Angelo Matni

Criticism in Review: Language Processing

Task

The work I set out to achieve was to predict whether a product critique is considered helpful by those reading it. Determining the usefulness or relevance of a discussion without manually sifting through the conversation is extremely valid, especially for large-scale blogs and forums. The first step in doing this is to be able to pick out strings of posts that stand out, containing language commonly associated with worthy and thoughtful remarks. Since blogs don't contain huge amounts of entries that are easy to pick out, nor standardized rating systems to compare my results with, I chose an Amazon review dataset.

Dataset

I was given a dataset from Amazon containing millions of reviews on hundreds and thousands of products. I chose to work with a few categories of product reviews, namely video games and other entertainment, totaling more than five million reviews. I then chose reviews where the number of helpful/unhelpful votes were higher than 10, leaving more than a million reviews to work with. I did this to be able to compare my results with reviews that had been deemed helpful or not by more than a few people to avoid outliers.

For most methods I tested, I used roughly 200000 training examples (an example being a single review of a product) and 200000 testing examples. The reason I didn't use more of my dataset was due to limitations in hardware I had available and amount of time I had to run the several methods I approached this problem with.

Methods, Features, and Results

I began by creating word embeddings for the vocabulary I was working with, scanning all 5 million reviews using a skip gram model and a neural network. This alone proved challenging, but was eventually successful. I used code offered by Tensor Flow, along with a few other examples, to write up the neural network and to plot the dimensionally reduced embeddings. After testing with several skip windows and several word embedding dimensions for the skip-gram model, I settled on 2 and 64. Figure 1 below shows the dimensionally reduced embeddings. Notice that there are several similarities in words based on positions, something encouraging.

Strong examples of the correlation these word embeddings hold include: the multitude of numbers listed together on the far right; the proximity of many adjectives in the upper middle region and of similar adjectives, such as amazing and excellent or difficult and hard.

