**COMP 6791 Project 3 Report**

**Name:**

**Mo Chen**

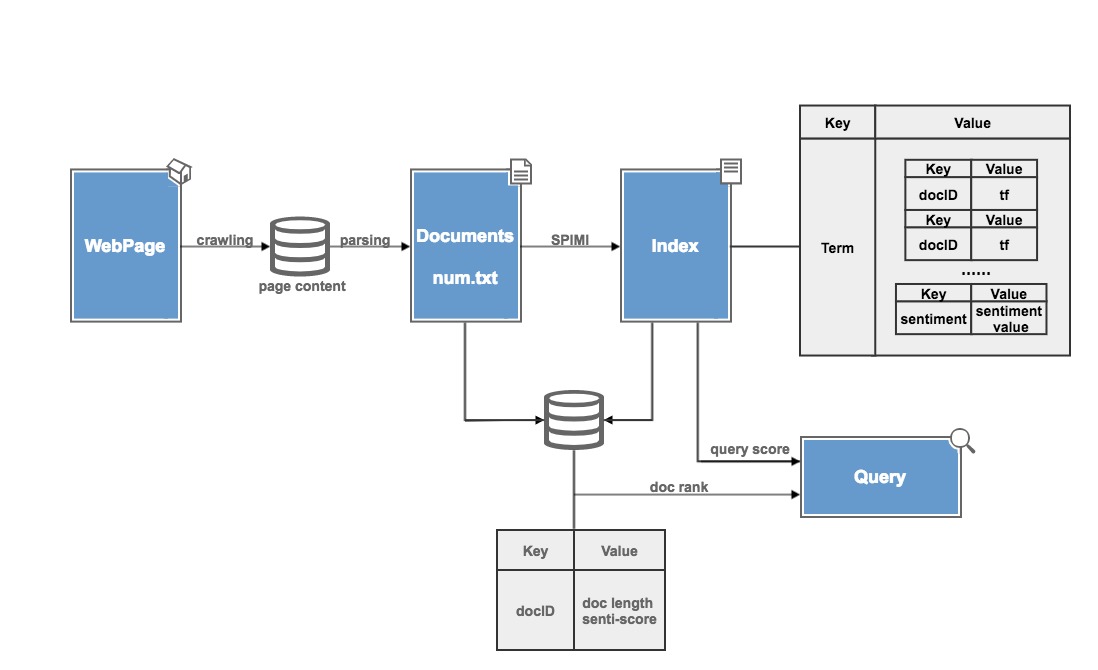
**Tianen Chen**

**Qi Xia**

**Introduction**

For this project, to crawl the web and construct an index for the web documents, we implement the program with Python, using the request and beautiful soup libraries to scrape and analyze the web content.

**Implementation**

****

1. Crawling and Parsing

Through using the request library, we can trace the web links and scrape the content of the website. Then, using the beautiful soup to parse the html documents, we take out the body of the content, and extract some useful information, apply normalization to these information, we can get the documents which meet the format requirements.

1. Constructing Index

We use the SPIMI algorithm to construct the inverted index like what we did in project1 & project 2. The difference is that, in the last two projects, the value of the index dictionary is the docID-tf pairs (also saved as a dictionary, let’s say inner dictionary), but now, we add a “sentiment”-value pair into the inner dictionary, so, we can keep the sentiment value for every term, then calculate the document sentiment value and the query sentiment value. The sentiment value for each term comes from the sentiment dictionary aFinn, if term in aFinn has a score, then give it this score, or make the score as 0, which means neutral sentiment.

1. Score and ranking

To calculate the sentiment score for a document, we also have used the formula of RSVd which we done in the project 2. The reason is we cannot only get the average sentiment score of all the terms in the document, but also should consider the relevance of the document to the query, to say, either the document is only relevant to the query or only has the same sentiment tendency is not entirely correct, thus, we do the multiply, the formula is:

For ranking, if the query is positive or neutral, rank the documents according to the decrease score, or, for the negative rank the documents according to the increase score. Finally, we give the top-10 rank result.

**Q&A**

1. What was the hardest step?

To calculate the sentiment score of document, we have thought about many approaches, such as, give the RSVd and sentiment score weights, then we can try different weights to give a better result. The problem is, sometimes the relevance score is too high, maybe it will reverse the sentiment tendency of the document. For example:

Query: I want to buy a car.

Document: I don’t want to buy a car.

There is only “don’t” is the difference, but it’s the key point of the sentiment guide. We cannot set the weight for this exception, but we also cannot ignore these special cases.

So, at the end, we do the multiply, to make the relevance and sentiment score in a harmonious.

1. How big is the index?
2. How did you define what constitutes a document in your index?

For each document, we have a document length and a sentiment score, document length is used to calculate both the relevance and sentiment value.

1. What observations did you make during your experiments?

We have found many useless web page contents, although we can filter most of them through the program, it still need some artificial recognize to delete those non-relevance content, such as, “page not found”.

1. What did you learn from your experience?

For this project, we exactly feel the complex of the natural language in real life, there is very little possibility that the task which you dealt with is only one language, clear formed, and always useful, a lot of effort should be made to analyze the documents. Besides, the standard how to construct the index and give the suitable rank score to the documents is also very important. Human language is complex, even the occurrence of a “very” in two documents different will bring some subtle gaps. We have considered many examples to evaluate our sentiment formula, and also have tried kinds of query on Google to see how results different, it’s really a difficult task to understand the real meaning of the words in our real life.