Christopher Doucette

Stephen Demuele

Machine Learning 4/29/16

Machine Learning Final Project Report

The goal of our natural language parser is to scan through text and identify a movie review as being positive or negative. The practical purpose of our code could potentially save the need to read several reviews to understand the general idea of critics' opinions on a film. When attempting to sum up the total reviews, one could just find a large number of different reviews, and run this code on them to determine what fraction of them were positive or negative without the task of manually reading through each review.

Previous studies into the machine learning approach of natural language analysis include many research projects that are not exactly on the same topic as our own. There are many studies taken, such as the detection of text genre of a given text (Karlgren and Cutting, 1994; Kessler et al., 1997; Finn et al., 2002). This is similar to the classification of a movie review for sentiment analysis, but not entirely on point with the specific approaches we would need to take. The ideas for identifying a genre of a text were intriguing in how they did not focus on individual word choice, but mainly the style of wording sentences. The style of sentences used could very likely prove useful to our identification, however it would be difficult to ascertain the stylistic differences between movie review writers. This is doubly difficult with the added complication that these are likely anonymous writers that have varying levels of writing ability, where one would be able to convey in their style their sentiment without easy, powerful words, but another could come across as middle-of-the-field, picking no sides until a single sentence at the end determines their ultimate overall rating of a film.

Another, much more related research topic we looked into, proved much more useful to the

direction we should move with our sentiment prediction project. A research project by Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan from Cornell University was completed in 2002 on the same topic we have been tasked with (Bo Pang, "Thumbs Up? …"). Our own approach strives not to copy their approach, while keeping in mind they chose to assemble a model of many words weighed using the Naïve Bayes algorithm. We believe that the results they obtained from the Naïve Bayes method was astoundingly successful, however just simply following an example by previous students without testing another method is not research. In light of this, we chose to forego the Naïve Bayes method and instead use our bag of words model in the end with the weighing based on the Gaussian-like distribution of word occurrence based on the ratio between positive sentiment occurrence and negative sentiment occurrence.

Our approach to our movie review sentiment prediction project started with our early pseudo-coding and brainstorming. The initial task the two of us had was to formulate a method of identifying patterns in reviews that could lead us to determining an estimate on a sentiment. The most efficient idea seemed to be a bag of words model with weights applied where needed. Our pseudo codes for parsing text and checking for key words was then successfully finished quite promptly.

The second step was to determine what key words would be tested for. Our first run for key word searching used words for which we expected a common occurrence and easy sentiment determination. Our scans initially showed that the expectedly common words were detected in reviews of both positive and negative opinions. When considering the distribution of word occurrences as if it were a Gaussian distribution, we were able to properly weigh words for a noticeable increase in our sentiment prediction based on the training data.

A much more involved task was the widening of our search data. There was a very apparent need to increase the number of words that we were searching for. We both manually read several reviews, searching for common occurrences in multiple reviews. One particular plan that was

formulated was to create a class that identifies the most common words appearing in each file of the training data. However, a challenge that appeared with this approach was that we would need to omit extremely common words as well as words that bear no weight toward one sentiment or the other. The ultimate decision not to use this method was because of the time requirement it would have taken to make a single-use program that would find words we want. It proved to be more time efficient to read reviews manually. Often times when reading the reviews, we would exchange them without telling each other the sentiment of the given review. This way we were both attempting to use our own abilities to determine whether a review was positive or negative, while at the same time searching for those words that could hint at the sentiment for us.

The data set were the 12500 positive, 12500 negative, and 11000 test files provided specifically for this competition. Each file was a text file that contained a human written movie review that was either negative (0-4) or positive (7-10). These files could range in length from no more than a few words to entire plot summaries of the epic. All were single line outputs and there were restriction on no more than four reviews for any given movie, (siting that the reviews might be to similar for proper testing). The first task was to take 10 positive and 10 negative files solely for code functionality test. For ease of access these files were modified by hand in order to make them more readable to human users. This included putting in paragraphs and removing remnant HTML codes that were a side effect of it being ripped from a web page. These files were used to test in the program could read in files would analyze words. Trial and error of a few types of code iteration in c++ was soon met with success. After completion of this phase we moved onto the next part of the data set which was to give it access to the full 25000 positive and negative files in order to extract better information. Then we ran the final test file through it.

