Classification of Book Titles

Gongmyung Lee

December 8, 2010

1. Introduction

The aim of this project is to apply machine learning classification to scraped data from the Stanford bookstore and Amazon to find incorrectly matched results.

The data comes from a website that I have been working on (whichtexts.com) that enables students to find out which textbooks they need for their courses - all they need to do is enter their classes and the website will automatically create a list of required books. Additionally, the website also gives a price comparison for each book on the major bookseller websites (Amazon, Half.com, Alirbis, etc). This involves a lot of data scraping - an automated script scrapes the Stanford bookstore website to get the required books for each course, then looks up the price of each book on Amazon, Half.com, etc and stores the price in the database.

Unfortunately, the Stanford bookstore site does not store the ISBN of each book, so the Amazon listing must be found using the title of the book (once the Amazon listing is found, the ISBN can be used to search the book on other websites). This would not be a problem except that the Stanford bookstore site sometimes arbitrarily changes a portion of the book title - examples are: shortening “principles” to “prin”, randomly putting the edition of the book in parentheses after the title, etc. This results in books having incorrect Amazon links or no Amazon link at all (when no search result is found). Additionally, some of the required materials can only be found at the Stanford bookstore, like course readers or PRS clickers, and thus does not have an Amazon link. This presents a classification problem - the data needs to be classified into 1) books with a correct Amazon listing, 2) books with an incorrect Amazon listing, and 3) books with no Amazon listing (like course readers and PRS clickers). The goal of my project will be to create a supervised learning algorithm that will correctly classify future book-link pairs based on the book and link data has been already scraped.

2. Designing the solution

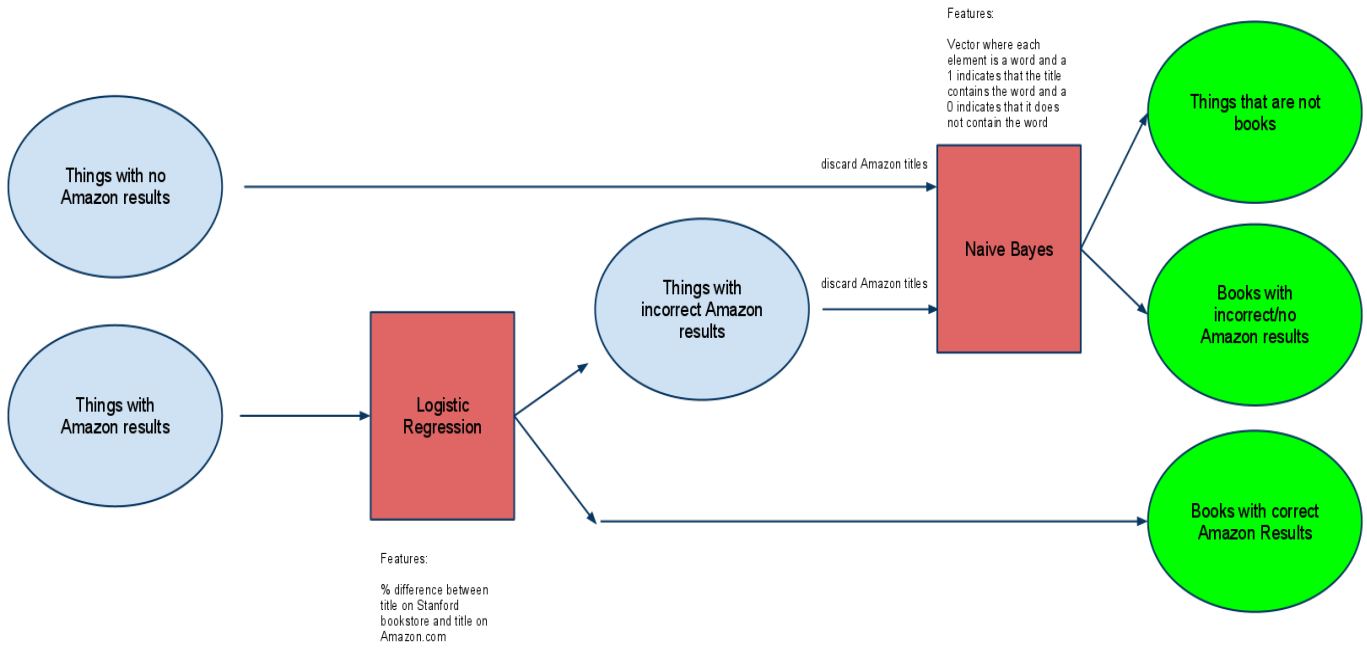
At first, it seemed like this project would be a simple one-step classification problem. Upon examination of the different types of inputs, however, it turned out that using multiple machine learning algorithms in sequence provided much more accurate classifications

