Information Retrieval Final Project –

Image Search Result Refinement Report

1. Motivation:

Image searching, especially provided by Google, is now a widely used service. Nevertheless, in many cases the retrieved result bias to one of the implicit topics behind the query.

If one searches 'kiss', for instance, most of the images retrieved by Google relate to the band named 'kiss', while it is more likely that the user actually wants to find images of kissing.

In this project, we improve the image searching by increasing the variety of retrieved images.

2. Problem Formulation:

When a user enters a query, our algorithm should return groups of images. The set of involved images is based on retrieved results of Google image search. Each image groups consists a label that concludes the topic of group as well as several images for preview. The way of image grouping should exhibit the following properties: The similarity of two images in the same group should be as small as possible, while the similarity of two images in different groups should be as large as possible.

3. Related Work:

Our problem formulation is somehow similar to search result clustering (SRC).

Many algorithms are proposed to solve SRC. A common method is to treat each search result as a bag of words, then cluster them by some good similarity features and a basic cluster method.

However, since the results of existing Google image search bias heavily, applying search result clustering directly can't improve the diversity of images. The fact implies that we should find the implicit meaning of query from other sources, instead use the image search result only.

4. Methodology:

First, we have about 10 labels and more websites retrieved by each label, total about 100. The text is extracted from HTML and tokenized. We can get the similarity by tokens of each website and also similarity between labels. We want to use this information to classify the labels.

a. Generate Group:

When user enters a query, we will generate about 10 labels, and each of them represents a group, i.e. a different implicit copy. We've tried some methods to generate these labels. A naive method is just to use the ten autocomplete suggestions provided by Google when the query is entered in search bar. Take the query 'apple' as an example, the labels will be:

apple daily, apple store, apple store hk, apple hk, apple ipad, apple tv, apple id, apple news, apple jp, apple line

The second method is search the query on Google. The TF of a word is the number of website whose title or snippet contains the word, and the IDF of the word is given from Wikipedia dictionary: frequency list. We extract those words whose TFIDF is higher than a threshold as labels. To avoid the bias from Google search, we re-search the query and exclude the extracted labels. Then extract others labels based on result websites in the same way. We repeat the procedure until no new label is extract. Take the query 'apple' as an example again, the labels will be: apple, [ago, reviews, os, inc], [valley], [pie]

b. Collect Webpage:

We use a python package called "nltk" (Natural Language Toolkit), to tokenize the tokens from the pure text. All words are transformed to lower case and stopword removed.

Finally, they are all stemmed by Porter Stemmer which is famous in indexing and retrieval. However, these are all for English tokenizing. Chinese tokenizing is so difficult.

For example, the punctuations in Chinese could not be tokenized correctly by nltk, so we replace them to English punctuations first. Then, we recognize the encoding of pure text and transformed to Unicode we just make each word to be a single token. Maybe, we can use bi-gram (haven't done).

c. Analysis Similarity:

In order to find out the similarities between labels, we first calculate the similarities between each document. We use LSI model as the main model for calculation, since it usually has better performance in terms of real world documents.

We use the library "ntlk" to fulfill the implementation again here. After having the result for each document, we sum them up and get the final label-similarity-table.

d. Modify Group:

While 4.1 only consider the overall diversity of the set of labels, it doesn't consider the similarity between two labels. That will produce many image groups related to similar topic. Once we have the similarity between each pair of labels, we can modify our image groups by clustering similar group by any clustering algorithm. However, we eventually disable this part from our project due to the problem of high noise of website contents.

5. Result Webpage:

The most difficult problem is to define "what they really need" for user. We first consider a huge graph with hundreds of clustered nodes and links with similarity value on it. However we quickly realize that it will be totally unreadable for user. So we then redesign the layout of data. Only focus on the labels we get, and the relationship between them. This not only simplified the output graph, by summing up the similarity value in each node, this also improves the accuracy.

We used parallel programming module to accelerate our searching and responding.

6. Discussion:

Initially, we believed this algorithm will work properly, since every step we did seemed reasonable. However, we found out that the presentation sometimes showed ridiculous result.

For example, when we used "lion" as query, we would get some labels, such as "lion king" and "lion king 2". In our opinion, the relationship between these two episodes was fairly high. Nevertheless, the graph showed merely relevance between them.

After studying our code and result for a while, we found out that this weird phenomenon happened only when the top ten related website for each label contained little text for verification. Thus, result of query would be affected by lots of "noise", such as useless links or ads.

To improve performance, one way is to increase the amount of retrieval documents. By doing this, we can eliminate a large sum of noise interference, and get a better result.

7. Future work:

(1) For now, we only use uni-gram tokenizing in Chinese, we can use bi-gram or word segmentation system provided by Sinica. Also we only use VSM to calculate similarities. We should try some other models in order to improve performance.

(2) We are severely limited when just using text data to evaluate the relationship between 2 websites. Considering the diversity of multimedia and those hidden data in hyper-links or picture, we would improve the result with more features adding to our model.

(3) We now rely on Google's autocomplete data, which will cause bias when the labels we get are not food enough. By merging the labels which are highly related, or generating labels by ourselves, this shouldn't bother us anymore.

8. Work Distribution:

簡伯宇: Search result parsing, generate labels.

馮　硯: UI and display layout design, and parallelism.

宋昊恩: Similarity calculation.

余孟桓: Webpage's tokenization.