COMP4650/COMP6490 Document Analysis
2025 Semester 2

Computing Lab 1

Q1: Text Pre-processing

Part I

(This is a theory question that does not require coding.)

Before attempting this question, please read the following sections from Chapter 2 of "Speech and Language Processing".[1]

- 2.2 Words
- 2.3 Corpora
- 2.5.1 Top-down (rule-based) tokenization
- 2.6 Word Normalization, Lemmatization and Stemming
- 2.7 Sentence Segmentation

Please also read the following section from Chapter 2 of "Introduction to Information Retrieval".[2]

- 2.2.2 Dropping common terms: stop words

(a) List the major steps to pre-process text that you have learned from the lecture and the reading materials above. Briefly describe each step.

Answer:
The major text pre-processing steps include tokenisation, word normalization, sentence segmentation (or sentence splitting), and stop word removal.

Here is a brief description of each step:

- Tokenisation is the process of breaking down a piece of text into a sequence of "tokens", where each token is a sequence of characters that forms a meaningful semantic unit for processing. We can loosely think of tokens as words or terms.
- Word normalisation is the process of putting words into some standard format. Examples include case folding, lemmatisation, and stemming.
- Sentence segmentation is the process of locating the sentence boundaries in a piece of text.
- Stop word removal is the process of removing words that are common in a collection of documents and do not convey valuable semantic meanings.

(b) List some major differences between stemming and lemmatisation.

Answer:
Some major differences between the two word normalisation methods are listed below:

[1] https://web.stanford.edu/~jurafsky/slp3/2.pdf
[2] https://nlp.stanford.edu/IR-book/pdf/02voc.pdf

- The result of stemming does not have to be a real word. The result is used for finding words that are equivalent. For lemmatisation the result should be an actual word and thus requires a dictionary.
- Stemming removes some portion from the end of words (normally using simple rules). Lemmatisation can replace words entirely. e.g., *good* → *better* (which requires a dictionary).
- Lemmatisation also requires knowledge of the context (e.g., the intended Part-of-Speech of a word in the context) while the result of stemming is context-independent.
- Usually, lemmatisation takes longer to compute.

(c) Discuss why stop word removal is not a good idea for certain NLP tasks or applications. Use one NLP task or application to explain the reason.

Answer:
Many NLP tasks and applications need to operate on the original sentences to obtain correct or meaningful results. For example, in sentiment analysis, some phrases that express a clear positive or negative sentiment may contain stop words. If these stop words are removed, the expressed sentiment will be lost. E.g., "one of a kind" becomes "one kind" when "of" and "a" (which are commonly included in a stop word list) are removed. This completely changes the meaning of the original text and the sentiment conveyed.

Part II

(This is a practice question that requires coding.)

In the notebook `lab1-text_preprocessing.ipynb` you will explore techniques for text pre-processing including tokenisation, stop word removal, stemming, and lemmatisation, using the Natural Language Toolkit (NLTK)[3]. Work through the notebook and answer the questions in it.

Q2: Term Weighting and Cosine Similarity

(This is a theory question that does not require coding.)

Before attempting this question, please read the following sections from Chapter 6 of "Introduction to Information Retrieval".[4]

- 6.2 Term frequency and weighting
- 6.3 The vector space model for scoring

Consider the following term-document matrix for the 3 terms "quick", "brown", and "fox" in a collection of 3 documents:

	quick	brown	fox
Doc1	3	0	2
Doc2	0	1	1
Doc3	0	3	6

(a) Calculate the tf-idf score of each term in each document.

Answer:
We first calculate the document frequency and inverse document frequency of all unique terms.

[3] `https://www.nltk.org`
[4] `https://nlp.stanford.edu/IR-book/pdf/06vect.pdf`

	quick	brown	fox
df	1	2	3
idf	$\log 3$	$\log \frac{3}{2}$	0

Therefore, the tf-idf score of each term in each document

	quick	brown	fox
Doc1	$3 \log 3$	0	0
Doc2	0	$\log \frac{3}{2}$	0
Doc3	0	$3 \log \frac{3}{2}$	0

(b) Now suppose that a user runs the query "quick fox". Calculate the cosine similarity between this query and each of the 3 documents, where the document and query vectors are given by the tf-idf score of each term. Which document is retrieved first?

Answer:
Note that the tf vector of query "quick fox" is $[1, 0, 1]$ and the idf vector of all unique terms from the document collection is $[\log 3, \log \frac{3}{2}, 0]$, so the tf-idf vector of the query is $[\log 3, 0, 0]$. The cosine similarities between the query and the 3 documents are

- sim(query, Doc1) = sim$([\log 3, 0, 0],\ [3 \log 3, 0, 0]) = 1$
- sim(query, Doc2) = sim$([\log 3, 0, 0],\ [0, \log \frac{3}{2}, 0]) = 0$
- sim(query, Doc3) = sim$([\log 3, 0, 0],\ [0, 3 \log \frac{3}{2}, 0]) = 0$

Therefore, Doc1 is retrieved first.

(c) Explain the importance of the idf component of the tf-idf score. How does the idf change the weights of rare terms and why is this useful in information retrieval?

Answer:
The IDF term weighs rare words higher, whereas words that appear in many documents are given small weights. This is useful for IR because a word that appears in only a few documents is likely related to the topic of those documents, which means they will inform the query results more.