CS 565: Assignment-1

Basic pre-processing: Segmentation, N-Gram Analysis, Collocation

Due Date: 5^{th} **February,** 2017

An *n-gram* is a sequence of *n* items from a given sequence of word or text. The items can be letters, words or akshara according to the application, for our case it would be words only. For instance lets say corpus is *"His relationship with many western nations was troubled during his tenure as chief minister..."* then the list of unigrams would be { *His, relationship, with, many, western...*}, bigrams would be { *His relationship, relationship with, with many,...*} and similarly trigrams would be {*His relationship with, relationship with many, with many western...* }. The given example of *n-grams* are contiguous, whereas we can also create a list of non-contiguous *n-grams* as we discussed in the class. While non-contiguous *n-grams* can be useful for finding collocations, contiguous *n-gram* analysis is more common for downstream applications such as language modeling, machine translation, text categorization etc.

Objective of the assignment is to get started with basic NLP tasks, exploring available tools and choosing the one which you like the most. Some of the famous tools are available on the course website. For example, NLTK is famous nlp toolkit for python language; Stanford corenlp tool is from the famous Stanford NLP group, Apache openNLP etc. For "Getting Started", Sunil has chosen the reference material for tokenization and *n-gram* calculation by Dragomir's notes.

Finer points of assignments are as follows:

- Your assignment and project group remain the same.
- Submit joint report and code. Mention individual contribution.
- Any assignment related queries and discussions will not be entertained on my personal email-id. You should use Canvas for the same.
- No late submission will be entertained.
- You will be informed about submission procedure of your report and code by a separate post on Piazza.

1 Getting Started: Part I

1. Download any freely available corpus or take any accessible corpus from your chosen tool. Explore available sentence segmenter from the tool and use that for sentence segmentation.
2. Create a dictionary after detecting words from all sentences.
3. Find all possible unigrams. For each unigram, calculate its frequency in the given corpus. Plot the frequency distribution.
4. Find all possible bigrams and calculate their frequencies. Plot the frequency distribution.
5. Similarly find all trigrams possible and calculate their frequencies. Plot the frequency distribution.
6. Each group is expected to explore two tools (e.g. NLTK and Apache OpenNLP).

7. In the report, summarize the options available for sentence segmentation as well as for word tokenization. Further discuss one method (machine learning, or rule based) used for each of the two tasks. Compare the results obtained by two different methods of word tokenization. In your repost, you should also discuss about frequency distribution of the three different cases. Specifically, can you say what distribution(s) they are following?

2 Few Basic Questions

1. How many (most frequent) words are required for 90% coverage of the selected corpus?
2. How many (most frequent) bigrams are required for 80% coverage of the corpus?
3. How many (most frequent) trigrams are required for 70% coverage of the corpus?
4. Repeat the above after performing lemmatization.
5. Compare the statistics of the two cases, with and without lemmatization.
6. Summarize the results in the report.

3 Writing some of your basic codes and comparing with results obtained using tools

1. Repeat section 2 after implementing discussed heuristics in the class for sentence segmenter and word tokenizer. If you want to improvise on the discussed heuristics, you can do that but you should describe your heuristics in the report. Also summarize your findings by comparing the results obtained using your heuristics and tools.
2. Implement Pearson's Chi-Square Test for finding all bigram (contiguous) collocations in your chosen corpus. Do not use libraries. Discuss if you have made any interesting observation.

4 N-gram language model

Consider any ebook of your choice from `https://www.gutenberg.org/` as a corpus. Prepare the training, development and test set as follows: After sentence segmentation and word tokenization, randomly shuffle sentences and split them into two parts of 90%(**part-1**) and 10%(**part-2**) of all sentences. **Part-2** constitues the independent test set and will remain fixed. **Part-1** will be used for training and development purpose.

1. Implement a tri-gram language model using Backoff and Interpolation smoothing methods.
2. Use **Part-1** to split the data into two parts of 90% (training data) and 10% (development/validation) set. Do this five times to prepare five different pairs of training and development sets.
3. Report model's performances in terms of perplexity and Likelihood on a held-out (development/validation) sets. Do you obtain the same set of parameters?
4. Report model's performance on independent test set.
5. Summarize results in the report including discussions on which smoothing methods had a least variance and what happens if only Laplace smoothing is used.

Reference: Speech and Language Processing, 3rd edition, Chapter 4. Url: `https://web.stanford.edu/~jurafsky/slp3/4.pdf`

Remarks: Assignment submission guideline will be provided by Feb 3, 2017.