Development of an Aggregated News Search Engine

Rodrigo Doria Medina and Thibault Roucou
Master Human Media Interaction
University of Twente, Netherlands

January 18, 2012

Abstract

1 Overview of the project

1.1 Description

The aim of the project is to create a search engine doing aggregative search by retrieving texts, images and videos using Twitter to build the ranking algorithm. Due to Twitter API restrictions, we decided to focus on news not older than 7 days. The results will be displayed as a NewsPaper.

1.2 Sources

1.3 Technologies used

1.4 Evaluation methods

2 Building the baseline

2.1 The APIs

2.1.1 Google News

Google News doesn't provide an API anymore. Therefore, we had to find another way to get the results from it. We decided to use the RSS feed and the search system. In order to get information from this RSS feed, we used an API called Rome, which allow to use easily RSS feeds.

A Google News query is of the following form :

"http://news.google.com/news?q=*user_query*&output=rss&hl=en&as_qdr=w"

Explanation of the different parameters :

- q : the query provided by the user
- output : the output format given by GoogleNews, here we choose rss.
- hl : the language of the provided results. We decided to retrieve in priority english results.
- as_qdr : this parameter allow to decide how old can the results be. Here "w" means week so we only retrieve results on week old at worst.

From this query, we get a lot of results and for each result, we extract the title, a summary and a link.

2.1.2 YouTube

2.1.3 Flickr

2.2 Choosing the documents for the baseline

3 The ranking algorithm

3.1 Passive ranking

3.2 Active ranking

3.2.1 The Twitter API

Our active ranking uses twitter to rank documents. We therefore need to retrieve informations from Twitter using the API provided. We use an implementation of this API in Java called Twitter4J which is really easy to use.

We use only the search API which allow to retrieve tweets that are less than 7days old. In order to have a useful sample of tweets, we retrieve 1000 tweets for each query and do some computation on this sample to extract some useful information for the ranking algorithm.

3.2.2 Using Twitter to judge the "interestingness" of documents

In order to define which are the most interesting documents, we use Twitter to define the most "trending" words for the user's query and find these words in the document to increase their ranking score.

Getting "interesting" words from Twitter

We first retrieve a large amount of tweets (1000 in our case) using the search API with the user's query. The second step in then to concat all the tweets retrieved, to remove the words which are in the query, the stopwords, the punctuation and the links. With the "cleaned" corpus, we can count the occurences of each words to determine the top 10 words.

Ranking system

We then give a score to each of the ten words according to their number of occurrences on a scale of 100 :

$$\text{Score for a word} = \frac{\text{Occurrences of the word}}{\text{Total number of occurrences of the 10 first words}} \times 100$$

When all the scores are defined, we look in each document if one of these 10 words are present in the description or in the title. For each word found in the description of a document, we will add to it the corresponding score and if it is in the title, we will add two times this score. We thought that if one of these words was in the title, it was probably a very important document for the query and therefore deserve a high score.

3.2.3 Using Twitter to define the most relevant vertical

One big problem in aggregative search is to define the percentage of each verticals for a query. Indeed, how can we know if a user wants more videos, more images or more text. The user maybe doesn't even know what she wants at the beginning but will see in the results if one vertical is better than an other for her query. To try to solve this problem, we have again used Twitter. Indeed, according to Twitter [1], in 2010, 25% of tweets were containing links and with most of the query we have tried, it was far more. So we have decided to use these links and analyze them to know if they link to an image, a video or something unknown.

Getting the verticals percentage

[1] http://techcrunch.com/2010/09/14/twitter-seeing-90-million-tweets-per-day/

We again retrieve 1000 tweets from the user's query and analyzed the different links leading to a video and define that these links contains these following words : "youtu", "video" or "vimeo". If a link contain these part of words, we then count them as videos. We did the same for images with the following list of words : "img", "pic", "flic", "photo", "instagr", "yfrog". All the remaining links are considered as "unknown". As a lot of links in twitter are shortened, it is impossible to know if they lead to a video or an image in a reasonable time. We therefore decided, after some test, that we could divide the number of unknown link by 5 and imagine they lead to text.

Ranking system

We can then set a score for each vertical :

$$\text{Score for a vertical} = \frac{\text{Occurrences of the vertical}}{\text{Occurence of images} + \text{Occurence of videos} + (\text{Total unknown links})/5} \times 100$$

When all the scores are defined, we had to each videos and images their vertical score. We didn't add anything to the text documents because they are already advantaged by the "interestingness" ranking because they contain more words than for videos or images. This ranking we therefore give a better balance to the results.

4 The evaluation

In order to evaluate the system, we have decided to compare it to a baseline. Like said before, the baseline is a round robin ranking based on the ranking provided by the different APIs.

To determine if a document is relevant or not, we need the users to say it. Thats why we published a test version of our system online. This version was providing 2 sets of results for a query. The first set was the baseline and the second was our ranked results. The user had the possibility to say for each result if it was relevant or not. The results were then sent to us by email and we were able to compare the different precision (P@5, P@10 and P@20)

5 Discussion and Conclusions

5.1 Discussions about the system

5.1.0.1 During the use of Twitter in the algorithm, when finding interesting words, we find in first "obvious" words that complete the query. For example when the user search for "Obama", the first two words being returns will be "president" and "barrack" so results with these words will have a higher score. But it's not giving any information about the news related to "Obama". So it's better if the user enter "president barrack obama" as a query, the results are improved in this case.

5.2 Difficulties encountered

5.2.0.2 Lack of possibilities for Google News

5.2.0.3 Lack of information in a Flickr image

5.2.0.4 Many of the link in twitter are shortened

5.2.0.5 The system is slow