**Chinese Document Similarity:**

**Analysis and Ranking based on Keyword Extraction**

Jian Chang, 3693026821, [jianchan@usc.edu](mailto:jianchan@usc.edu)

Yunjie Zhao, 3687586919, [yunjiezh@usc.edu](mailto:yunjiezh@usc.edu)

Tian Lu, 4156843753, [tlu554@usc.edu](mailto:tlu554@usc.edu)

1. **Introduction**

The main purpose of our project is to explore the relationship between documents written in Chinese. At present there are millions of online articles of all fields, and chances are that when we are reading one of these articles, we actually want to get more information about the same topic or read more related essays. It’s natural to come up with the idea to compare essays word by word to make sure whether they are related, but it would be very costly when we have large amount of essays.

Under this consideration, the basic idea to find the most related essays in a large essay pool is that first we extract the keyword for every essay, and then use a Sim-Hash function in the keywords pool to get the most related essays for each given document.

1. **Method**
   1. **Material**

We choose the following corpus to realize our project: Sogou Lab Data (SLD, http://www.sogou.com/labs/dl/c.html). SLD is comprised of a wide variety of corpus. In our project, we analyzed the relationship between about 18,000 pieces of Chinese essays with around 1,000 Chinese character in each essay, and most of them are news report.

* 1. **Procedure**
     1. **Chinese words segmentation**

Unlike English, Chinese words have no space between each other in a sentence. Thus, in order to extract the keywords of each document, the first thing we should do is to parse the sentence into separate words.

We use a third party Chinese words segmentation module called “Jieba” to segment the text. The example is as follows:

Original sentence:近期有许多城市的房价都迎来了一轮新涨幅。

Translation: The housing price in many cities increased again recently.

After parsing:近期/有/许多/城市/的/房价/都/迎来/了/一轮/新/涨幅/。

Translation: recently/ exist/ many/ city/ ‘s/ housing price/ all/ come/ ed/ a round/ new/ increasement/ .

By given the separate words, we can continue the analysis of similarity between documents.

* + 1. **Keywords extraction**

In our plan, we decided to use the TF-IDF algorithm to extract 16 keywords for each document. Detailed procedures are as follows.

* + - 1. **Stop words design**

To make the result more accurate, we have to rule out as many noisy words as possible. We store all the unnecessary words into the file stopwords.txt with one word in each line.

The stop words including basically 4 types:

1. Punctuation, including half-width and full-width. Though punctuation sometimes help us understand the emotion of a sentence, consider the whole view of an essay, they usually mean nothing.
2. Common Chinese words which mean nothing. There are several words appears very frequently in Chinese but have no semantic meaning. Like 的(mean ‘s), 了(show a thing happened in the past), 是(be).
3. Single number or single English character. They usually appear in essays as a mark for listing and are not meaningful.
4. Noise from other sources. For that some essays in the corpus are get from web but without accurate extraction. For example, “&nbsp” and “nbsp” appears many times in the essays, and it’s a space in html which is a completely noise.

In the following calculation, we read in the stop list and ignore the word in essay every time when it’s in the list.

* + - 1. **IDF calculation**

IDF means Inverse Document Frequency, and it’s a parameter to present how often a word appears in the corpus. Generally, IDF gives out the weight of a word. The more common a word appears in all kinds of documents, the less important it is. For example, the word 和(means “and”) appears in most of the document and it’s not crucial at all, and thus in the procedure of extracting keywords, we should give it a low score. And that’s why we use IDF parameter.

To calculate IDF:

**IDF(*t*, *D*) = log( *N*/ *nt* ) ;**

*Where t is a given word, D is the union of all documents, N is the number of documents in D, and nt is the number of documents that word t appears in it.*

We can see from the expression that, the more times a word appears in documents, the lower IDF it will get.

And consider the case that in the target document we may get a word never appears in the corpus, we should do some smoothing:

**IDF(*t*, *D*) = log( *N*/ *(nt+1)* )**

And it’s the final expression we use in our project.

We record the IDF result in the file called IDFfile.txt. The first line is the number of all documents, and from the second line, every line contains a word and the number of documents it appears. The detailed IDF will be calculated in the next step for considering accuracy.

