Figure 1.2

As you can see in figure 1.2, the top ten languages include “French”, “Middle French” and “Old French”. These languages are very similar to each other and would be extremely difficult to tell apart so as a result of this I decided to combine them all into just “French”. I also decided to do the same for “Middle English” and “Old English”. From this point on I may reference a word in modern English coming from “English”, keep in mind that this means it comes from middle or old English.

2. Heuristic Baseline Prediction

For my heuristic baseline predictor, I decided to trial the idea that different languages will have some letter or character combinations that are somewhat unique to that language. Of

course this is not always the case and can become difficult especially when trying to differentiate between languages of the same family such as the romantic languages. To come up with these combinations I employed a mix of my own knowledge of several languages, some trial and error as well as asking ChatGPT. In ChatGPT I would give a prompt such as "Give me common letter combinations for the French language". I would then cherry pick the result I believed would be the most useful or unique. I then defined a function which was comprised of if and else statements as you can see below:

```
def ept_classifier_rule(word):
    if "ua" in word or "eca" in word or "us" in word or "um" in word or "ia" in word or "ex" in word or "lib" in word or "oc" in word or "au" in word:
        return("Latin")
    elif "ia" in word or "ee" in word or "oy" in word or "ow" in word or "ev" in word or "ng" in word or "ch" in word or "th" in word or "ph" in word:
        return("Ancient Greek")
    elif "gh" in word or "kn" in word or "wr" in word or "ghth" in word or "tion" in word or "ou" in word:
        return("English")
    elif "au" in word or "eau" in word or "ai" in word or "ei" in word or "oi" in word or "eu" in word or "gn" in word or "ill" in word:
        return("French")
    elif "ka" in word or "ki" in word or "ku" in word or "ke" in word or "ko" in word:
        return("Japanese")
    elif "ce" in word or "ge" in word or "sce" in word or "gh" in word or "gli" in word or "io" in word or "za" in word:
        return("Italian")
    elif "ll" in word or "qu" in word or "gu" in word or "rr" in word or "gui" in word or "ves" in word:
        return("Spanish")
    else:
        return("Latin")
```

Figure 2.1: Baseline Test

An important part of this function is the order of the elif statements. Latin is the most common language in the etymology data so I have put it first to try and "catch" any words that might also match with another language and Spanish is the least common thus why it is second last. Again, Latin is then given as an answer if nothing matches as it is the most common and a word at random has the biggest probability of being Latin.

This baseline test achieves an accuracy of 0.3175 which isn't good but could be worse given there are seven different classes to predict. However it does provide a good baseline that should definitely be able to be improved upon by using supervised machine learning. I use a train_test_split function from sklearn with a random state of 42 to split and test the data so that we can compare this to all future results.

3. Supervised Machine Learning Experiment

Before performing any machine learning I first had to vectorize the word data. I used the TfidfVectorizer with the parameters "analyzer='char'" and "ngram_range=(2,3)". Since eac word only occurs once anyway I could have used CountVectorizer as it didn't have any effect on the results so I stayed with tfidf. I specified the model to look at the characters instead of the words as each input only has one word and I found that a smaller ngram range yielded better results through trial and error.

The best three standard supervised machine learning algorithms were Multinomial Naive Bayes, Multinomial Logistic Regression and Support Vector Machines model. I also tried random forest and gradient boosting but the results for those models were sufficiently below the other three to ignore them. I then also built an ensemble model combining the three mentioned models. The following table shows a comparison of the results of the four models:

	Accuracy	Precision	Recall	F1
Naive Bayes	0.6238	0.6058	0.6238	0.5934
Log. Regression	0.6582	0.6622	0.6582	0.6415
SVM	0.6677	0.6608	0.6677	0.6599
Ensemble	0.6649	0.6617	0.6649	0.6499

As you can see from the table above the SVM model performed the best overall. The ensemble model was very similar to the logistic regression and SVM model hinting that all three models were getting the same things wrong.

