Raymond Duncan

CSCI5622

Feature Engineering Writeup

I ended the feature engineering competition with about a 4 percent increase from the baseline. That's not a stellar increase in accuracy, but it's not horrible either. In order to test my added features, I simply split the training data into three portions: Training (60%), Validation(20%), and Holdout (20%). In hindsight, this may not have been the best decision considering the training set was only around 14800 items, but I also didn't think it would be worth it to use cross-validation when I didn't yet have that many additional features. Here's a brief explanation of my choice of features and how I came about them.

The first features I added were support for bigrams and trigrams. I tested the impact of these and found it gave me about a 3 percent increase from the baseline. I did this to add some context from each example thinking that it would pick up on certain keyword sets like <Name>,<Verb,Past Tense>,<Name>. This would not work unless the testing example came from the same show. That brings me to my next addition.

I attempted to use nltk to add pos-tags to all the words in the examples. This proved to be a little harder than expected since I didn't completely figure out how to use the CountVectorizer to add distinguishable features (i.e. not in the same string), so I decided to try a couple different orderings of the data using pos-tags only, and words paired with the pos-tags in parens. With both of these accompanying the base example text I gained an additional 2 percent on my offline testing.

Unfortunately these are the only tags I added, but given the ability, I likely would have used nltk's stemming capabilities to get more resolution from words that have the same infinitive, but different use cases (i.e. die, died, dead).