* + - 1. **TF expression selection**

TF stands for Term Frequency, it means the frequency of a word in a specific document.

**TF(*t, d*) = *Ct* / *C* ;**

*Where t is a given word, d is a given document, Ct is the count of word t appears in document d, C is the count of all words in d.*

Normally, the keywords of an essay should appear many times. This parameter gives a positive effect on the final score.

This expression is a raw frequency of a word, mark it as f(t, d). And we have considered the following variants:

1. **TF’(t, d) = 0.5 + 0.5\*( f(t,d)/ MAX{t’ belongs to d}(f(t’,d)) );**
2. **TF’’(t,d) = 1 + log( f(t,d) );**

Analysis from math, we know that when calculating TF-IDF, TF’ weaken the weight of raw frequency by give a smoothing of 0.5, while TF’’ strengthen the weight of raw frequency by using Logarithm to make it more sharp.

During our practice, we find that the expression works best, for that when using TF’ it usually gives the strangest word a high score just because its low appearance even it’s just a noise word or spell error; and TF’’ always mark the most frequent word in a document as keyword even it’s not so important.

Finally, we decide to use TF(*t, d*) = *Ct* / *C*, the raw frequency to extract the keywords.

* + - 1. **TF-IDF calculation and keywords extraction**

To calculate TF-IDF score:

**TF-IDF(t, d, D) = TF(t, d) \* IDF(t, D);**

From the expression we know that, the more frequent a word appears in a given document, the higher score it can get; the less frequent a word appear in corpus documents, the higher score it can get. Thus, the higher score one word gets, the more likely it is to be the keyword.

And for each file, we get the top 16 words who get the highest score and extract them as keywords.

The result are stored in file docKeyWords.txt, with each line represent a document.

* + 1. **Similarity calculation**

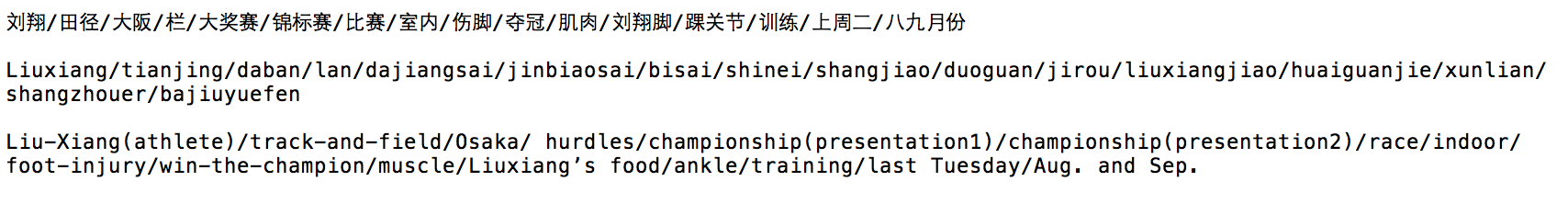
We calculate the similarity and relationship between documents by considering their keywords which are stored in docKeyWords.txt in previous steps.

* + - 1. **Standard Sim-Hash function**

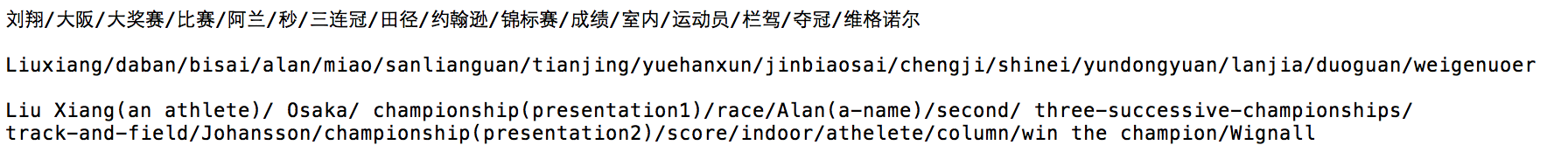
The standard Sim-Hash function are as follows:

1. Use hash function to hash all words into 128 bits 0-1 string;
2. For each bit of the hash code, if it’s 0, then make it –wi; if it’s 1, make it +wi; where wi is the assigned weight of the word, and it can be calculated by IDF;
3. Given all the keywords hash code of a certain document, for the 128 bits get the sum of each bit;
4. For each bit, if it’s greater than 0, make it 1; else, make it 0, and make the 128 bit 0-1 string to be a binary number ***bi*** represent the document;
5. To compare two documents ***i*** and ***j***, do XOR function between ***bi*** and ***bj***. The number of 1 in the result is called humming distance. The smaller it is, the similar they are.