After this testing I moved onto a neural network approach and built an LSTM model. I tweaked the model by trying a different amount of layers, several optimizers and loss functions. I also tried building a simple RNN as well as a CNN. However the best model ended up being an LSTM with twelve hidden layers using the adam optimizer and the sparse categorical cross entropy loss function. The LSTM, on the same testing data, was able to at best match the SVM model, and at worst match the naive bayes model with an accuracy of 0.6422 on my latest run of the model at the time of writing.

With the testing done, I decided to look into the best model I was able to build, the svm model. I created a confusion matrix to try and see where it was going wrong:

Figure 3.1: Confusion Matrix SVM

As you can see in figure 3.1, it seems that French and Latin in particular are getting mixed up by the model. This is probably due to the fact French and Latin are similar languages as well as the fact that Latin is a dominant class in this data and the model is probably skewed towards it. In an attempt to handle this issue, I decided to take French out of the data and see how that improved the model. After running the svm model on the dataset without French, the model performed as follows:

No French Data	SVM
Accuracy	0.7961
Precision	0.7905
Recall	0.7961
F1	0.7889

As you can see the model's performance dramatically improves simply by taking French out of the data. However, I decided not to move forward with this approach as I believe it is bad practice given the significant size of the French language represented in the data.

4. Fine-tuning BERT

For the fine-tuning of the BERT model I used BertForSequenceClassification as I found it performed marginally better compared to AutoModel. I also used bert-base-uncased. I was able to utilise my own GPU to fine tune the model locally. I had ten training epochs, with a train batch size of 64 and an evaluation batch size of 128 and a weight decay of 0.02. The results of fine-tuning the BERT model was an accuracy of 0.7095, a precision of 0.6420, a recall of 0.6040 and a F1 score of 0.6134. These are interesting as the model is able to achieve a higher accuracy than the supervised model but it has a lower precision, recall and F1 score.

5.

Chat GPT was given a random sample of 30 words and asked to give the etymological root language of each word. The accuracy of both zero shot and one shot prompts was both 0.7667 or 23 out of 30. What I found interesting about the results was that three of the seven it got wrong were a latin/french mix up and another one was a english/french mix up. These are the same mistakes that were reflected in all of the machine learning models so far and it seems as if ChatGPT of GPT 3.5 is no exception to this rule.

temporal	Latin	Latin
poltroon	French	French
venus	Latin	Latin
albescence	Latin	Latin
comma	Latin	Latin
ya	English	Japanese
derailleur	French	French
Septembrist	French	Latin
eutely	Ancient Greek	Ancient Greek
amelioration	French	Latin
calligraphy	Ancient Greek	Ancient Greek
field	English	English
personate	Latin	Latin
blush	English	English
mal	French	Latin
pinnatisect	Latin	Latin
demonize	Latin	Ancient Greek
flaccify	Latin	Latin
mercable	Latin	Latin
placation	Latin	Latin
omertà	Italian	Italian

pandoro	Italian	Italian
convex	Latin	Latin
frush	French	Middle English
tortile	Latin	Latin
tetrarchy	Latin	Ancient Greek
manteau	French	French
Aeolis	Ancient Greek	Ancient Greek
Smyrna	Ancient Greek	Ancient Greek
twire	English	English

6. Comparison of Results

The baseline test was by far the worst test however the idea of looking at the combinations of different characters ngrams was useful when applied to the supervised machine learning techniques. We were able to make tweaks to the vectorizer to achieve an accuracy of 0.6677 with the SVM model with similar precision, recall and F1 scores to support it. Moving to the fine-tuning of the BERT model, we were able to increase our accuracy at the cost of precision, recall and F1. An interesting area that could be explored in a future project is using sentences in different languages to predict the etymology of English words. However, this would most likely only work for languages that are written in the same alphabet as English which would rule out languages such as Greek and Japanese. Finally, ChatGPT was again able to increase the accuracy but getting 0.7667 on the random 30 samples. One of the most interesting things I found was that all models from naive bayes to chat gpt struggled with differentiating French and Latin, as well as French and English to a certain extent.

