Project 1: The Smart Search Engine Company

40 points, allowed to submit a result with a group of up to 3 people

Setup: You, an enterprising student in the arts of NLP, have decided to create a startup based on a search engine for text. But this isn't just any search engine. You will construct a smart search engine which will amaze people with its ability to read minds and return related information that isn't even in the query. You will do this by building two seperate models: one for part of speech (POS) tagging and an embedding model. The POS tagging model will scan your documents for entities and store these entities in one index. The embedding model will allow you to perform a smart search over these entities by finding ones that are close in similarity within semantic (or embedded) space.

Background: How does a search engine work? https://computer.howstuffworks.com/internet/basics/search-engine.htm (https://computer.howstuffworks.com/internet/basics/search-engine.htm)

At its core, a search engine is built off of an *index*, a way to store pieces of text that can then relate to the original document that will be returned. When you submit a query, the query is compared to those pieces of text to determine *relevancy*.

In our case, the index will be based off of phrases tagged by the POS tagging module, and relevancy will be determined by the cosine similarity between the embedding of the query and the embedding of the entities.

Steps:

To perform this task, you will need to:

1. Train a POS tagging model
2. Train or use an existing word2vec model
3. Run the POS tagging model on a corpus
4. Embed the entities using a word2vec model
5. Build two indices - an entity to vector index and an entity to document index
6. Construct a search algorithm to compare a query to the entity index and retrieve a document
7. Write code to query and return results

Training a POS tagging model:

Build your POS tagging model using a recursive neural network from Keras. Ideally, use an LSTM based model since this will reduce errors. Code for BiLSTM: `model.add(Bidirectional(LSTM( # of nodes )))`

Train your LSTM from the data provided in class (see the POS tagging module)

Here is some information on the meaning of this data: http://nlpforhackers.io/named-entity-extraction/ (http://nlpforhackers.io/named-entity-extraction/)

We will go over how to build this model in class, but here are some helpful websites:

- http://dirko.github.io/Bidirectional-LSTMs-with-Keras/ (http://dirko.github.io/Bidirectional-LSTMs-with-Keras/)
- https://gist.github.com/dirko/1d596ca757a541da96ac3caa6f291229 (https://gist.github.com/dirko/1d596ca757a541da96ac3caa6f291229)

Training a word2vec model:

Use material from Lecture 5 on how to do this. Feel free to use a pretrained embedding (such as the Google News embedding) if you prefer.

Run an POS tagging model on a corpus: