Literature Search

Team Member: Zecheng Zhang, Chenxi Liu, Jun Kui Chen, Shuhao Du

1. Introduction:

As the volume of published literary texts continues to increase rapidly, it has become increasingly difficult to locate specific contents in a timely manner. To address this issue, a literature search engine focused on specific literature sources could facilitate quick and efficient retrieval of desired texts. By inputting paragraphs, quotes, or even individual words, such a search engine could be used to search publicly available patent literature and identify related works. We aim to create a keyword and phrase search system based on literary masterpieces, allowing users to enter any word or phrase and receive a list of books containing those words. This work proposes a solution on processing enormous amounts of texts that could be a fundamental component of a library management system. Thus this design will be a scalable solution for larger-scale applications in the future.

2. Preliminary Design

2.1 Basic Features:

The basic workflows the system needs to support are:

- Submit new book -> Process the book
- User executes a search -> Look up the data -> Return the key words and names/authors/links of the books

In the basic feature we don't need to store book content. When the user searches something, only the name, author and link of the book will be returned.

2.2 Extra Features

As an extra feature, we can consider storing the whole book. Then the corresponding paragraph inside the book will also be returned when users search.

3. Preliminary implementation plan

From an engineering perspective, we need to process the book contents and build the inverted index for the key words. And we also need to create a tool for users to search the data.

3.2 Basic Features

- Process the book content and extract words
    - Process the book line by line and extract the words

    - If we use MapReduce, the key will be the words and the value will be the book information
- Ignore meaningless words like: a, an, is, the…
    - Identify these kinds of words and ignore them
- Single word, bigrams, trigrams or even more needs to be supported
- Storage for words to book mapping
    - If we use MapReduce, the reducer's output can be our database. We need to build an index like skip-list for it though.
    - Or we can use databases, in this case a relational database like MySQL should suffice.
    - We can also use HBase.
- A tool that can query the storage efficiently
    - Once we have the index built or database populated, we can query them easily.

3.2 Extra Features

- Support more complex wording like sentences and even paragraphs
    - This means we need larger storage.
- Another storage for book content
    - Divide the book content into paragraphs, and give each paragraph an id, store them with the id.
    - Each word will map to name, author of the book as well as the id of the paragraph.
- A tool that can query the two storages and combine the results and return
    - Query the word mapping db first, then use the id to fetch the corresponding paragraph.
- Autocompletion in the search bar.
    - We may use a similar probability language model in assignment 1. And outputs e.g. top 10 results.
    - May also use other natural language processing techniques if time permits.
- How do we determine the ranking of the results?

4. Data Source

We can use the website where we can download these books in plain txt format. I.e. http://www.authorama.com/