The Classifiers

Christopher Akatsuka, Qi He

Introduction

After reading academic papers on author attribution, reviewing features we learned in class, and examining the methods adopted by PAN competition participants, we examined a list of relevant features:

- Token-based/Lexical: word and character n-grams, synonyms, syllable counts, letter counts, bag of words and distribution of words with various lengths.
- Vocabulary-based: Legomena, unique words, Richness ratios, Readability Index
- Sentence-based: lengths of sentences.
- Syntactic features: POS occurrences (e.g. nominative pronouns), function words, punctuation and symbols.

After printing and comparing the occurrences and distributions of each feature, we found that many features lack predictive power. For example, sentence-based features hardly apply as we are labeling only one paragraph instead of an entire article. NYT authors are also highly educated and have similar Richness ratios.

The synonym feature, however, stood out among all the choices. The approach to incorporate synonyms is unique and logically sound. More importantly, both of us want to experiment with something we did not learn in class. So we analyzed the feature in depth.

The theory behind the synonym feature is that the author's repeated choice among synonyms demonstrate a writing style. If an author often uses the the word "impervious" instead of its synonyms ("impenetrable", "resistant", "immune" and etc), "impervious" then becomes representative of his/her writing style (see our Analysis for more interesting findings). The more synonym a word has the more alternatives the author could have chosen. So a word is more representative if 1) it is repeated more, 2) it has more synonyms, and 3) the synonyms frequently occur in the larger corpus (common words that other authors use). Following this logic, we used Wordnet to get synonyms and implemented the feature.

After experimenting with other features (see Discussion and Analysis for more), we went back to the fundamentals and implemented word and character n-grams since our results for the synonym feature ended up being lackluster. We knew from class that word n-grams were good for labelling unlabeled text. Character n-gram is also particularly powerful as it captures word stems, syntactic characteristics such as gaps between words and punctuations, and word frequency fluctuations[1]. A paper from the University of Houston suggested character n-grams are uncannily good at determining authorship both in single-domain and cross-domain settings, so we were particularly interested in seeing its effectiveness in conjunction with our other features.[2] Our cross-validation tests and five test results ended up validating our predictions. Word and character n-grams were a great way to determine authorship.

Method

Since our original goal was to experiment with new techniques, we first built the synonym feature without relying on 3rd party ML tools. We relied on NLTK to tokenize strings, and we used Wordnet to generate synonyms and lemmas. We also built our own stop list, which includes the file used in previous homeworks, Wordnet stopwords, and the top 2500 first and

---

[1] N-gram-based author profiles for authorship attribution.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.7388

[2] Not All Character N-Grams Are Created Equal: A Study In Authorship Attribution.
http://www.aclweb.org/anthology/N15-1010

last names from the census. Note we did not want to exclude all named entities via taggers, as some scientific names might carry significance in our predictions, especially because Kolata's articles are mostly scientific in nature (see our later Analysis).

As a result, each test paragraph will have a match score with Kolata training set and another with non-Kolata set. Each unique word that is in the test paragraph as well as the Kolata training set will get a match score, and the score equals to its occurrence, normalized by its global occurrences, times total global occurrences of its synonyms. Of course, words will get a non-Kolata match score as well for comparison.

On the other hand, since we were relatively familiar with word n-grams and character n-grams (since it was a similar setup to word n-grams), we tried to avoid 'reinventing the wheel' and used the sklearn library.[3] Through this, we were able to create word and char vectors for n-grams between ranges we could specify, and union (literally concatenate the two vectors together to make a single, massive vector) these vectors to form a model, which we applied a linear kernel to, to create a pipeline (classifier). Then, we could simply train the classifier using our 'train' data and predict whether each new paragraph from the 'test' data was written by Kolata or not. Settings such as a cap on the feature set, using idf with smoothing (used on the training data to weight occurrence counts), case-sensitivity, and stop words were available and used as explained in our Experiments section to varying degrees of success. Initially, we also implemented a method to read the output from our word and character n-gram and integrate synonym feature results, but this ended up decreasing our accuracy, as our experiments showed.

Final System

We, The Classifiers, utilized the feature sets of the union of a word 3-gram vector space with a character 4-5-gram vector space (created character n-grams for values 4 and 5) with no feature set maximum, idf with smoothing, no case sensitivity (we lowercased everything -- we also lowercased everything in our assignments, so we wanted to be consistent), and applied the stop words mentioned in the Method section to the word n-grams.

