CS6370: Natural Language Processing

Assignment 1 (Part-A)

Release Date: 20th Feb 2025 Deadline: 8th March 2025

Name:	Roll No.:

General Instructions:

1. This assignment consists of two parts: A and B. Part B will be released later with a separate deadline.
2. The template for the code (in Python) is provided in a separate zip file. You are expected to fill in the template wherever instructed. Note that any Python library, such as nltk, stanfordcorenlp, spacy, etc, can be used.
3. The programming questions for the Spell Check and WordNet parts need to be done in separate Python files.
4. A folder named 'Roll_number.zip' that contains a zip of the code folder and your responses to the questions (a PDF of this document with the solutions written in the text boxes) must be uploaded on Moodle by the deadline.
5. You may discuss this assignment in a group, but the implementation must be completed individually and submitted separately.
6. Any submissions made after the deadline will not be graded.
7. Answer the theoretical questions concisely. All the codes should contain proper comments.
8. The institute's academic code of conduct will be strictly enforced.

The goal of this assignment is to build a search engine from scratch, which is an example of an Information Retrieval system. In the class, we have seen the various modules that serve as the building blocks for a search engine. We will be progressively building the same as the course progresses. This assignment requires you to build a basic text processing module that implements sentence segmentation, tokenization, stemming/lemmatization, spell check, and stopword removal. You will also explore some aspects of WordNet as a part of this assignment. The Cranfield dataset, which has been uploaded, will be used for this purpose.

Part 1: Sentence Segmentation [Theory + Implementation]

1. Suggest a simplistic top-down approach to sentence segmentation for English texts. Do you foresee issues with your proposed approach in specific situations? Provide supporting examples and possible strategies that can be adopted to handle these issues. [2 marks]

2. Python NLTK is one of the most commonly used packages for Natural Language Processing. What does the Punkt Sentence Tokenizer in NLTK do differently from the simple top-down approach? [1 marks]

3. Perform sentence segmentation on the documents in the Cranfield dataset using:
   a. The proposed top-down method and
   b. The pre-trained Punkt Tokenizer for English

   State a possible scenario where
   a. Your approach performs better than the Punkt Tokenizer
   b. Your approach performs worse than the Punkt Tokenizer [4 marks]

Part 2: Tokenization [Theory + Implementation]

1. Suggest a simplistic top-down approach for tokenization in English text. Identify specific situations where your proposed approach may fail to produce expected results. [2 marks]

2. Study about NLTK's Penn Treebank tokenizer. What type of knowledge does it use - Top-down or Bottom-up? [1 mark]

3. Perform word tokenization of the sentence-segmented documents using
   a. The proposed top-down method and
   b. Penn Treebank Tokenizer

   State a possible scenario along with an example where:
   a. Your approach performs better than Penn Treebank Tokenizer
   b. Your approach performs worse than Penn Treebank Tokenizer

   [4 marks]

Part 3: Stemming and Lemmatization [Theory + Implementation]

1. What is the difference between stemming and lemmatization? Give an example to illustrate your point. [1 marks]

2. Using Porter's stemmer, perform stemming/lemmatization on the word-tokenized text from the Cranfield Dataset. [1 marks]

Part 4: Stopword Removal [Theory + Implementation]

1. Remove stopwords from the tokenized documents using a curated list, such as the list of stopwords from NLTK. [1 marks]

2. Can you suggest a bottom-up approach for creating a list of stopwords specific to the corpus of documents? [1 marks]

3. Implement the strategy proposed in the previous question and compare the stopwords obtained with those obtained from NLTK on the Cranfield dataset. [2 marks]

Part 5: Retrieval [Theory]

1. Given a set of queries Q and a corpus of documents D, what would be the number of computations involved in estimating the similarity of each query with every document? Assume you have access to the TF-IDF vectors of the queries and documents over the vocabulary V. [1 marks]

2. Suggest how the idea of the inverted index can help reduce the time complexity of the approach in (2). You can introduce additional variables as needed. [3 marks]