REAL-TIME SUMMARIZATION OF SOCIAL-MEDIA INFORMATION STREAMS

Web Search 2017/2018

Submission should be done by email until 23h59 of the 20th of December.

This project guide is largely based on the TREC Real-Time Summarization Task.
A good advice is for you to read this report first.

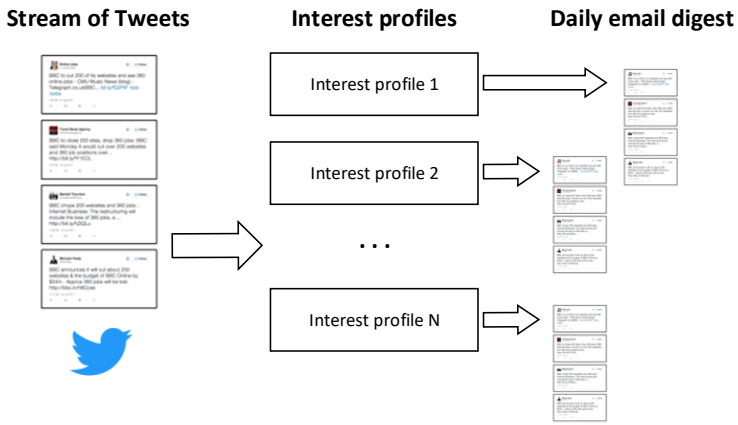
INTRODUCTION

There is emerging interest in systems that automatically monitor streams of social media posts such as Twitter to keep users up to date on topics they care about. We might think of these topics as "interest profiles", specifying the user's prospective information needs. For example, the user might be interested in poll results for the 2016 U.S. presidential elections and wishes to be notified whenever new results are published.

The simplest method for disseminating updates consist of a daily email digest. In this scenario, users may wish to receive a daily email digest that summarizes "what happened" that day with respect to the interest profiles. At a high level, these results should be relevant and novel; timeliness is not particularly important, if the tweets were all posted on the previous day.

SETUP

The basic setup is described by the following figure. You will be provided with a stream of Tweets (documents and timestamp), and a list of "interest profiles" representing users' information needs (similar to topics in ad hoc retrieval).



Your system should be able to parse the "interest profiles" and create information filters to create daily email digests with the 10 Tweets that best summarize the event on each day. The data workflow is illustrated above: the daily email digests will be evaluated according to its relevance and novelty.

STREAM OF TWITTER DATA

The evaluation data cover the days from August 2, 2016 00:00:00 UTC to August 11, 2016 23:59:59 UTC. You will be provided with a sample of Tweets from this period. The relevance judgments are also provided together with the dataset.

INTEREST PROFILES

An interest profile is a JSON-formatted structure that contains the same information as a "traditional" ad hoc topic. The "title" contains a short description of the information need, similar to what users would type into a search engine. The "description" and "narrative" are sentence- and paragraph-long elaborations of the information need, respectively.

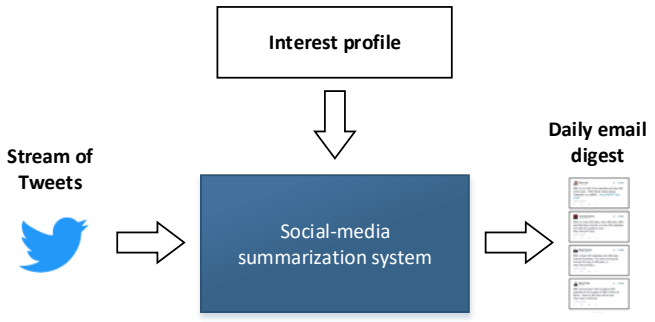
```
{ "topid" : "MB246",
"title" : "Greek international debt crisis",
"description" : "Find information related to the crisis surrounding the Greek debt
      to international creditors, and the consequences of their possible withdrawal
      from the European Union.",
"narrative" : "Given the continuing crisis over the Greek debt to international
      creditors, such as the International Monetary Fund (IMF), European Central
      Bank (ECB), and the European Commission, the user is interested in information
      on how this debt is being handled, including the possible withdrawal of Greece
      from the euro zone, and the consequences of such a move."
}
```

The interest profiles should be divided into train/test as follows:

- **Train interest profiles:** use the even queries (e.g., RTS2, RTS4, RTS6, …) to tune and develop your system.
- **Test interest profiles:** evaluate your system on the odd queries (RTS1, RTS3, RTS5, …)

OUTPUT: DAILY DIGEST

At the end of each day, and for each interest profile, the system should select the 10 most relevant tweets and submit them as a daily email digest. You must build the index incrementally, considering only the tweets that were published until that day.



Your system must identify a batch of up to 100 ranked tweets per day per interest profile. All tweets from 00:00:00 to 23:59:59 are valid candidates for a particular day. It is expected that systems will compute the results in a relatively short amount of time after the day ends (e.g., at most a few hours), but this constraint will not be enforced. However, participants are not allowed to use future evidence in generating the ranked lists (e.g., term statistics from the following day).

