A Similarity Web Search Engine

Mengmeng Chen
Computer Science Department
University of Wisconsin-Madison
Wisconsin, 53706

Abstract

Traditional search engines (i.e., Google web search engine) use keywords and perform exact matching to find the target web pages and assign each page with some priority using some algorithms (i.e., PageRank) to capture the popularity. However, there are two problems with this kind of searching. First, it treats the input as individual words instead of a whole document. Second, it performs exact match such that the found pages actually contain all the key words from the user input rather than try to find similar pages that could still render some related information of what the user is trying to look for. This paper describes a project that aims at implementing a web document similarity search engine. Instead of using the input as independent individual words, the engine takes input in the form of document. Furthermore, the matching between the input document and all other pages which are also treated as documents is performed based on the similarity between each other. The results are also ranked based their similarities to the original document. Moreover, to make the results easier to read for the user, LDA (Latent Dirichlet Allocation) is performed on the pages, where each is shown with a label of a sequence of keywords that reveals the its main topic.

Introduction

Web search engine is one of most useful services when users use the Internet. It provides its users the extreme convenience in retrieving information from the Internet. Users with the help of web search engine can fish out a lot of information that are of particular interest to himself/herself without having to go through numberless web pages which is infeasible.

Most of the modern search engines (such as Google web search engine) take input in the form of keywords and perform exact matching between the entire keyword set to the content of target web page. When the related pages are found, the search engine will apply certain algorithm to allocate each page with certain priority to help users find the most useful information quickly.

What is missing here is that it is hard to use this kind of search engine when the user input is instead of keywords a short paragraph or an entire document when the user aims at finding the 'related' or 'similar' web pages. Another noticeable problem is that the result is basically raw web page without any hint or summary on what the content of the web page would be. Users who use traditional web search engine will have to go through a few words context surrounding the key words that is shown along with the URL of the web page. Sometimes, it will not provide enough useful information to allow the user to determine whether the page is interesting or not immediately.

All these problems suggest that the interface will be more user-friendly if the results are could reflect some kind of similarity between the input and the output and if the results can be shown in a more informative manner.

Design

Feature vector

In the area of natural language processing, there are many standard ways of representing a document in the form of feature vectors. I have considered two classical feature vector representations of the input document

Bag of Word Bag of word is a classical feature vector in natural language processing where each document is represented by a vector of the size of the entire vocabulary of the corpus with each bit in the vector is 1 or 0 to indicate if the document contain the corresponding word or not, respectively. This form of feature vector assume that the occurrence of each word is independent to each other and the order can be permuted without affecting the nature of the document.

tf.idf A document d can be represented by a tf.idf vector (many other variations exist, such as raw count vector, or binary indicator vector). tf is the normalized term frequency, i.e. word count vector scaled, so that the most frequent word type in this document has a value of 1.

$$tf_w = \frac{c(w,d)}{\max_v c(v,d)}$$

where c(w, d) is the count of word w in d. idf is inverse document frequency.

The idf part is defined as

$$idf_w = \log \frac{N}{n_w}$$

And we have the complete representation in the form of

$$tf \cdot idf_w = tf_w \times idf_w$$

In my experiment, I have tried both of them and the difference is not significant and is not very clear that one representation is persistently better than the other one.

Similarity metric

To define the similarity of two documents is essentially equal to try to find a distance metric. Again there are many candidates as distance metric. One of the most widely used metric is call cosine similarity, which is defined as

$$\begin{aligned} sim(d,q) &= \cos(\theta) \\ &= \frac{d^T q}{||d|| \cdot ||q||} \\ &= \frac{d^T q}{\sqrt{d^T d}\sqrt{q^T q}} \end{aligned}$$

where the dot product is defined as

$$u^T v = \sum_{i=1}^{V} u_i v_i$$

LDA labeling

I try to come up with a meaningful label with the results from the search engine so that the user without further looking into the exact content of the web page can immediately get a hint of what each web page is about. One way to assign meaningful label to a document is called 'Latent Dirichlet Allocation', which is a generative probabilistic model for collections of discrete data such as text corpora. Essentially, LDA assumes that each document consists of a mixture of latent different topics. Each of the topics has its own weight and its particular multinomial distribution over the vocabulary.

