## Abstract

This paper provides an overview of a standard descriptive data mining processing pipeline including data collection, feature extraction, clustering and data-analytic visualization. This pipeline was applied to a set of 24 text documents about Antiquity. By mapping all the documents to a multidimensional space, K-Means and Hierarchical Clustering were used to explore the relationships in between documents. These documents were clustered into different number of groups among which clustering them into 7 groups were the most accurate according the title of each book. And then these documents were visualized on a 2-dimensional space through Multidimensional Scaling to present their relationships.

Keyword: data mining, feature extraction, K-Means, Hierarchical Clustering, Multidimensional Scaling.

## Introduction

(specify the specific environment with the version of each package used in this coursework)

The 24 documents

## Data Collection

The 24 text documents are 24 books of which the content of each page is stored in HTML page. By analysing the structure of the HTML source code, all the text was found inside “**span**” tag with class attributes as “**ocr\_cinfo**”. With that being found, BeautifulSoup, a Python HTML parser library, was then used to parse all the text out of all the HTML files. By combining each book’s text together, it was then stored as JSON format for further processing:

For the convenience of easy observation of clustering results, real book’s names were used.

## Feature Extraction

### Vector Space Model

Vector Space Model(VSM) was used in this experiment to represent all the documents as vectors in a N-dimensional space. Term Frequency (TF) and Term Frequency- Inverse Document Frequency(TF-IDF) were the two main ways to convert each documents to vectors.

Before applying TF and TF-IDF, the text was tokenized and cleaned up. The aim of VSM is to extract important features that distinguish one document from another, thus certain words should be ignored.

1. **Stop-words** (e.g. the, a, is and in etc.) that present frequently but carrying less meanings were ignored;
2. **Low frequency terms**, terms that only occur in one documents were ignored;
3. **High frequency terms**, terms that present in every documents were ignored.

By ignoring these words, a dramatic reduction in terms of dimensionality, from 230 thousand down to 30 thousand, were achieved.

### Building Indexing Dictionary

The first step to model documents was to build up a term indexing dictionary by select all distinct terms present in every documents, using term as the key and index as the value denoted as:

### *tf*

TF is the simplest way of representing each document into N-dimensional space. It simply counts how many times each term in present in a certain document di. The function is defined as follows:

where is defined is:

Therefore, documents are represented as vectors:

Every single term maps to the corresponding dimension, of which the value is the times of the term occurs in the document. And the dimensionality of vectors is the number of vocabularies in the dictionary.

The set of documents were then represented as matrix:

Note: This matrix tends to be very sparse since one term may occur in one document but may not occur in any other documents.

#### Limitation of tf

scales up the terms that occurs many times in a document and scales down terms that might be more informative. Frequency of a term in a document might not accurately reflect the significance of the term. is introduced to solve this problem.

### TF-IDF

### 

measures how important each term is to a document. is the frequency count of a term in a document. IDF stands for inverse document frequency which assigns high values to rare terms and low values to common words. is defined as:

where is the same as before:

where is:

weights vector is then calculated as:

In order to easily calculate , a diagonal matrix was created:

And then can be calculated as:

Finally, I applied L2-normalization process to *. (Note, the normalization is applied on row basis not matrix-basis.)* With being implemented, terms having high frequency in one document but low frequency in other documents (carrying significant meaning in that document) would be given higher weight, vice-versa.

## Clustering

The 24 text documents had been mapped into a multi-dimensional space and were ready for further processing.

Grouping these documents into different groups based to their textual similarity would be helpful to better understand the documents. Two most common and practical clustering techniques, K-Means Clustering and Hierarchical Clustering were used in this experiment.

Before I applied clustering techniques, I took a simple look at of each document. Roughly these documents can be grouped into 7 groups according to their content.

### K-Means

### Hierarchical Clustering

## Visualization

Just being able to analyse the data is certainly not good enough for people to perceive it. Visualizing the findings would definitely help.

In this paper, I used 1-cosine similarity to calculate the distances in between each documents which received the matrix calculated from feature extraction section ( or ). Multidimensional Scaling was used to reduce the dimensionality of the entire corpus down to 2, to which the 24 documents can be plotted using Python library “***matplotlib***”. Another possible method to reduce dimensionality is ***Principal Component Analysis***.

After these documents being clustered and mapped to 2-dimensional space, I assigned different colours to different clusters for the sake of easy observation.

### Multidimensional Scaling