CS 388: Natural Language Processing Spring 2016

Project Report — Original Language Detection

Students: M. Denend, J. Hoffmann, L.Prakash *Prof. Ray Mooney*

Abstract

Detecting source language of written text is an interesting potential use for machine learning that has been used in the past. We take some of these past works and attempt to combine them together in order to detect the source language of novels that were translated from their original language into English. A previous work did this successfully, but did not attempt to test outside of the given domain, so we decided to put different novels for the languages we tested in a train set and a test set. We computed features on these novels and used machine learning to calculate the differences. With 5 languages and 4 train set novels per language of same-sentence size, we get a maximum classification accuracy of 65%.

1 Introduction

Our task was to determine the original language of translated text. Determining the language of origin for written text is something that can be of great importance to historians in order to trace back the path in which ancient novels may have been translated from, for example, the Bible. Doing so and using improving machine translation to translate a book back into prior languages may possibly open up avenues of research to better understand intended meanings of the first authors, as meaning can get lost when novels are translated. Furthermore, previous studies have shown that machine translators are much more accurate in identifying whether writing was from an original language, or translated, as mentioned in [1]. Our goal was to build upon past research. In particular, we found that [5] had some good results when trying to determine the original language, however, it only tested against slices of the same books, which we felt defeats any generalization ambition. We therefore decided to have our train set and test set computed from completely separate novels, which is one of our major contributions. We reused the horizontal slices of parse trees mentioned in [7], as well as some features from [5] and add parse features to this. We built our dataset in a manner like in [5], but decided to extend it by adding a couple more books and two more languages. By combining all our features, we reach a 65 % accuracy at the slice level.

2 Problem Definition

(a) Description

Our project is to take novels, freely available from Project Gutenberg [2], and translated from a wide variety of languages, and determine which language they originated from. To do this, we take a novel electronically available in plain text format, and run some algorithms to determine various features that we detect from the books. We group these into three types of features (to be elaborated upon more later): lexical features, parse features (horizontal slices), and homemade features. Once we've computed these features for both our train and test sets, we feed them to a machine learning algorithm to see if it can determine the source language for the novels in the test set.

(b) Algorithm

We take a text format novel from Project Gutenberg. We clean out of it Chapter headings, subheaders following chapters, beginning and end informational text, and anything else that isn't words that we can use

to compute. From here, we use the Stanford Parser's PCFG [4] to build best-guess parse trees (in Penn Tree format) for the sentences in this novel, and we use the Stanford Parser's tokenizer to build tokenized sentences and detect ends of sentences. Both the tree format and text format are used in our machine learning, but we could just use the tree if space efficiency was a problem. The text format is used to compute word unigrams, using Weka's toolkit [3] to do so, and the tree format is used to compute our parse features and homemade features as well. Both the parse and homemade feature computations were built from scratch with the ideas from [5] and [7], with some help from Stanford's Parser package to read the trees. Finally, all the features were combined together using weka instances and trained/tested upon using four different machine learning algorithms.

3 Building the dataset

From Project Gutenberg, we built a dataset of books written in different languages, and translated into English. We decided to add up to 6 languages and at least 6 novels from each language. We ended up with a total of 39 novels. For each novel, we used the Stanford Parser to tokenize sentences, and created 400 sentence chunks of text and parse trees. We chunked by sentences to ensure that we could slice them correctly.

(a) Pitfalls

Since we built it ourselves, our dataset is rather small. We therefore were extremely careful not to overfit on books particularities that are not specific to the language, such as topic, style, author, geographic places, time period. We believe the way we've constructed the dataset satisfies these criteria.

(b) Description of the dataset

When building the dataset, we emphasized using books written from 1700 to 1920, mostly getting books from the 19th century, to control the time period as best as we can. We made sure that we never used the same author in train and test, and we tried varying the topics between train and test as much as we could once the previous criteria were met. We tried to never use the same translator twice, but couldn't find enough translation data (when the name of the translator was available, we never used the same). When using Lexical Features, we only remembered the most common 1000 English words to control overfitting on topics as well. The dataset we used is further described on this table:

