**NLP with classification & Vector Spaces**

**Week 1 Logistic Regression based classification of Tweet Sentiment**

1. Build a vocabulary of unique words from the corpus
   1. Pre-process the data
      1. Tokenize
      2. Stem
      3. Remove stop words
2. Count frequency of each word in negative and positive classes.
   1. This can be done by creating a dictionary
3. Represent each tweet as 3d vector
   1. The first dimension is 1 for bias
   2. Second dimension will be sum of +ive frequencies. So all the words appear (sum their positive frequencies)
   3. Third dimension will be sum of -ive frequencis. So all the words appear (sum their negative frequencies)
4. Fit logistic regression model to train and test data
   1. Use entropy loss function with gradient descent to fit the data
5. Get the final evaluation

**Week 2 Naïve Bayes Classification of Tweets Sentiment**

Naïve – because features used for classification are all assumed to be independent.

Steps to implement Naïve Bayes

1. Build a vocabulary of unique words from the corpus
   1. Pre-process the data
      1. Tokenize
      2. Stem
      3. Remove stop words
2. Count frequency of each word in negative and positive classes.
   1. This can be done by creating a dictionary
3. Sum the positive and negative frequencies
4. Create another table like frequency table but it will be the tweets freq divided by total freq to get the conditional probability that the word is either positive or negative.
5. Compute the Naïve Bayes prob as

1. However, if the word is not present in negative or positive but just one of them, its P will be zero and so we may not be able to solve this. Hence, we use **Laplacian smoothing** to get non-zero probabilities.

P(wi|class) = freq(wi,class)/Nclass,

P(wi|class) = (freq(wi,class) + 1)/(Nclass + V), V 🡪 number of unique words in vocab

Ensures no 0 probability

1. Add a prior ratio – probability of positive/prob of negative

Prior Likelihood

1. Log Likelihood (to account for risk of underflow as we are multiplying lot of numbers between 0 & 1). Log the above equation to get log prior + log likelihood
2. Compute λ. Word Sentiment (as opposed to entire tweet Sentiment)
   1. Ratio(w) = P(w|+ive)/P(w|-ive)
   2. λ(w) = log(ratio(w))

**Training Naïve Bayes Classifier**

1. Pre-process
   1. Lowercase
   2. Remove punctations, stop words, url handles,
   3. Stemming
   4. Tokenize
2. Compute vocabulary 🡪 freq(w,class)
3. Conditional probability with Laplacian smoothing formula
4. Calculate λ
5. Compute log prior 🡪 log(# of +ive tweets/# of -ive tweets)

Naïve Bayes analysis can be used for more than sentiment analysis.

* Author identification using conditional probabilities
* Spam filtering
* Information retrieval (relevant vs irrelevant doc given query)
* Word disambiguation. Determining context of word. For instance, bank (is it river bank or bank bank). Determine the probability of word given other part of tweet to determine which it is.

**Assumption of Naïve Bayes**

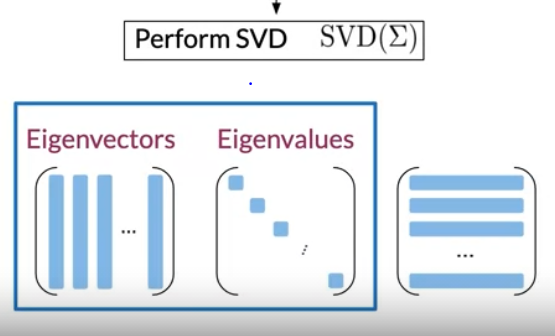
* Independence of words in a sentence. Word go together.
* Relative frequency in corpus

Issues

* Word Order
* Punctuation
* Adversarial attack – common language phenomenon like sarcasm, irony, euphemism.

**Week 3 Vector Space Models**

* Helps determine relationship amongst words (Naïve assume no relationship which was incorrect).
* Create a co-occurrence matrix (number of times words occur together within a certain distance k)
* Co-occurrence Word/Word or Word/Document
* Calc Euclidean Distance. Issue – if the number of docs in corpora are unevenly distributed we may get incorrect Euclidian distance.
* Instead of Euclidean distance use cosine similarity.
* Word embeddings 🡪 represent word as a vector but the vector has meaning associated with it. Word2vec. King – man + woman 🡪 queen
* PCA to reduce dimensions to help with visualization.
  + Find non-correlated features – eigen vectors
  + Project data on the non-correlated axis. Retain as much as value as possible – eigen values.
  + Eigen values and eigen vectors are obtained by taking SVD of covariance matrix



**Week 4 Machine Translation & Document Search**

* Use Forbenius norm to determine the distance the between XR & Y.
  + X 🡪 Language A word embeddings
  + R 🡪 Matrix to transform X to Y
  + Y 🡪 Language B word embeddings
* Use locality sensitive hashing to bucket the or cluster the vectors.
* By value of dot product, we can determine the relative position of the vector in the plane (above or below)
* Approximate nearest neighbors
  + Multiple sets of random vector planes to determine all the possible nearest neighbors as given by random planes. Like a random forest.

https://code.google.com/archive/p/word2vec/ --> pretrained word embeddings