IIT BHU at FIRE 2016 Microblog Track: A Semi-automatic Microblog Retrieval System

Ribhav Soni
Department of Computer Science and Engineering
Indian Institute of Technology (BHU) Varanasi
ribhav.soni.cse13@iitbhu.ac.in

Sukomal Pal
Department of Computer Science and Engineering
Indian Institute of Technology (BHU) Varanasi
spal.cse@iitbhu.ac.in

ABSTRACT

This paper presents our work for the Microblog Track in FIRE 2016. The task involved utilizing microblog data (tweets) to retrieve useful information during times of disasters. In particular, given a set of tweets posted during the Nepal earthquake in 2015, the goal was to judge relevance of each tweet against a set of topics which reflected useful information needs. Our approach made use of manual query formation for searching relevant tweets based on the information required for each topic, after indexing them using Lucene.

1. INTRODUCTION

This paper describes our approach for the Microblog Track in FIRE 2016 [1]. Microblogging sites like Twitter are important sources of real-time information, and thus can be utilized for extracting significant information at times of disasters such as floods, earthquakes, cyclones, etc. The aim of the Microblog track at FIRE 2016 [1] was to develop IR systems to retrieve important information from microblogs posted at the time of disasters. The task involved identifying tweets relevant to the given topics which reflect the information needs at critical times. The topics were provided in a standard TREC format, containing a title, a brief description, and a detailed narrative specifying what type of tweets would be considered relevant to the topic. An example of a topic is:

<top>
<num> Number: FMT4
<title>
WHAT MEDICAL RESOURCES WERE REQUIRED
<desc> Description:
Identify the messages which describe the requirement of some medicine or other medical resources.
<narr> Narrative:
A relevant message must mention the requirement of some medical resource like medicines, medical equipments, supplementary food items, blood, human resources like doctors/staff and resources to build or support medical infrastructure like tents, water filter, power supply, ambulance, etc. Generalized statements without reference to medical resources would not be relevant.
</top>

Three types of runs were considered in the track, based on the amount of manual intervention in different stages such as query formation and document retrieval:

(1) Fully automatic, where no step involves manual intervention
(2) Semi-automatic, where there is some manual intervention but only in the query formation stage
(3) Manual, where there is manual intervention in both the query formation and document retrieval stages

We submitted one run in the semi-automatic category. We used Lucene [2] for indexing, and retrieved relevant tweets for each of the topics by manual query formation. Results show that our system performs reasonably well, given its simplicity, but is outperformed by more complex systems.

The rest of this paper is structured as follows. We describe the training data used in Section 2, the main challenges involved due to the nature of microblogs in Section 3, our approach in Section 4, results and discussions in Section 5, and conclusion and future work in Section 6.

2. DATA

The training data was a collection of about 50,000 tweets posted during the Nepal earthquake in April 2015, along with the associated metadata for each tweet [4].

3. CHALLENGES

Tweets have a stringent word limit, and users often make use of innovative abbreviations which are difficult to handle for retrieval systems. Besides, they are mostly informal, and may involve the use of multiple languages in the same tweet (called code mixing), or even multiple scripts in a tweet. It is also difficult to make sense of emoticons, especially innovative ones made up by users.

4. OUR APPROACH

Our run was in the semi-automatic category, which includes systems with manual intervention in the query formation stage. We used Apache Lucene, an open-source textual search engine library, for indexing the available tweets.

For retrieval, we used Lucene's search facility with manual search queries, which were formed on the basis of requirements for each topic. The search queries that we used for each of the topics are given in Table 1.

Lucene first selects the documents to be scored based on Boolean logic from the query specification, and then ranks them via the specified retrieval model [3]. We made use of the default similarity model, which computes scores using a combination of the Vector Space Model (VSM) and probabilistic models such as Okapi BM25.

Table 1: Query strings for each topic

TOPIC	QUERY STRING
What Resources Were Available	"food water clothes volunteers power charge available^2"
What Resources Were Required	"food water clothes volunteers power charge available^2"
What Medical Resources Were Available	"doctor medicine ambulance blood milk baby food nurse water tent power available^2"
What Medical Resources Were Required	"doctor medicine ambulance food blood nurse water tent power require^2 need^2"
What Were The Requirements / Availability Of Resources At Specific Locations	"location^3 place^2 town^2 kathmandu village available need"
What Were The Activities Of Various NGOs / Government Organizations	"NGO^5 government^4 work^3"
What Infrastructure Damage And Restoration Were Being Reported	"road railway house damage place town"

Table 2: Results of our run

RUN ID	Precision@20	Recall@1000	MAP@1000	OverallMAP
iitbhu_fmt16_1	0.3214	0.2581	0.0670	0.0827

For tokenization, we used the StandardAnalyzer, which creates tokens using the Word Break rules from the Unicode Text Segmentation algorithm specified in [5]. It is capable of handling names and email address, lowercases each token, and removes stopwords and punctuations.

Lucene ranks the returned search results (retrieved tweets) based on the degree of relevance using its scoring algorithms, and returns the ranked list as well as individual scores. Since the task involved tagging all relevant tweets, all the returned tweets were marked relevant. In addition, the returned scores were normalized to the range of [0, 1] for assigning relevance scores to each returned topic-tweet pair as per the run submission instructions.

5. RESULTS AND DISCUSSION

The results of our run based on several metrics are given in Table 2.

Our system performed reasonably well in the semi-automatic category. However, it was outperformed in the task by more elaborate systems.

6. CONCLUSION AND FUTURE WORK

Our system was overly simplistic, and offers much scope for improvement by making use of state of the art IR techniques.

Some approaches to improve on the system include better preprocessing of tweets (which is very essential for microblog retrieval tasks, given the challenges with the nature of microblogs that we described in Section 2), taking the quality of tweets into account by considering the prior tweets by the author, query expansion approaches using external information (like Google search results, or Wikipedia corpus), and using pseudo-relevance feedback techniques to better tune relevant search results. Also, Lucene was too liberal in returning tweets with a very low score as part of search results, most of which were found to be non-relevant. Thus, it is important to further refine the relevant results returned by the search query by setting a threshold so that only those tweets are considered relevant to a query which have a reasonably large similarity score.

7. REFERENCES

[1] S. Ghosh and K. Ghosh. Overview of the FIRE 2016 Microblog track: Information Extraction from Microblogs Posted during Disasters. In *Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation, Kolkata, India, December 7-10, 2016*, CEUR Workshop Proceedings. CEUR-WS.org, 2016.

[2] Apache Lucene. https://lucene.apache.org/.

[3] org.apache.lucene.search (Lucene 6.2.1 API). https://lucene.apache.org/core/6_2_1/core/org/apache/lucene/search/package-summary.html#package.description.

[4] Tweets – Twitter Developers. https://dev.twitter.com/overview/api/tweets.

[5] Unicode Standard Annex #29: Unicode Text Segmentation. http://unicode.org/reports/tr29/.