KNOWLEDGE EXTRACTION FROM MEDICAL LITERATURE

{ ABHIMANYU BANERJEE } STANFORD UNIVERSITY

BACKGROUND INTRODUCTION AND PROBLEM STATEMENT

- Extracting knowledge from research papers is a time intensive task for humans
- Can we speed up the extraction of relationships such as drug, target gene- drug using NLP? This can potentially speed up drug development by helping curate the vast body of medical literature
- It is extremely difficult to extract relationships from unlabeled data, co-occurance of two entities in a sentence is not enough, it is the context that is important.

DrugName -- BacteriaName with probability **0.000** appeared in sentence 3 of document 6086:

Following ciprofloxacin administration , there was a decrease of Escherichia coli in feces and after clindamycin administration a decrease of Bacteroides in feces and Leptotrichia in saliva , which all returned to pretreatment levels within 1 to 4 months .

Given such a sentence with a drug mention and a bacteria mention, can we say something about the relationship they are in?

DATASETS AND PRE PROCESSING METHODOLOGY USED

Medline contains abstracts of all medical journals

- Data available: A subset of the full Medline corpus containing all the medical abstracts from journals. The size of the dataset is 1.5 GB
- The dataset is parsed and lowercased into a single file to easily train word vectors
- The implementation of word2vec on tensorflow was used to train word vectors on this corpus.
- We need to train word vectors and can not use a pre trained set of vectors as we need vector representations for the entities of our interest which are bacteria names and drug names. We throw out very infrequent words which occur < 5 times in the entire corpus
- We then extract relation "mentions" , sentences which have a drug name and a bacteria name . This is our training data, note that it is **unlabelled** as of now

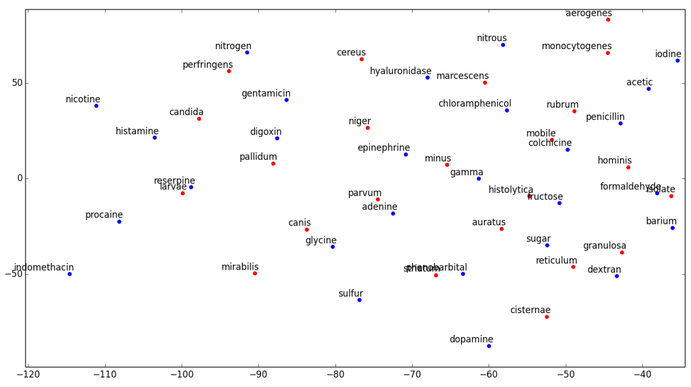
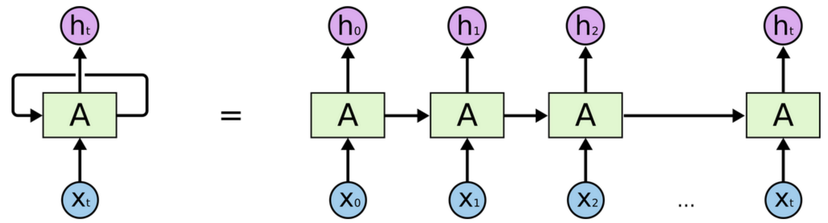


Figure 2: PCA of word vectors generated for drugs and bacteria names

TRAINING METHODOLOGY

- **Creating labeled data with distant supervision:** We use **ddlite** to label our sentences with either a positive or a negative label.
- These computer generated labels for the sentence are based on **heuristic rules** written by us. For example, a sentence with **targets** in the sentence(relation) is marked with a +1 while negative sentences with words like **"not affect"** are marked with a -1
- We now have a computer labeled dataset ready to train(I have 59 different such distant supervision rules) consisting of **1099 sentences**. I split **900 of these for training and 199 for testing**
- I used a recurrent neural network **(LSTM)** to train the data
- All sentences are padded to their average length from the dataset (length 38) and fed into the LSTM.
- The final hidden state is then used to make a judgement on our relationship by feeding it to a Softmax function



An unrolled recurrent neural network.

Figure 1: We keep feeding in the new word, the final hidden state encodes the sentence information. We use the LSTM version of this RNN (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

RESULTS

- **The model quickly overfits on the training set and achieves a classification accuracy of** 1.0 **while training.The best accuracy on the test set is** $.68$
- It is actually quite heartening that we have a number greater than 0.5 on the test set, because this indicates that we are **actually learning what a positive relationship is using only machine labeled data**
- The Neural net is very sensitive to the number of steps in the LSTM. There is a **sweet spot near the average sentence length** (38) at which we have the best test accuracy
- This is somewhat intuitive, a very small length does not capture information while keeping a very long length for the sentence vectors ends up ruining information from the small sentences by too much padding. We got our best results on the $num - steps = 38$
- The hyperparameters that worked best were:
  - batch size: 30
  - base learning rate: 0.001
  - size of hidden layer: 200
  - size of input(word vectors): 128
  - num-steps: 38

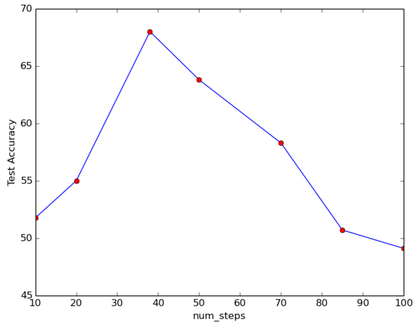


Figure 3: Test Accuracy vs. num-steps in the LSTM(length of padded sentences)