Isai Mercado Oliveros
April 11, 2016

Funny Classifier

When I presented my slides in class, I described a naive based funny classifier that would classify large documents. Large documents contain enough words for a "bag of words" Algorithm that finds interesting word patterns. However, my original idea was to see if small sentences were funny such as a small joke, so I kept looking and found the Stanford NLP sentiment analyzer. The way it classifies sentiments has the potential to classify funny sentences, but sadly it is not implemented. I will describe what I learned from its implementation and what I think would make a good funny classifier.

1 - Exploratory Data Analysis

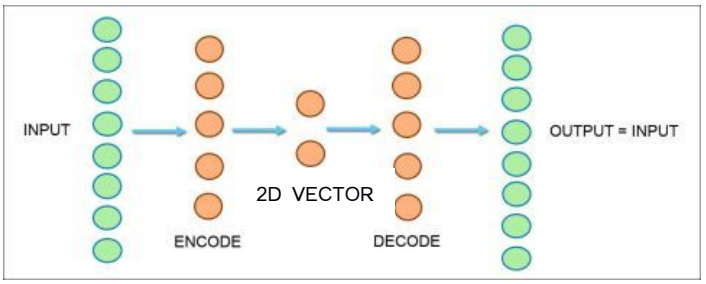
First of all, I spent a lot of time looking at jokes, and I found out that they are very hard to classify by a bag of words algorithm because they do not contain enough words, and a lot of the funny elements are implicit. Jokes may contain funny words, verbs, or adjectives, but this is not necessary. Many jokes are made out of normal words that are funny because of how the sentence is structured, such as rhymes, or because they remind the reader about something funny in the culture. Therefore, I learned that in order to classify funny sentences, we need a form to represent context, words' meanings-sentiments, and sentence structure.

2 - Description of Technical Approach

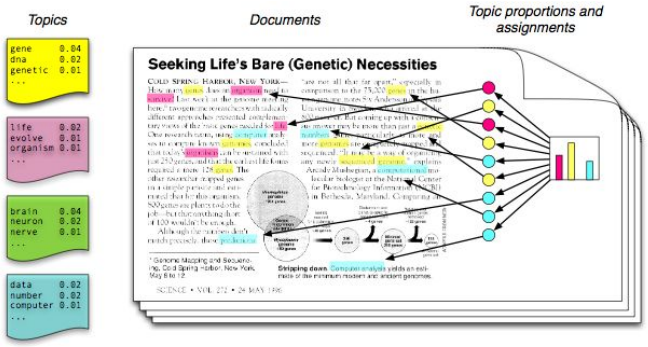
After I analyzed jokes, I started reading about algorithms that would take into consideration context, words' sentiments and sentence structure. I read about recurrent neural networks, specially LSTM, thinking that they would learn sentence structure, but they learn word sequences not syntactic structures. Therefore, I kept reading and found the Stanford NLP Library which implements sentiment analysis by a recursive neural network. Sadly they do not implement a funny classifier, but the way they classify positive and negative sentiments can be used as a based to classify funny sentences. First I will explain how the Stanford NLP Library works and secondly, I will explain how think that this technique can be used to make a funny classifier.

A) Explaining Stanford NLP Approach

First, The Stanford NLP Library represents words as vectors. They do this by running all words into a neural network auto encoder. When the auto encoder has learned to produce the same input word as its output, it has actually learned how to map words to vectors of arbitrary length.



