ECE 467
Spring 2018
George Ho

# Final Project Proposal

## Basic Idea

I am interested in creating and comparing word embeddings for the purposes of text categorization. In essence, I'd like to essentially repeat the first NLP project (on text categorization), but instead of using tf-idfs to vectorize words, I will create and utilize word embeddings. The difference in performance will be interesting to see: in lecture, the truism that "something simple will get you 90% of the way there, but the next 2 or 3% require a lot of sophistication" really struck me, and I want to see how much improvement these cutting-edge techniques will give me.

## Background Research

I intend to create the word embeddings using the popular, robust and well-documented topic modelling library gensim (Rehurek), which implements the most well-known word embedding algorithm, word2vec (developed by Google). Another algorithm I want to use is fastText (developed by Facebook), which is an extension of word2vec, and generally expected to work slightly better, although having a slightly more complicated interface and training time.

It is also good to note that there are the open-source GloVe word embeddings for general-purpose English, which are trained on a large crawl of the web. However, Nadbord Drodz finds that creating a corpus-specific word embedding is almost always better than using a general-purpose word embedding, as one might expect (Drodz).

There have been attempts to benchmark text categorization systems with and without word embeddings: Drodz finds that word embeddings can generally yield a couple of percentage points of accuracy, but suspects that fine-tuning how the word embeddings are used (i.e. how do we get from word embeddings to “document vectors” which Rocchio will use) may yield much more substantial results. This is something that will require exploration and evaluation on my part: once we have word embeddings, what do we do to get a vector for a document? Possible ideas are outlined in the detailed description.

## Detailed Description and Evaluation

As described above, I want to re-implement a Rocchio text categorizer, but instead of using tf-idfs, I will create and use word embeddings. The vectorization scheme will be one part that requires experimentation. Below are some ideas proposed by Drodz and on CrossValidated:

- For each document, simply average the word embeddings of each word in the document (Drodz)
- For each document, take the component-wise maximum value (across all words in the document) for each component (CrossValidated)
- For each document, take the component-wise minimum value (across all words in the document) for each component (CrossValidated)
- Concatenate the componentwise maxima and minima (CrossValidated)
- Use an alternative word embedding algorithm known as doc2vec, which takes the text categories into account while learning the embedding (Drodz, Rehurek)

Not only will the vectorization scheme require experimentation, but there are many different word embeddings that are possible: I intend to try 3 embeddings in total: the GloVe general-purpose embedding and corpus-specific word2vec and fastText embeddings.

Of course, there are other components to worry about as well: should stemming or lemmatization be used? Is part of speech tagging helpful? How will tokenization and stopwords work? While these questions have all been answered in the first project, I suspect that they will change a lot depending on the specific word embedding used.

In terms of the actual program, the text categorizer will look very similar to that of my previous project, but will now prompt the user to train a word embedding in the training phase, and prompt the user for a word embedding file in the testing phase. Note that due to the way I implemented Rocchio in the first project (I assumed that only tf-idfs would be used), I expect that a large portion of the categorizer will need to be rewritten. If this is the case, then I would like to use open source implementations of similar nearest-centroid classifiers.

Finally, to evaluate the categorizer, I will look at the classification accuracy, but also the precision, recall and F1 scores of the categorization, and use my previous Rocchio-tfidf categorizer as a benchmark. It should be the case that the Rocchio-word embedding categorizer does better: in fact, if it does not, I would be willing to say that there was a bug in my implementation.

## Deliverables and Demo

Deliverables:

1. The word embeddings themselves (probably as zipped tarballs)
2. The code for the Rocchio-word-embedding text categorizer.

Demo:

1. If we have not yet covered word embeddings in class, I will provide a brief 3-minute explanation of what they are.
2. I will then demo my text categorizer with and without the word embeddings, and summarize what I've discovered about how word embeddings affect the performance of the text categorization system.

## Bibliography:

Drodz, Nadbor. "Text Classification With Word2Vec." DS Lore, 20 May 2016, nadbordrozd.github.io/blog/2016/05/20/text-classification-with-word2vec/. Accessed 31 Mar. 2018.

D. W. "Apply word embeddings to entire document, to get a feature vector." CrossValidated, 7 Oct. 2016, stats.stackexchange.com/questions/221715/apply-word-embeddings-to-entire-document-to-get-a-feature-vector. Accessed 31 Mar. 2018.

GloVe: Global Vectors for Word Representation. nlp.stanford.edu/projects/glove/. Accessed 31 Mar. 2018.

Olah, Christopher. "Deep Learning, NLP, and Representations." Colah's Blog, 7 July 2014, colah.github.io/posts/2014-07-NLP-RNNs-Representations/. Accessed 31 Mar. 2018.

Rehurek, Radim. "Word2vec." Gensim Documentation. Gensim: Topic Modelling for Humans, 1 Mar. 2018, radimrehurek.com/gensim/models/word2vec.html. Accessed 31 Mar. 2018.

---. “Word2vec.” Gensim Documentation. Gensim: Topic Modelling for Humans, 1 Mar. 2018, radimrehurek.com/gensim/models/doc2vec.html. Accessed 31 Mar. 2018.