Experiments

Our synonym feature experienced relatively low accuracy, high false positive and low false negative when tested on the training set. Our first submission, which only has the results from the synonym feature, only showed around 70% accuracy.

On the other hand, our 3-gram word, 3-gram character features with a 2000 size cap on our feature set, no stop words, idf with smoothing, and no case sensitivity resulted in a 90.11% accuracy when submitted. With this promising result, we then decided to focus our future submissions and development around word and character n-grams.

Next, we tried out a 3-gram word, 3-gram character feature with stop words added to the n-gram words, no size cap on feature set, and incorporated the synonym feature and incorporated the synonym predictions (everything else from last experiment remained same). To incorporate synonym feature, we checked for when the n-gram model said 1 and synonym model said 0 and took 0. When running our cross-validation tests, we broke down our errors and noted that the synonym feature produced few false negatives and the n-grams generated many false negatives, so we theorized this incorporation would cut down on our n-gram's false negative count. This yielded accuracy of 93.09%, a significant improvement.

[3] http://scikit-learn.org/stable/

Then, we increased the n-gram count to 5 for characters (word gram was still at 3) and maintained all of the settings in the previous test. This yielded accuracy of 94.57%

We tried again, but without incorporating the results from the synonym feature, to see how much this feature was actually contributing. We had 95.39% accuracy.

Finally, we wanted to shoot for the #1 spot on the leaderboards, so we arbitrarily tried char grams for 4 to 5 characters with all other settings remaining the same. This unfortunately only got us to 95.46%, which was a mere .15% away from the #1 spot as of submission.

Discussion and Analysis

We were not extremely surprised by our synonym feature results. Currently our model includes all words whose lemmas are synonym of the given token. Our results would be greatly improved if we determine the sense of each word and only account for synonyms under the correct sense. However, this additional process would drastically slow down the algorithm. And each submission could potentially take more than a day to run (currently it takes hours).

Compared to other features we discussed in introduction, synonym feature does moderately well with only paragraphs to classify (instead of articles). In fact, we found some very surprising results. Of course, one would expect more adv and adj that are representative of her writing style in our findings. However, we found that these words might reflect on the actual topic of the paragraph as well. Take “impervious:” it might seem like a fancy word at first, but in fact Kolata exclusively uses it to describe immune systems and drugs. In addition, we even found nouns like the word “statistician” (instead of “researcher” and etc). So we would argue that using similar topic words might not have found enough scientific articles with similar topics to “confuse” the students. This finding also somewhat justifies our decision to build a custom stop list instead of excluding all named entities.

More importantly, the feature does a good job limiting false negatives (41 out of 1500 in hold out). We could further improve the false negatives by imposing a “delta” in matching, effectively requiring the difference between Kolata and non-Kolata to be large enough to mark as a non-Kolata. So we decided to find features that complements the synonym approach. In addition, because the training file was generated using topic words, we want to find features that are not “fooled” by the similar topic words.

We tried to use POS tagged unique words to improve the results. However, because the non-Kolata set does not have a uniform writing style and many paragraphs do not contain these unique words, this feature was not particularly useful in eliminating false positives. More importantly, in order to avoid topic words, we would have to spend more time analyzing all the unique words to find the best indicators (e.g. adverbs that occur frequently in non-kolata set). This approach turned out to be very time consuming.

So we went back to implementing the features we are most familiar with. The combination of character and word n-grams ended up working out very well. This high accuracy for determining authorship was consistent with the University of Houston article we cited in the Introduction. Along the way, making minor changes like removing the cap on the size of our feature set and adding stop words to increase accuracy also made sense. And, when taking a closer look at the n-gram character feature (since the n-gram word feature was already covered in class, and thus, less interesting) the n-gram character feature included whitespace, punctuation, and all character groupings. From little things like starting a quote with lowercase letters, like “no” as opposed to “No” and using phrases such as the “‘60’s” as opposed to 1960s, differentiation from other authors was made clearer. There was 315,865 entries in the n-gram character feature.

If we had more attempts, we would have liked to see how well we could have done if we used specific parts of sentences for the char n-grams, as the University of Houston article found effective. Regardless, it is clear that n-grams are very effective ways to determine authorship.