CE/CZ4045 Assignment Part 1

Lim Jun Hong
Nanyang Technological University
@e.ntu.edu.sg

Tammy
Nanyang Technological University
@e.ntu.edu.sg

Delphine
Nanyang Technological University
@e.ntu.edu.sg

Pang Yu Shao
Nanyang Technological University
C170134@e.ntu.edu.sg

ABSTRACT

TODO:

KEYWORDS

NLP, Text Tagging, Information Retreival

1 INTRODUCTION

TODO:

2 APPLICATION: SEARCH ENGINE

Based on the reviews collected in the previous section, A simple search engine is developed which allows the user to search for related reviews based on the user's input.

With the user's input (i.e., the search query), the list of relevant reviews is retrieved and the reviews are ranked according to their similarity, which is presented in descending order to the user.

2.1 Implementation

The application is implemented using `Python 3.7.9` with the following external libraries:

- `pandas`: For general DataFrame processing
- `scikit-learn`: For calculating tf-idf / cosine similarity

2.2 TF-IDF Vectorization of Reviews

For implementing the search engine, the reviews must first be represented by `tf-idf` vectors instead of plain-text. This will allow the computation of the similarity between the document/review vector and the query vector.

To generate the `tf-idf` vectors, every review is first preprocessed by case-folding, removing stopwords and tokenized to generate a **Bag of Words** representation. The term counts are used to generate the `tf-idf` values.

- **Term Frequency** (`tf`), is a measure of how often a term occurs in the document.
- **Inverse Document Frequency** (`idf`), is a measure of how much information the term provides (i.e., how rare the term is) across **all documents**.

The `scikit-learn` library provides a module,

`sklearn.feature_extraction.text.TfidfVectorizer`

which automatically performs basic preprocessing of the text mentioned above and transforms the array of documents into a tf-idf vector while also performing some normalizing and smoothing to the data.

2.3 Querying and Ranking Reviews

A Search Engine is required to take in a user's query and display *relevant* documents to the user. Since tf-idf vectors have already been built for the existing documents (reviews), the user's query can be transformed to a tf-idf vector as well and the similarity can be computed based on some similarity measure.

`TfidfVectorizer` also provides a method `transform`, which builds a tf-idf vector from a document based on the Language Model that it has learned. With both the query and documents represented in `tf-idf` vectors, the similarity between the query vector and all document vectors can be computed. A few distance measures that can be used include:

- Manhattan Distance
- Euclidean Distance
- Cosine Similarity

For our search application, the **Cosine Similarity** measure is used. Search queries have a short length in nature, therefore using Manhattan / Euclidean distance could cause a potential document of interest have a large distance (i.e., appear dissimilar) when compared to the query. For instance, consider the example vocabulary in Table 1 and the following documents and query:

- Document 1: apple apple apple apple apple apple
- Document 2: banana orange
- Query: apple

These would yield the following term count vectors:

$$D1 = \begin{pmatrix} 6 & 0 & 0 \end{pmatrix}$$

$$D2 = \begin{pmatrix} 0 & 1 & 1 \end{pmatrix}$$

$$Q = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$$

By calculating the euclidean distance between the query and the documents, the following Euclidean distances are obtained:

$$Euclidean(D1, Q) = 5$$

$$Euclidean(D2, Q) = 1.732$$

Here, the distance between D2 and Q is smaller than that of the distance between D1 and Q, however it can be observed by D1 should be the more similar document. Therefore, it can be concluded that measures such as Manhattan and Euclidean distances can be affected by the magnitude of the vector, even when both the query and the document vectors share similar components.

The Cosine Similarity overcomes this as it is a measure of the *angle* between two vectors. As search queries are short in nature, it is able to retrieve documents which share similar components to the query vector.

Table 1: Example Vocabulary

Word Id	Word
0	apple
1	banana
2	orange

2.4 Examples

```
Reviews 1 to 5 (out of 167) for search query
"Fountain light show"
==============================================
1: The Jewel is the gigantic shopping mall with varieties
of shops and restaurants. The famous one is the fountain
that shall make you stop to see it.
At night time there are the light show at Fountain.

Recommend for visiting

2: This has to be seen to be believed. Gardens, waterfalls,
restaurants, shops, adventure park and monorail. Allow at
least a few hours here and stay for light show. Light show
not great but colored waterfall is impressive. A must see.

. . .
```