Assignment 1

Aim

1. Corpus Compilation: The dataset contains ids of the tweets on 2016 United States Presidential Election. A corpora has to be created out of the tweets by concatenating the individual tweets. Remove the URLs and hashtags if they are present in tweets (you might need to use regular expressions to identify URLs). You may use Tweepy API or any other library to crawl tweets for given ids.

2. Build Language Models: Use the code snippets used during the tutorial to build n-gram language models.

3. Experiment: Use different smoothing techniques, Laplace, Good-Turing and Kneser-Ney and report perplexity values using hold-out test.

Procedure

1. Downloaded 2016 US election tweets of Democratic candidates `democratic-candidate-timelines.txt` from https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/PDI7IN.

2. Compiled a corpus after scraping around 38000 tweets from twitter using tweepy api, removing urls and hashtags which identified using regex.

3. Built an n-gram language model using nltk python module.

4. Performed different smoothing techniques, Laplace, Good-Turing and Kneser-Ney and got perplexity values using hold-out test.

Observations

The following values were obtained after experiment with various smoothing techniques.

1. Laplace smoothing

$$P(w_i|w_{i-1}) = \frac{1 + c(w_{i-1}, w_i)}{c(w_i, *) + |v|}$$

Sentance probability of “donald trump is president” using 1-gram model: *2.2366754357194043e-16*
Sentance probability of “donald trump is president” using 2-gram model: *1.9563683223160525e-32*
Sentance probability of “donald president is trump” using 3-gram model: *1.0742966618801247e-41*

Entropy of 1 gram model: *0.030073968845450105*
Entropy of 2 gram model: *2.942622120276869e-06*
Entropy of 3 gram model: *1.8519259228263996e-11*

Perplexity of 1 gram model: *1.0210644756648108*
Perplexity of 2 gram model: *1.0000020396723062*
Perplexity of 3 gram model: *1.0000000000128366*

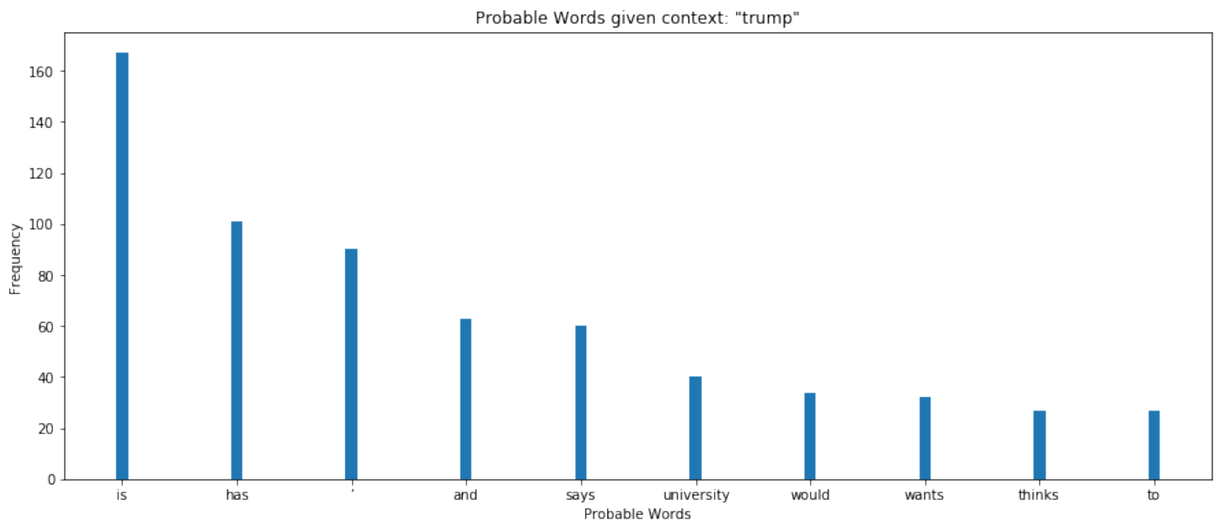

Fig 1.1 Probable words using 2 gram model

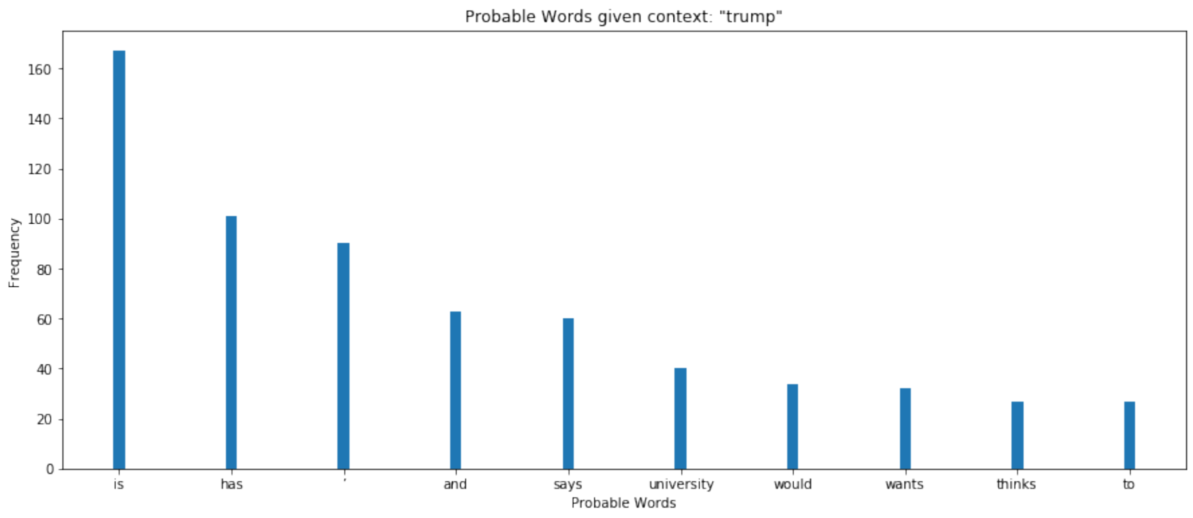

Fig 1.2 Probable words using 3 gram model

2. Good-Turing smoothing

$$c^* = (c+1) * \frac{N_{c+1}}{N.\, N_c}$$

Sentance probability of “donald trump is president” using 1-gram model: *3.0550359208594214e-13*
Sentance probability of “donald trump is president” using 2-gram model: *0.0*
Sentance probability of “donald president is trump” using 3-gram model: *1.326867473598325e-38*