We tried this function, but the result is quite disappointing. Here is one of the typical example. Our target file is C14/1824.txt, call it File1, keywords as follows:

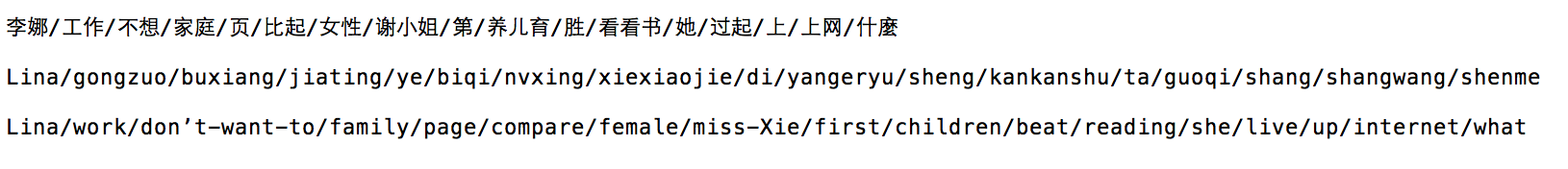


The news talks about athlete Liu Xiang’s race after his injure in Osaka, and our idea related file is C14/1452.txt, call it File2, which talks about the same thing, and given its keywords:



These two keywords set get a humming distance of 41.

However, to our surprise, by using the Sim-Hash function, we get file C22/1582.txt as the most related one with a humming distance of 39, keywords as follows:



It’s obvious that they are totally different! And this situation appears in many testing files.

And we have tried different hash function to generate the result, though every time our idea file File2 get a good score, there are always some totally unrelated file comes out with a lower humming distance.

To explain this, we discussed a lot and got a conclusion that it’s because when we compare these files, each file is compared with other 18,000 code. There may exist some files that with nothing in common but when doing calculation, the keywords of a file may just offset the character of each other and make the humming distance a meaningless thing.

* + - 1. **Modified Sim-Hash function**

To make the result more accurate, we use a modified Sim-Hash function. Instead hash a word into 0-1 string, we take each word as a bit of 0-1.

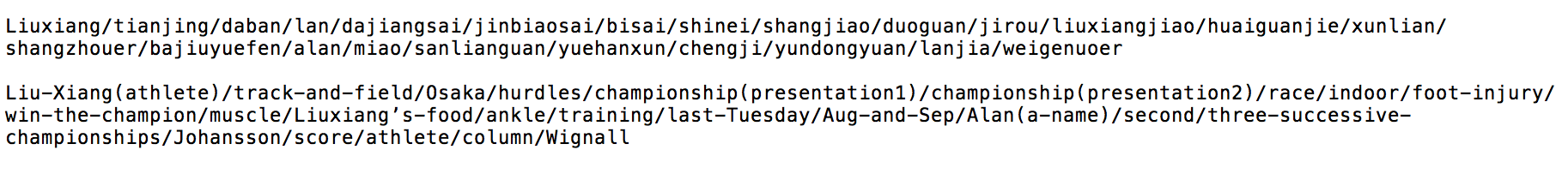
When comparing two sets of keywords, we first get a union of the two sets, then for each file, consider each word in the union, if the word is in the file, then this bit is 1, else, this bit is 0. The final score is decided by the hamming distance of two file divided by the length of the union:

**FinalScore=1-(HammingDistance/UnionLength);**

The larger it is, the more similar the documents are.

Consider the previous File1, File2 and File3. When compare File1 and File2:

*Union sequence:*

**

*File1: 111111111111111100000000*

*File2: 111011110100000011111111*

They have a hamming distance of 16, and the length of union is 24, and final score is 1-(16/24)=0.33. For File1 and File3, hamming distance is 32, and union length is 32, so final score is 1-32/32=0. And thus, File2 is much better than File 3.

Also we tried many other files and this function gets a good result.

* + 1. **Visualization tool**

TBC!