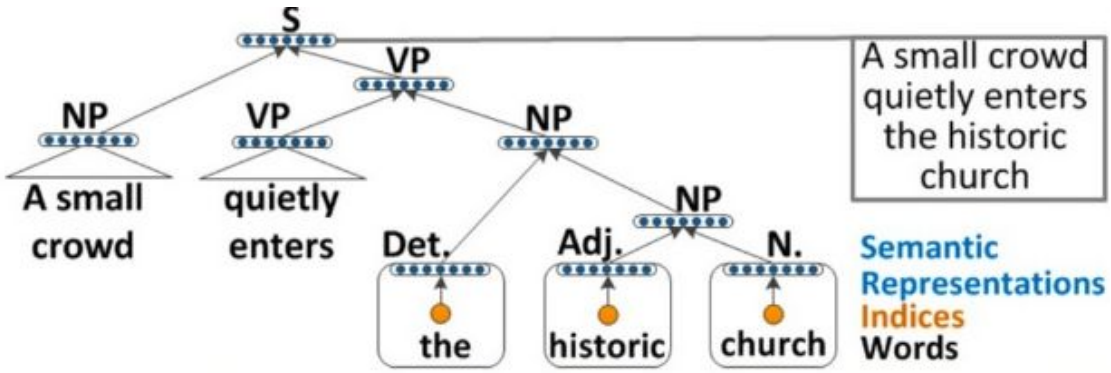
Another way to map words to vectors is by running clustering algorithms like LDA, and make a vector of topics-distribution per word from the words-distribution per topic vectors.



- Each **topic** is a distribution over words
- Each **document** is a mixture of corpus-wide topics
- Each **word** is drawn from one of those topics

Making a vector of topics distribution per word has the advantage that now the vector has an actual sentiment/topic meaning attached to the dimension.

Once a word can be represented mathematically by a vector, it can be used as an input to a recursive neural network. Recursive neural networks are called recursive because they form a tree and every node contains a neuron network whose weights are shared by every single node in the tree. Because of the shared weights, it works as if the same neural networks was being called recursively. Error propagation is very similar to vanilla back propagation except for minor changes to make it work in a tree structure. Recursive neural networks are used to map sentences to the same vector space that is currently describing the words. Therefore, they receive as input a sentence and output a vector.



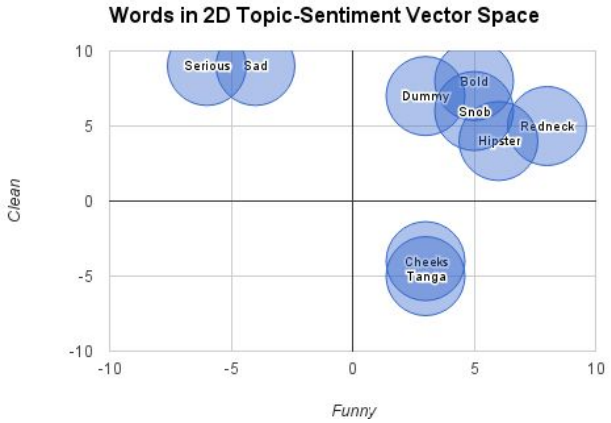
Recursive neural networks can map a sentence to the vector space by following its syntactic structure. For example, sentences like “the virus is destroying antibodies” vs “the antibodies are destroying viruses” will form different trees so the resulting vector will be different.

As mentioned before the Stanford NLP Library cannot be used a funny classifier because it defines its word/sentence vector as very-negative, negative, neural, positive, and very-positive.

B) Explaining my version of funny classifier

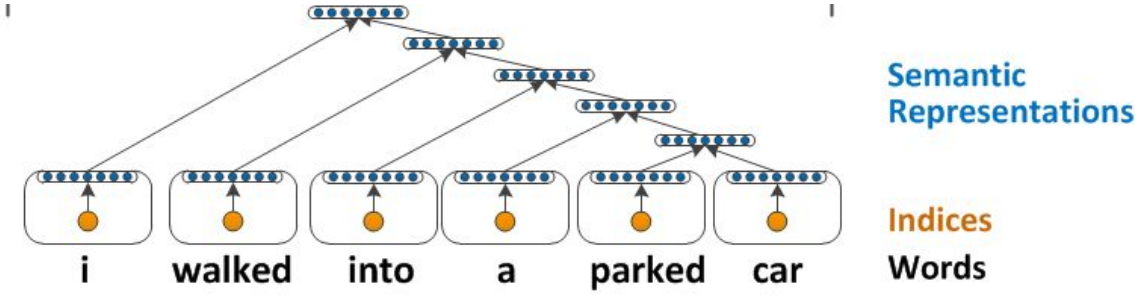
As mentioned before, I think that funny is a combination of context, words’ meanings, and sentence structures. First of all, words topics/sentiments can be represented by vectors made with algorithms such as LDA, but we need to label topics once words have been clustered in order to have meaningful dimensions.