Taking a random sample of 30 from out dataset and testing on a model from each part of this project yielded the following results:

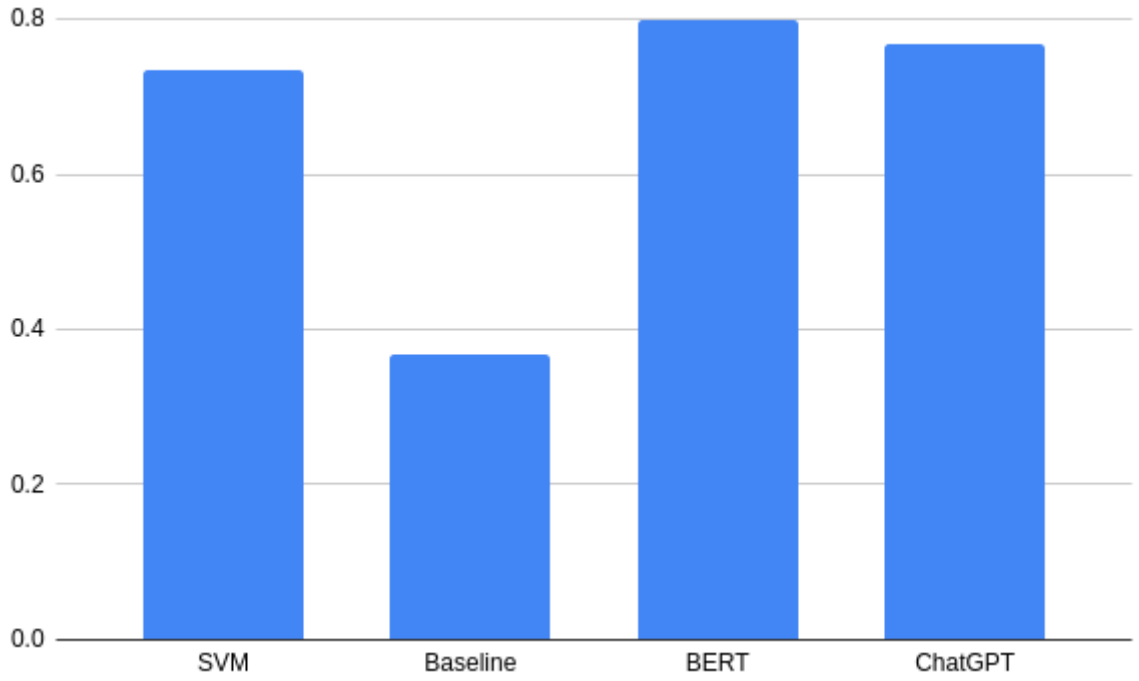

Figure 6.1: Random 30 Accuracy Comparison

Svm=0.7333 **Baseline=0.3667** **BERT=0.8** **ChatGPT=0.7667**

In this particular case the fine-tuned BERT model was able to outperform ChatGPT. To conclude the comparison, ChatGPT was the best but no model was able to achieve a high enough accuracy with precision, recall and F1 score to support it to be considered great at this task. The main issue found in all of these models is its inability to tell the difference between closely related languages. This is an inherent problem when attempting this task as can be proven by the struggle that ChatGPT has with this problem.

References

Durkin, Philip. The Oxford Guide to Etymology. Google Books, OUP Oxford, 24 July 2009, books.google.ie/books?hl=en&lr=&id=OiiQDwAAQBAJ&oi=fnd&pg=PR5&dq=word+etymology&ots=snG9iahNY9&sig=Lxs59Io6cVHX51QYViB02A5NpeE&redir_esc=y#v=onepage&q=word%20etymology&f=false. Accessed 18 Dec. 2023.