* 1. **Evaluation**

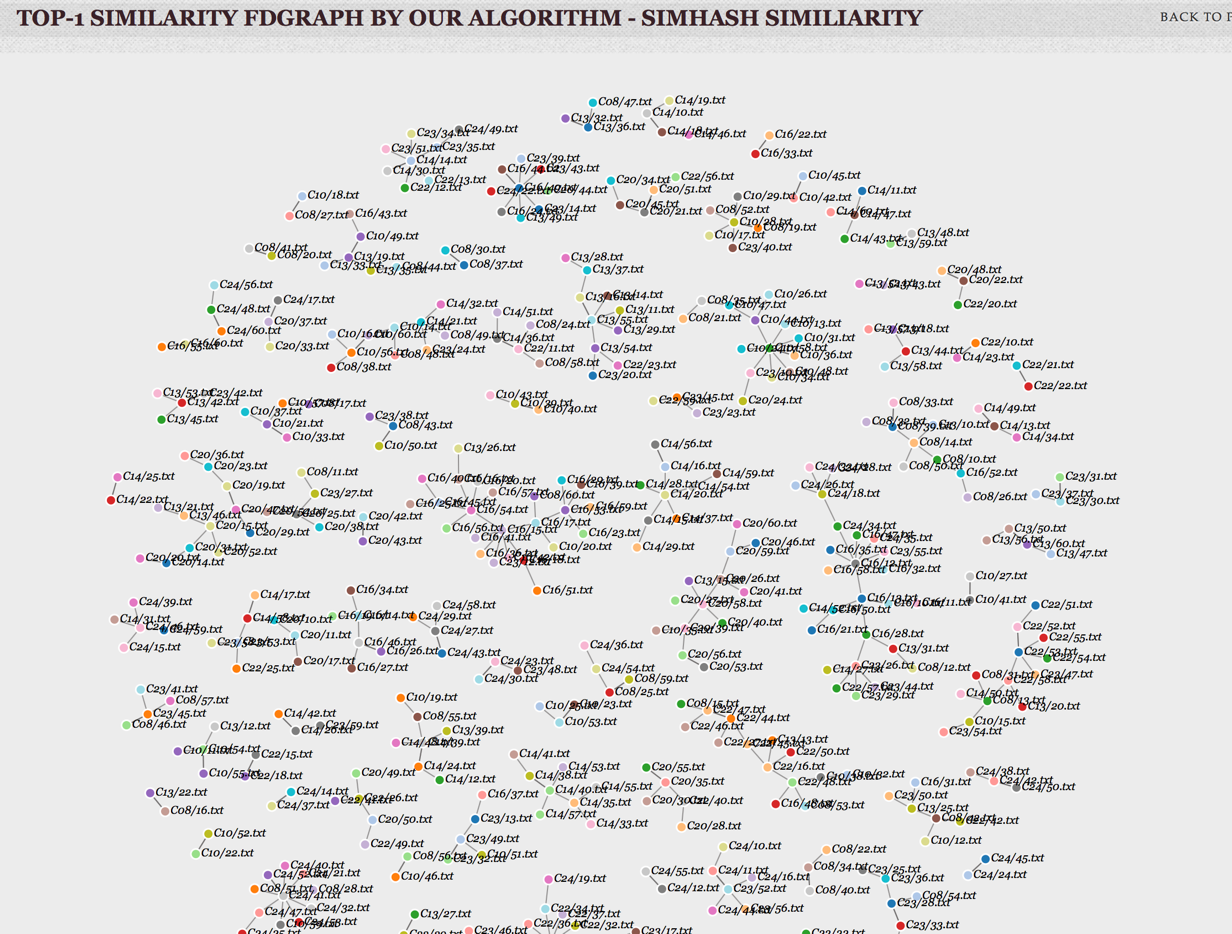
TBC!