Then, context can be represented by expanding the word vector representation by manually adding topic/sentiment dimensions to the vectors. For example, we first define 5 more dimensions to describe context. Thus, the first one means clean/dirty, 2 - funny/normal, 3 - positive/negative, 4 - religious/anti-religious, and 5 - uplifting/sad with domain from -10 to +10. Then, several people can manually label words. Love can be define as the vector [...,...,10,0,10,5,8]. Bikini can be defined by the vector [...,...,-3,4,-2,-5,0], and so forth. I think that words have to be label manually because there is no algorithm that can map cultural context. It is something that has to come from labeled data.



This image shows how labeled dimensions are so much useful.

Finally, labeled words together with labeled sentences can be used to train a recursive neural network to map sentences to our meaningful vector space. This technique would be more useful than just classifying a sentences as very-negative, negative, neural, positive, and very-positive

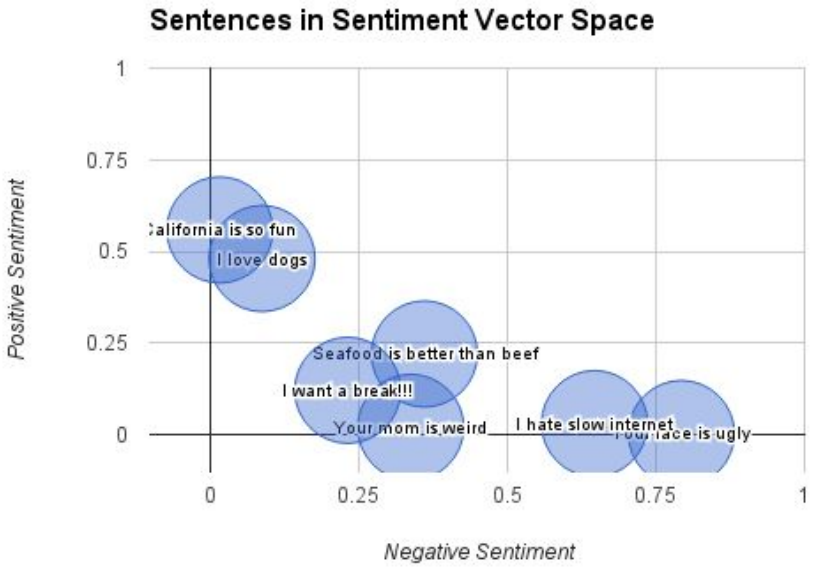


C) Applications

- Neighboring sentences would represent syntactically similar sentences
- Neighboring words would represent synonyms
- Orthogonal words would represent antonyms
- It would allow us to easily identify clean-funny jokes
- It would let us have a richer sentiment analysis representation
- Could help translation. For example, an English sentence is mapped to a vector which encapsulates meaning, and sentiment as an ID, then a decoder could take this meaning-sentiment vector and decode it into a French sentence with the same meaning and sentiment.

3 - Analysis of performance of method

Even though the Stanford NLP Library did not implement a funny classifier, I did run it and studied its outputs. I was able to classify 100 sentences with 90% accuracy. As mentioned before, it returns a vector of very-negative, negative, neural, positive, and very-positive; however, the following image only shows the positive dimension vs the negative dimension.



In conclusion, I think that classifying funny from not funny is harder than classifying positive from negative because we need to know cultural context which can only be completely described if words are manually labeled. On the other hand, the Stanford NLP algorithm is the state of the art when it comes to sentiment analysis, but I think that it would be infinitely more useful if it returned a vector of several sentiments/topics instead of 5 gray scales of positive-negative.