Language	Title	Author	Pub.Date
American	The Adventures of Huckleberry Finn	Mark Twain	1884
American	The Invisible Man	Orson Wells	1897
American	The Scarlet Letter	Nathaniel Hawthorne	1850
American	Moby Dick	Herman Melville	1851
American	Uncle Toms Cabin	Harriet Beecher Stowe	1852
American	The Awakening	Kate Chopin	1899
English	Great Expectations	Charles Dickens	1861
English	The Picture of Dorian Gray	Oscar Wilde	1891
English	Jude the Obscure	Thomas Hardy	1895
English	Treasure Island	R.L Stevenson	1883
English	Middlemarch	George Eliot (M. Evans)	1874
English	Frankenstein	Mary Wollstonecraft Shelley	1818
French	Around the world in 80 days	Jules Verne	1873
French	Father Goriot	Balzac	1835
French	Les Miserables	Victor Hugo	1862
French	Madame Bovary	Gustave Flaubert	1857
French	The Man in the Iron Mask	Alexandre Dumas	1637
French	Candide	Voltaire	1759
French	The Red and the Black	Stendahl	1830
German	The Devil's Elixir	E. T. A. Hoffmann	1815
German	Debit and Credit	Gustave Freytag	1855
German	Old Fritz and the New Era	Luise Mühlbach	1868
German	Venus in Furs	Leopold V. Sacher-Masoch	1870
German	The Sorrows of Young Werther	Johann Wolfgang von Goethe	1774
German	The Banished: A Swabian Historical Tale	Wilhelm Hauff	1839
Russian	Fathers and Children	Ivan Turgenev	1862
Russian	The Man who was afraid	Maxim Gorky	1899
Russian	War and Peace	Leo Tolstoy	1869
Russian	The Brothers Karamazov	Fyodor Dostoyevsky	1880
Russian	A hero of Our Time	M. Y. Lermontov	1840
Russian	The Death of the Gods	Dmitri Mérejkowski	1895
Spanish	Don Quixote	Miguel de Cervantes	1605
Spanish	Dona Perfecta	Benito Pérez Galdós	1876
Spanish	The Visions of Quevedo	Francisco de Quevedo	1626
Spanish	Tragic Sense of Life	Miguel de Unamuno	1912
Spanish	The Life of Lazarillo of Tormes	Lazarillo of Tormes	1554
Spanish	The Four Horsemen of the Apocalypse	Vicente Blasco Ibanez	1916
Spanish	Pepita Ximenez	Juan Valera	1874
Spanish	The Fourth Estate	Armando Palacio Valdés	1901

4 Features

(a) General Description

When computing the features, we cut the books in slices of n sentences. With $n = 1$, we get a lot of training examples, but finding out the original language from only 1 sentence seems unreasonable (which was proven by experiment). With the whole book as a sample, we didn't get enough training instances. We chose $n = 400$, as it provided good balance between number of train samples, and informativeness of the samples.

This number also happened to work well given our dataset; enough books had at least 4000 sentences to build a robust set. In training, we take 10 slices of each book, so that we have the same size of training for each language (most machine learning algorithms are sensitive to unbalanced datasets). In testing, we take all the slices, except the incomplete last one.

(b) Lexical Feature

Three different kind of lexical features were used:

1000 most common English words After overfitting on words like "Moscow" or "Madame" when using unigrams, we decided to only use the 1000 most common words in English to avoid overfitting on topics, local idioms, or geographic places.

POS unigram We only used POS unigrams, since we believed using bigram or trigram would just add redundancy with the Parse features.

Etymology For each word used, we computed the etymology of the word from parsing Wiktionary [6], and added all possible etymology as a feature. Rational being that if you're translating from a Latin language, you may be more likely to use a Latin word when choosing between synonyms. We only included the etymologies of nouns, verb, adjective and adverb, since we believed preposition, pronouns and others wouldn't be significant and would only drown the relevant data.

(c) Parse Features

Inspired by [7], we ran the Stanford parser on our slices, and computed the parse trees. By running a BFS on our parse trees, we then compute the generation rules and count them. We don't take into account the leaves of the trees or nodes containing just part-of-speech information and a child to a word, in order to avoid redundancy with lexical features (and possible topic overfitting).

(d) Homemade Features

We also computed most the features of [5], since this paper suggested strong power in their "Document-level" features. We did not grab all of these features as some of them (readability metrics, for example) we felt were either not important enough for their complex properties (neither readability metric made the top 50 features), or didn't work well with the Stanford Parser Part-of-Speech toekns (example: discourse markets were tough to detect given the POS). Table 1 shows which features we did compute: 14 of the 18 features from [5].

5 Classification task

(a) Features Extraction

Following [7], we were ready to use Information gain for feature extraction. However, it turned out that we didn't have that many features and could run all the algorithms on our dataset. Using Information Gain or Anova didn't help the results, so we finally removed this step.