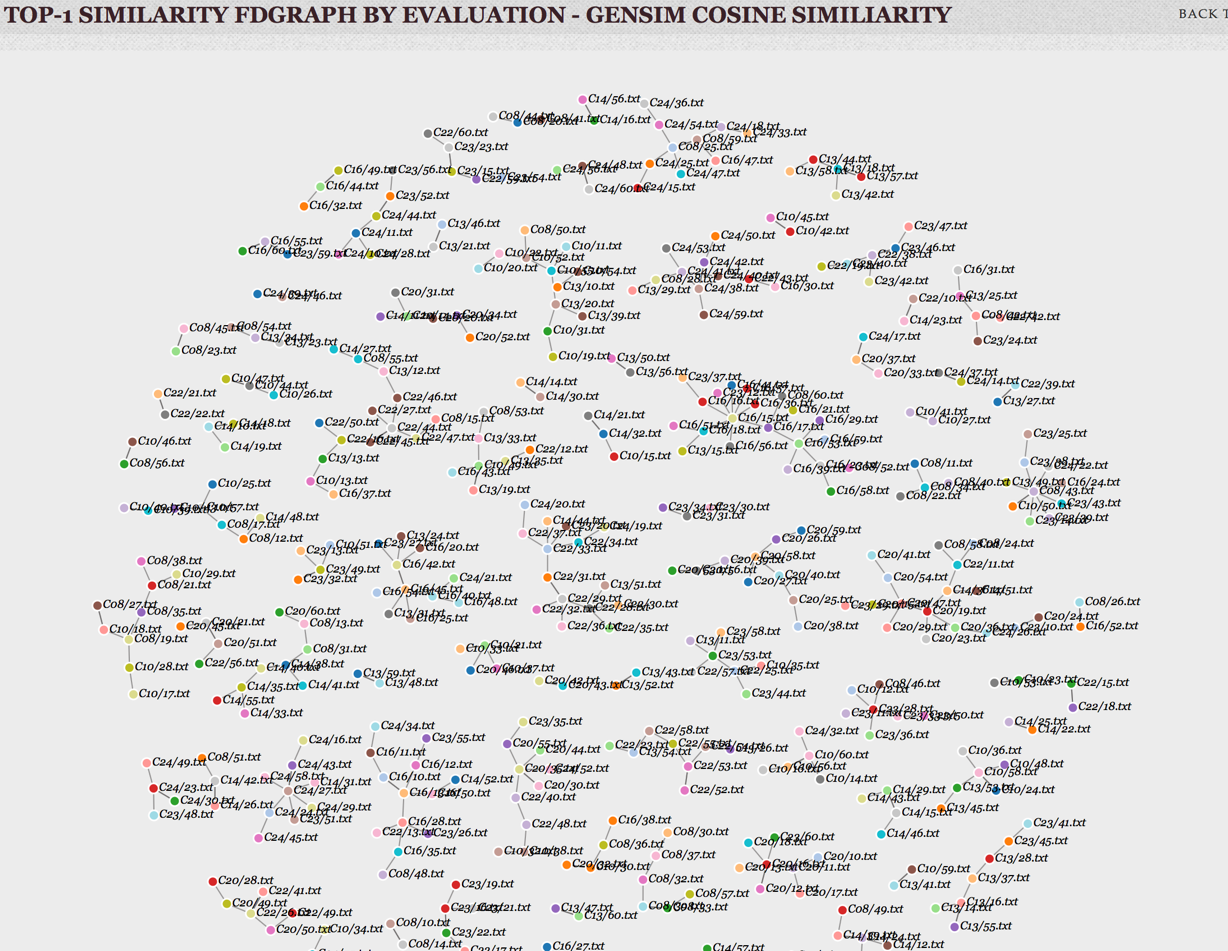
1. **Result**

The final results are stored in Similarity.txt. In each line, the first file name is the target file name, and it is followed by the names of the most related files in the corpus.

**TBC!**

Also, we have put our result in <http://www-scf.usc.edu/~yunjiezh/csci544/projecthome.html> and realize visualization.





1. **Discussion**

From the result we can see that by extracting keywords and using TF-IDF algorithm, we successfully measured the relationship between corpus documents in acceptable time. And the research can be applied in providing the readers essays whose topic is similar to the one they are reading. Especially in the area of academic research, providing readers with relative papers can make their word more efficient.

During our discussion and research, we found that one of the reasons that may causes the inaccuracy is Chinese words segmentation. That’s to say, many proper none, like company names set up by simple Chinese characters are not able to be recognized. In the future research we can study more about the words segmentation and make the final result more accurate.

1. **Reference**

TBC!