The inputs could be separated into two classes - books with an Amazon result and books

with no Amazon results. I differentiated between these two classes because the books with an Amazon result had both the title from the Stanford Bookstore and the title form Amazon available, whereas the books with no Amazon results only had the title from the Stanford bookstore. Thus the books with Amazon results had a different set of input features then the books with no Amazon results. I decided to do a multi-stage classification process described as follows:

1) Take all the books with a corresponding amazon listing as inputs. Classify the books into having a correct Amazon listing or having an incorrect Amazon listing. (example: if a book's title on the Stanford Bookstore is "Prin. of Programming in Scheme" and the listing we find on Amazon is "Intro to Scheme Creation (Prince edition)" then we would classify it as incorrect)

2) Take all the books identified as having an incorrect Amazon listing from part 1, and all the books with no Amazon listing. These will be the inputs for step 2. Use input features based on the title of the books from the Stanford Bookstore to classify these books into 2 categories: things that are valid books and things that are not books. (example: "Intro to Programming in Scheme" is a valid book while "PRS clicker" is not)



4. Implementing the solution

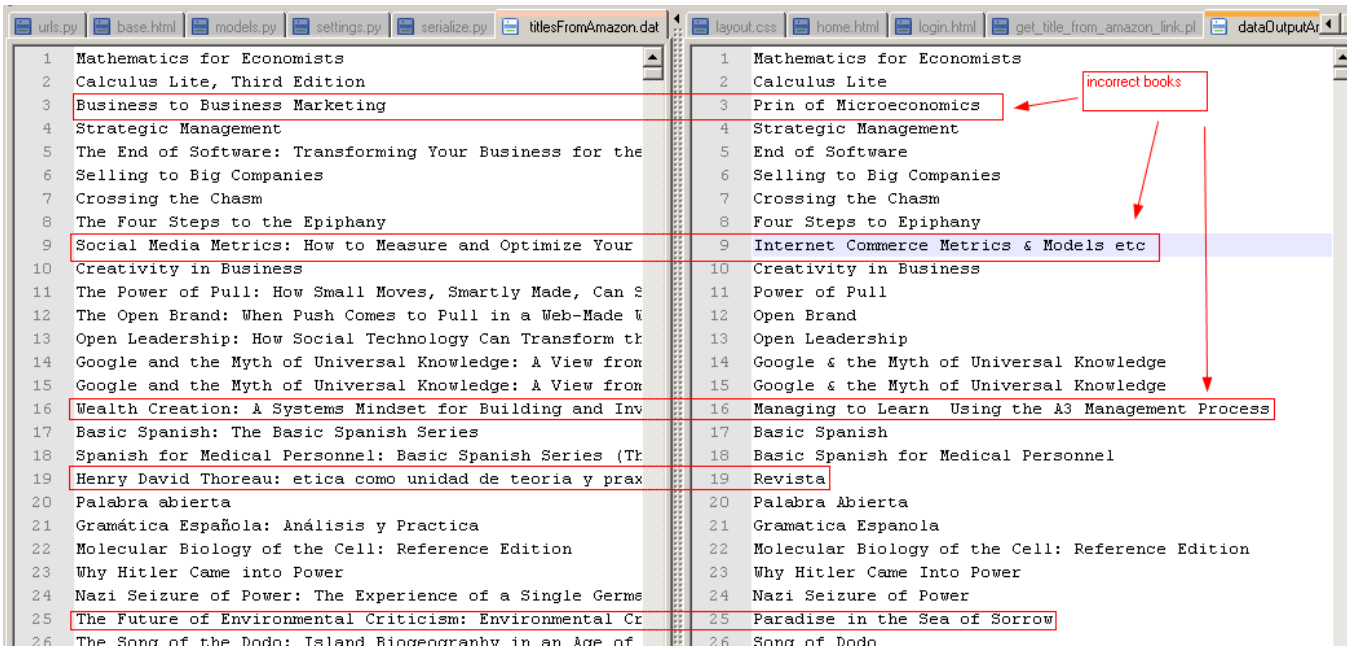
Training Data:
The first step was to compile the training data that was needed for step 1. A perl script was used to compile the necessary information from the database, as well as do some basic data-scraping to get the book titles on Amazon (the links to Amazon were stored in the database but not the titles).