Especially, I think LDA will make a lot of sense in my application because the input document usually is long enough to cover more than one topic. Therefore, the documents that are similar to the input document may have their similarity residing in different topics. It would be helpful for the user to have some labels that render further information about the results and how they are 'similar' to the input document.

Experiments

Experiment data

To evaluation the system, I crawled a set of personal web pages as the experiment data. All the web pages come from the Computer Science(Engineering) departments from University of Wisconsin-Madison, Stanford University, UC Berkeley and etc. The data is chosen particularly within the domain of computer science domain because given the limited data set we have, it would be much more easier if the number of topics can be relatively small and the cosine similarity within the same domain will be more meaningful since the overlapping between different pages is expected to be high.

Cosine similarity calculation

I compiled a collection of multiple pages for one person into a single document and used both BOW and tf.idf to represent the document. As mentioned earlier no clear conclusion of which one is better has been observed. When user provide a document for a user, the cosine similarity will be calculated between this page to all the other pages and a list of top 10 similar web pages are returned.

LDA training and inference

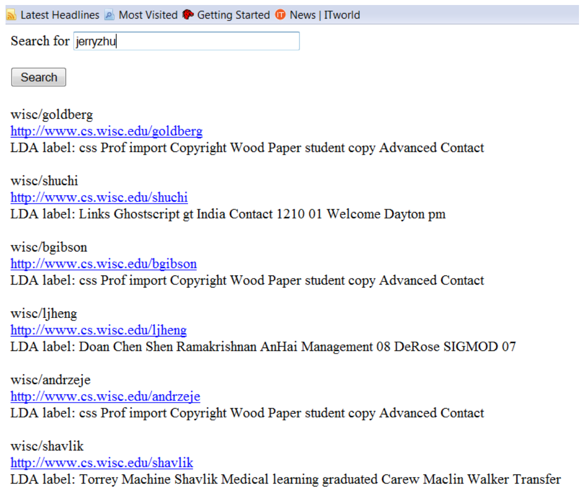
The LDA model is trained using the LDA software package acquired from http://www.cs.princeton.edu/ blei/lda-c/. The package will automatically generate files to indicate the learned topics and the LDA inference given any specific web page in the system.

My first training did not work out well because I failed to get rid of the words that happens too frequently within the corpus. Not surprisingly these words are usually words like it, and, that, and etc., which offer little help in differentiating the topic of a document. Therefore, I pruned the top 150 words that occur the most frequently in the corpus and trained the LDA with 50 different topics. The learned topics turned out to make much more sense than before.

Figure. 1 shows a typical output of the searching results. The web pages are ranked with respect of similarity to the original document.

Conclusion

The experiment shows that the cosine similarity and bag of word vector is well suited for finding the similar documents. The similar pages we find are usually from the student, advisor or co-workers with the owner of the input documents. However, I also noticed that the LDA labels sometimes were not very conclusive. I have tired to trained the model with less frequent words cut off or less latent topics but the results were not obviously improved. I think this was due to the defect of LDA model itself.



Search for jerryzhu

wisc/goldberg
http://www.cs.wisc.edu/goldberg
LDA label: css Prof import Copyright Wood Paper student copy Advanced Contact

wisc/shuchi
http://www.cs.wisc.edu/shuchi
LDA label: Links Ghostscript gt India Contact 1210 01 Welcome Dayton pm

wisc/bgibson
http://www.cs.wisc.edu/bgibson
LDA label: css Prof import Copyright Wood Paper student copy Advanced Contact

wisc/ljheng
http://www.cs.wisc.edu/ljheng
LDA label: Doan Chen Shen Ramakrishnan AnHai Management 08 DeRose SIGMOD 07

wisc/andrzeje
http://www.cs.wisc.edu/andrzeje
LDA label: css Prof import Copyright Wood Paper student copy Advanced Contact

wisc/shavlik
http://www.cs.wisc.edu/shavlik
LDA label: Torrey Machine Shavlik Medical learning graduated Carew Maclin Walker Transfer

Figure 1: A typical search result

References

Jerry Zhu, Lecture notes for CS769, Computer Science Department, University of Wisconsin-Madison, Wisconsin, 2009.

Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer Verlag, 2006.