Entropy of 1 gram model: *0.031993538436617442*
Entropy of 2 gram model: *0.12840002076037144*
Entropy of 3 gram model: *0.096746549144416095*

Perplexity of 1 gram model: *1.0224239513525097*
Perplexity of 2 gram model: *1.0930807791600625*
Perplexity of 3 gram model: *1.0693592076976735*

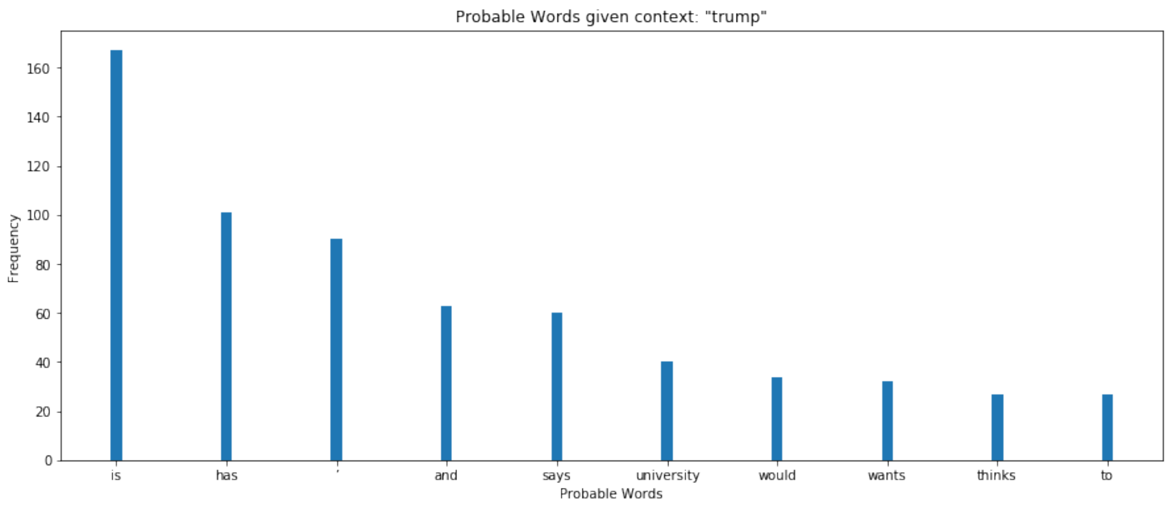

Fig 2.1 Probable words using 2 gram model

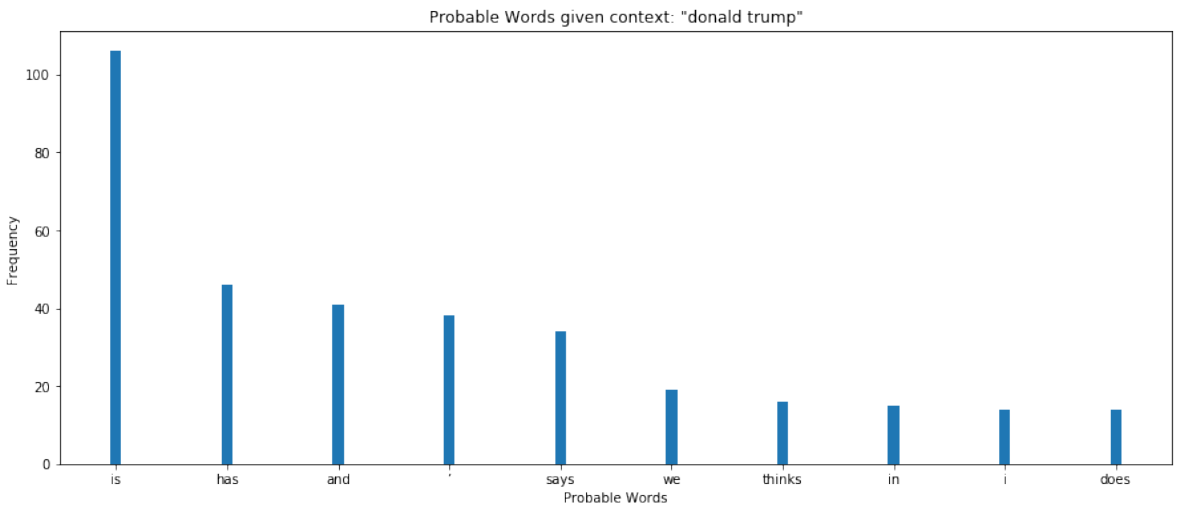

Fig 2.2 Probable words using 3 gram model

3. KneserNey smoothing

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{CONTINUATION}(w_i)$$

λ is a normalizing constant; the probability mass we've discounted

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} \left| \{ w : c(w_{i-1}, w) > 0 \} \right|$$

Sentance probability of “donald trump is president” using 3-gram model: *0.0*

Entropy of 3 gram model: *0.12448071741346098*
Perplexity of 3 gram model: 1.0901152867506776