Feature	Computation
simplecomplex	Simple:Complex Sentences
simpletotal	Simple:Total Sentences
complextotal	Complex:Total Sentences
avgsent	Total Words:Total Sentences
avgwordlength	Total Characters: Total Words
fverbratio	Finite Verbs:Total Words
numratio	Numerals:Total Words
prepratio	Prepositions:Total Words
conjratio	Conjunctions:Total Words
infoload	Open-Class Words:Total Words
nounratio	Nouns: Total Words
grammlex	Open-Class Words:Closed-Class Words
pronounratio	Pronouns:Total Words
typetoken	Word types:Total Words

Table 1: Contains homemade features we used in our parsing.

(b) Comparisons of different algorithm

We compared the results of a Logistic Regression, a Random Forest, a SVM with gaussian kernel and XGBoost with trees. Surprisingly enough, even if XGBoost is known to perform extremely well for classification tasks, the Logistic Regression always outperformed all the other classifiers, whatever number of features we picked.

6 Experiments

(a) Description of different experiments

We first decided to reproduce the experimental conditions of [5], to show that they're non representative of the task.

We then experimented with the different kind of features we computed, to pick the best possible mix.

(b) Results

Experiment	Accuracy	Best classifier
1000 Most common words	96.67 %	LR

Figure 1: Testing and training on the same books (different parts)

As expected, using the same books for training and testing gives excellent results even when using only the 1000 most common English words. Since we used a lot of books in common with [5], we're wondering how they got such poor results.

We get better results when using only the 1000 most common English words, as opposed to all the words (Unigrams). This was a very positive result, since it suggests that we did built our dataset in a robust way. Indeed, using unigrams completely caused overfitting on the topic. For example, we used Moby Dick in our training, and therefore *whale* is the third most relevant feature for American books. Contrary to intuition, POS unigram dramatically reduced the accuracy. It's interesting to notice that etymology features decrease the accuracy when considering only lexical features, but increased it when considering everything.

Experiment	Accuracy
Unigrams	61.80 %
1000 Most common words	62.92 %
1000 Most common words + etymology	61.80 %
1000 Most common words + POS	46.06 %
Parse features only	50.56 %
Homemade only	33.70 %
1000 most common + Parse	64.04 %
1000 most common + Homemade	62.92 %
Parse + Homemade	51.69 %
1000 most common + Parse + Homemade	64.04 %
1000 most common + Parse + Homemade + etymology	65.17
1000 most common + Parse + etymology	**65.17**

Figure 2: Lexical Features

When using Homemade only, we classify everything as Spanish or Russian, which is even worse than what the already poor results led to believe. As the results suggest, Homemade features are not informative, and they never appear in the top 100 most relevant features for a language. This surprised us because it was counter-intutive to what [5] claimed, and should be further investigated.

Adding the Parse Features and the etymology got us a 3.37% increase in accuracy, which correspond in our dataset to 3 new correctly classified slices out of 89, for a total of 58 correctly classified slices.

7 Future Work

While our classifier isn't as strong as it could be, We tried using it across five different closely related languages and still had a decently positive result of 65%, given our ground truth was 20%. We think that more work in the future should continue to look at the horizontal slices using parse trees, as that, along with 1000 most common word unigrams, seem to be the strongest predictors of language origin.

8 Conclusion

We have successfully built a scientifically robust experiment for Original Language Detection, which had never been done before. We built a consistent dataset, computed Lexical, Parse and Homemade features, chose the most suitable classification algorithm, and reached 65% accuracy on our testing set. However, our testing set is made of slices of 400 sentences. For a given book, if we compute those slices and take the majority vote for the whole book (the book is assigned the class that got the most slices), we reach a 100 % accuracy on our testing set! Obviously, the dataset is not big enough for this experiment to be conclusive, and we wish we had included more languages. Nevertheless, the results are promising!

References

[1] Marco Baroni and Silvia Bernardini. A new approach to the study of translationese: Machine-learning the difference between original and translated text. *Literary and Linguistic Computing*, 21(3):259–274, 2006.

[2] Project Gutenberg. Free ebooks by project gutenberg.

[3] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11:10–18, 2009.

[4] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association of Computational Linguistics*, pages 423–430, 2003.

[5] Gerard Lynch and Carl Vogel. Towards the automatic detection of the source language of a literary translation. 2012.

[6] Christian M. Meyer and Iryna Gurevych. Wiktionary: A new rival for expert-built lexicons? Exploring the possibilities of collaborative lexicography. In Sylviane Granger and Magali Paquot, editors, *Electronic Lexicography*, chapter 13, pages 259–291. Oxford: Oxford University Press, November 2012.

[7] Sze-Meng Jojo Wong and Mark Dras. Exploiting parse structures for native language identification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1600–1610. Association for Computational Linguistics, 2011.