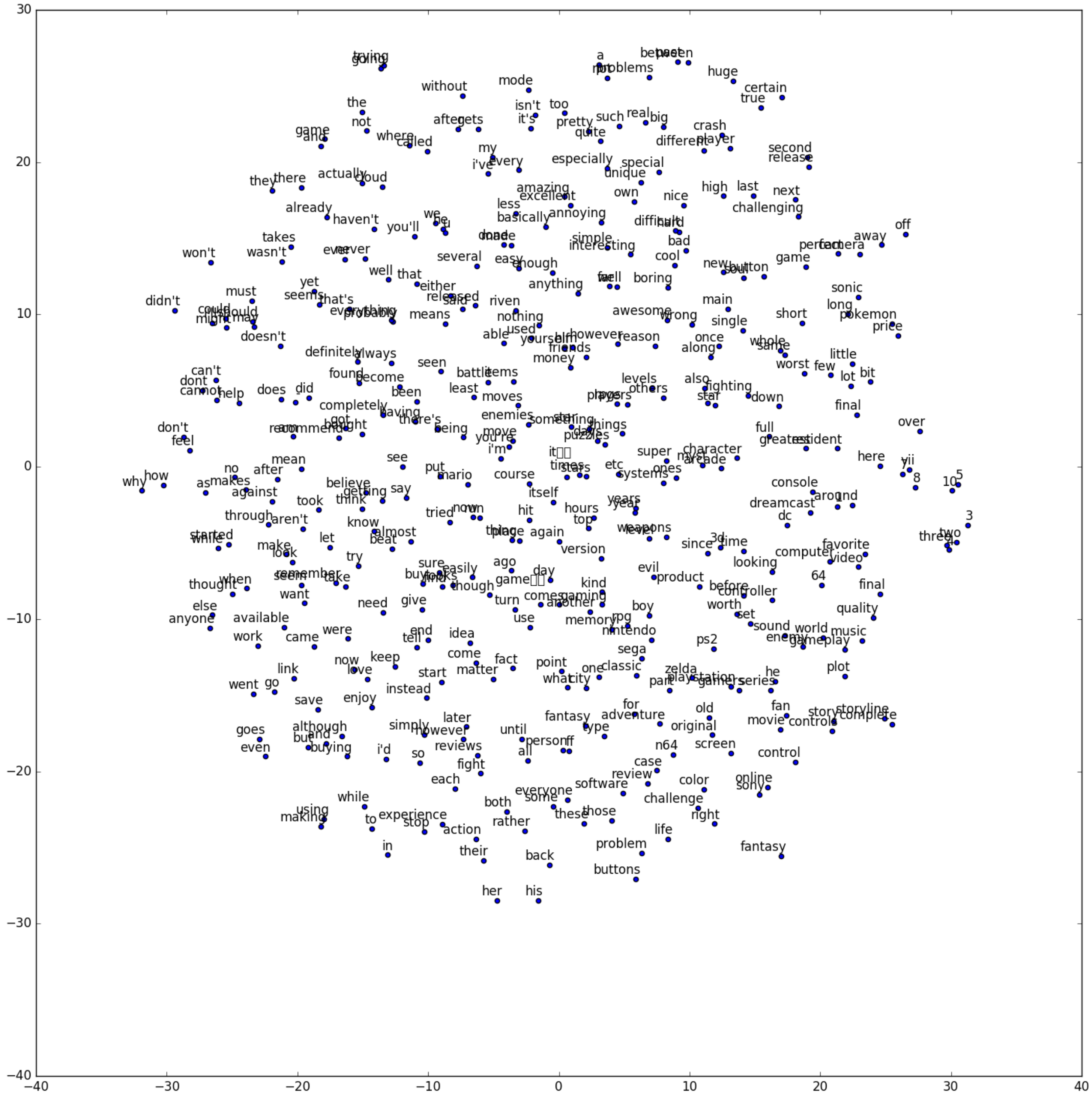

Figure 1: Dimensionally reduced word embeddings using a skip-gram model

Once embedding was taken care of, I moved to feature design. This was where I spent the bulk of my time. Each approach I attempted is documented in the python code I used for preprocessing/feature design and creation of the .arff files used in Weka to train several models.

At first I clustered the word embeddings using k means, comparing the effectiveness of different numbers of cluster and repetitions of the k means algorithm that yielded reasonable groupings. I did this by listing sets of words from each cluster after the algorithm ran and seeing whether they had any reason to belong together. I found that for the huge variety of words I had, no number of clusters could capture fairly logical groupings without breaking up the vocabulary into too small chunks. Unfortunately, I couldn't try hierarchical clustering on such a large vocabulary due to computational limits. Just in case my eyes were deceiving me, I went ahead with the optimal 30 clusters, defining features as the percentage of words occurring in each cluster for a review. This achieved testing accuracies, using decision tree and kNN, of 52% and 47%, respectively. By accuracy, I mean how many instances were classified correctly as helpful or not helpful (whether a majority of Amazon users up voted the review). I can only assume the accuracy for kNN was lower because of the prevalence of noise in my system, namely the inconsistencies in clusters.

I then attempted to define my features as the sum of all word embeddings for every word in a review. Hence, this created 128 attributes, one for each dimension of the final review embedding. This was a shot in the dark based on the assumption that the aggregate values would somehow generalize a review without losing too much information. This method only yielded stronger accuracies when I ignored the hundred most common words in my vocabulary, achieving testing accuracies, using decision tree and kNN, of 51% and 49%, respectively.

After this, I decided to reduce the number of dimensions of the word embeddings to see if that would reduce computation time or avoid over-fitting, since describing semi-useless features can cause both upon classification. To do this, I used linear regression on the word embeddings themselves, classifying each word embedding by the cluster it fell into. The clusters I used in this linear regression method were of much finer grain, totaling 100 instead of just 30. With this method, once again ignoring the hundred most common words in my vocabulary, I achieved testing accuracies using decision tree and kNN, of 67% and 58%, respectively. Training a neural network in Weka yielded slightly higher testing accuracy, at 71%.

Any other methods of concatenating word embeddings and then dimensionally reducing the entire vector, or summing word embeddings weighted by how common the word is, yielded similar results. For the sake of time, I will not list these approaches and rather leave their descriptions to comments within my code.

Future Work

In the future, I will more systematically devise features that can effectively separate pairings of words and not just words on their own in order to better classify entire reviews. This begins by addressing the balance between complexity of word embeddings and ability to combine them to accurately represent sentences. Once this is done, more meaning can be ascribed to any set of attributes defined so that more can be drawn from a model, other than the end-goal accuracy the model achieves. Also, a feature design that yields several highly irrelevant attributes should be questioned far sooner than I did whilst working on this project.

Web Page Link

http://criticisminreview.weebly.com/