We had 2983 books in the database, and 2098 of those books had an Amazon listing. A randomly selected sample of 500 books was taken from the latter as the training data for the first classifying step. The 500 books were manually labled as having correct Amazon listings and incorrect Amazon listings. After going through those selected books, 73 incorrect books were found. For the second classifier's training data, I took another random sample of 500 books, where 400 of the books were books with no listing and 100 of the books were books with an incorrect Amazon result. (the ratio of books with no Amazon result to books with an incorrect Amazon result was about 4:1).

Step 1:

In general, It was very clear when a book had an incorrect Amazon result - the titles would often only share a word or two and otherwise be completely different. Thus I decided that just the feature of the % of similar words between the two titles would be sufficient by itself to correctly classify if a book had a correct Amazon listing or not.with Amazon links.

Amazon Titles | Stanford Bookstore Titles



More formally, the feature of how "closely" the titles match was calculated for each title by counting how many words in the bookstore title is contained within the amazon title, then dividing by the total number of words in the title to get a word matching percentage. The other way was calculated as well (how many words in the amazon title are contained within the bookstore title), and the bigger percentage was taken. This accounts for cases where the bookstore title is a subset of the amazon title vs the amazon title being a subset of the bookstore title.

Step 2:

For step two of the algorithm, I used a vector of all the words in the titles of books in our database as the inputs to the algorithm. I used the method outlined in class used for email spam classification - each title would be a vector where each component of the vector represented a word, and a 1 would mean that word was in the title and a 0 would mean that word was not in the title. I then used a naive bayes classifier to classify the books into two possible categories - books and non-books (I included anything that could only be found at the Stanford bookstore, such as course readers, as non-books).

I expected this to work reasonably well because most things that are not valid textbooks have a specific subset of words that they share (course readers always have the words “course” and “reader”, PRS transmitters always have the word, “PRS”, art supplies typically have the words “Canvas”, “brush”, etc

5. Results:

Step 1:
It turned out that linear regression worked extremely well with the %words matching feature for step 1. I used a 5-fold cross validation, where each time I trained the classifier on 400 examples (of which 50 were incorrect examples), and tested it on the remaining 100. There was a 99.8% accuracy rate - out of the 5 times the algorithm ran (on 100 test examples each time), there was only 1 error for an error percentage of 0.2%.

Step 2:
Again, the naive bayes classifier worked very well. It did not work with the accuracy rate of the first classifier, but that was to be expected because there were many examples that were outliers (for ex: “Inequality Reader” - this is a course reader that does not contain the word “course” and “First Course in Finite Element Method” - this is a textbook that contains the word “course”). Again I did a 5-fold cross validation with 400 training examples and 100 test examples. This time, the classifier had a 89.4% accuracy - out of the 5 times the algorithm ran on 100 test examples each time, there was 53 errors for an error percentage of 10.6%.

6. Conclusions:

Overall, both the linear regression and naive bayes classifier worked well with the selected features for the problem of classifying book title matching. Unfortunately, the overlap of features for multiple examples during the second stage of classification made that step much less accurate then step 1. It was, however, still good enough to use to identify books that needed to be manually updated on the site. Since we could use the first classifier to filter all the books that had incorrect links, that reduced the number of books we needed to look at to about 1/8 of the total (plus 900 for the books without amazon links), and then the second algorithm reduced that number by 90%. In the end, we were left with a list of 132 books that we needed to look at and fix manually, which was much more manageable the the original number of 3000.