The files were parsed through program one at a time. Each file in the given directory was analyzed word by word. The main file would grab a file. The file would be passed along and a line

would be grabbed on at a time. Each line when the broken apart into its individual words. These words were first make into a all lowercase so that further parts of the string set up could be read by the stream with a few duplicates as possible (Terrible vs terrible). Following that we cleaned up all unneeded punctuation (. ? !), remnant HTML code (
), and other oddities. The word was then grabbed and passed along into a list of words in order to be counted. This count was recorded to be either in the positive or negative reflecting the review it came from. Given that a point was attached to that word. With this information passed along the data was analyzed. The words that were deemed useful (common occurrences in the files) where then given weights that properly reflected the number of times each word popped up. For example if a word popped up twice as often in The negative files than the positive it would be given a weight of negative two, if a number appeared five times more often in the positive files than the negative files it was given a value of plus five. We capped the value off at plus or minus eight for sake of over-weighing words to impossible proportions. Following this we implement the Not functionality. If a negating word (not doesn't didn't. Etc.) were to appear a switch would occur and the computer would reverse the value of the words value. For example if the word had previously been a plus two value it would be a minus two value. Each word of any given review was read in and analyzed. The values added (or subtracted in case of negative words) were added into a variable that would keep track on the file as a whole. This number was then passed into a function that would report out the files name and if it was negative (0) or positive (1) into a csv file.

With all of this in place the project could begin. Using Codeblocks to aid in the creation the program that was written in c++. The directory locations were copied into the launch line. I used the positive and negative test data to see how accurate the testing is. With this information we did fine tuning to various weights by hand. We ran them until the situation was desirable enough to run though the test. After we ran the final test directory through and checked the results. Although we did not use a purely automatons programming. It did require us to manually manipulate various weights for our experiment, but the machine was the primary decider for this. The group lacked the time and skill sets

to make it completely an AI unit. The code was written almost entirely using the basics of c++ and our heads. The only borrowed functionality was for reading in an entire directory found inside of the main function. The rest of the code was written by the team.

The results were not as well as we were hoping but a passable. With about 72% accuracy on the final test it was acceptable. The positive learning was about 82% and the negative was roughly 65%. Although we original set out for a higher value, upon closer inspection of the actual data it became clear that we set our bar too height. The largest issues with the project was that humans were involved with data. Starting with the rating system of one to ten stars. These numbers are an arbitrary unit that has no base in reality. There is no set scale universally about how a movie is a four. And this led to some very irritating situations. A four to some people is just a touch below average not a bad review just not good. One such case used the lines, “Plot is not worth discussion”, “Good movie to watch after dinner”, “The whole is watchable”, “better than most TV shows”, and the rest of the lines do not hint at anything wrong with the movie. This leads to the second issue with this project, the English language. The English language is something so bizarre complex and intricate that we have yet to be able to create a computer that can hold a fluid conversation with a human. This exercise helps show this. There was a loss of information by having people type in text. When you here a person speak they inflect words and convey emotions. When your only read text without such emotions a vector of information is lost. Often it is regained by looking at what the review was scored, but seeing as that would make the experiment moot it can't occur. So all things considered a 72 is good in my mind.

In short the project was successful enough. Future additions might look into different words sets, new ways to weight them, and possibly looking into more subtle methods of determining what the persons feelings toward the film may be. The largest lesson to learn is that when trying to analyze text, try to make sure the basis of the text is set to something concrete. Trying to do this caused problems mostly from the text files. But that is just a thought for another time.

The groups work was split rather evenly between the two members. The project's live code was

handed off between the two of us. Each of us ran tests and experiments to create the final formula. Even this paper was written as a joint effort. If roles must be assigned though. It could be noted that one member was more focused on the coding itself while the other was more focused on the algorithms, but to say such a thing would be splitting hairs. In short the total project was a summation of a joint group effort.

Brett Kessler, Geoffrey Nunberg, and Hinrich Sch¨utze. 1997. Automatic detection of text genre. In Proc. of the 35th ACL/8th EACL, pages 32–38.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. PDF “Thumbs up? Sentiment classification using machine learning techniques”