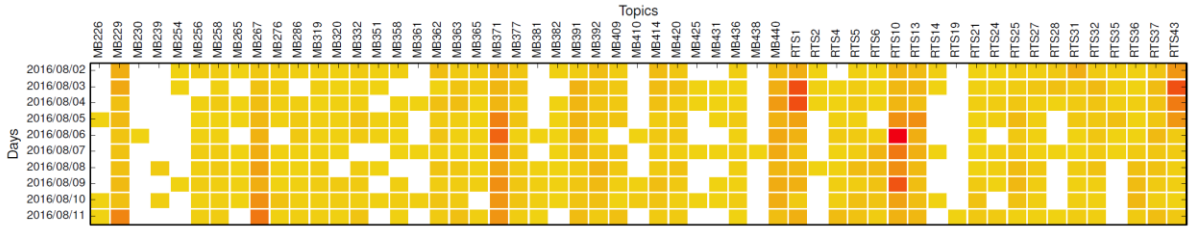


FIGURE 1. HEATMAP OF THE DISTRIBUTION OF ALL RELEVANT AND HIGHLY-RELEVANT TWEETS: INTEREST PROFILES IN COLUMNS, DAYS OF THE EVALUATION IN ROWS.

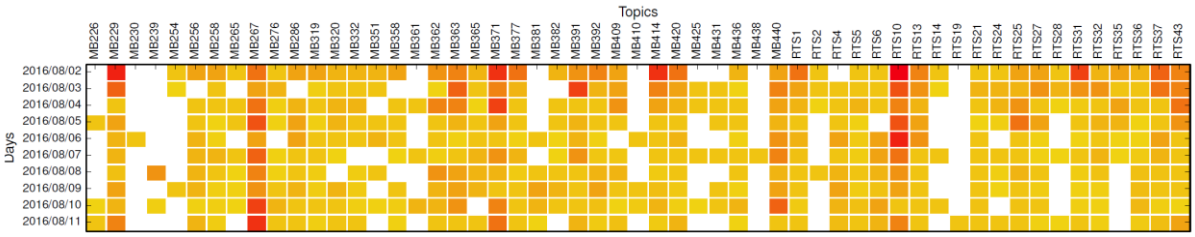


FIGURE 2. HEATMAP OF THE DISTRIBUTION OF THE FIRST TWEET IN EACH SEMANTIC CLUSTER: INTEREST PROFILES IN COLUMNS, DAYS OF THE EVALUATION IN ROWS.

The runs should be formatted as a plain text file, where each line has the following fields:

```
YYYYMMDD topic_id Q0 tweet_id rank score runtag
```

Basically, this is just the standard TREC format prepended with a date in format YYYYMMDD indicating the date the result was generated. The rank field is the rank of the result, starting with one; score is it's score. This format allows us to easily manipulate the runs and pass to existing scoring scripts to compute nDCG, etc. on a per day basis. Please make sure that rank and score are consistent, i.e., rank 1 has the highest score, rank 2 has the second highest score, etc. Otherwise, scoring ties will be broken arbitrarily.

IMPROVED OUTPUT: DIVERSE DAILY DIGESTS

Many search results talk about the same information facts. In principle, given the short-length of Tweets, one can assume that the same information fact is be present in several different Tweets, and each Tweet only talks about one fact. Grouping search results by their content, enables the computation of equally relevant documents, but more informative results.

Pos 1: D4, Info Fact A	⟹	Pos 1: D4, Info Fact A
Pos 2: D9, Info Fact A		Pos 2: D1, Info Fact C
Pos 3: D1, Info Fact C		Pos 3: D5, Info Fact B
Pos 4: D5, Info Fact B		Pos 4: D8, Info Fact D
Pos 5: D11, Info Fact C		…
Pos 6: D7, Info Fact B		
…		

Figure 1 and Figure 2 illustrate the importance of detecting Tweets from the same cluster. In Figure 1 you see the heatmap for all relevant Tweets, while in Figure 2, only the first Tweet from a given cluster is considered relevant (all sub-sequent Tweets are considered not relevant).

Detecting duplicate information is an important task for every information search engine. There are two main methods to achieve this task:

- **K-Means** is clustering algorithm that organize documents into groups of documents that discuss the same information topics or fact. However, in most clustering algorithms, the creation of a cluster (or document group) need to have a large support from data.
- **MinHashing** is a Locality Sensitive Hashing method that map similar documents to the same hash code. Using methods like MinHashing one can detect near-duplicates, even when there are as few duplicates as two, which is clearly insufficient to create a cluster.

You should integrate and evaluate both algorithms. The comparison should illustrate the advantages of each method.

EVALUATION

To evaluate your system runs, we compute the nDCG@10 score for each day for each interest profile, and then average across them. **Normalized Discount Cumulative Gain (nDCG)** for an interest profile on a particular day is defined as follows:

$$nDCG = \frac{1}{best_{DCG}} \sum_{i=1}^{n} \frac{2^{r(i)} - 1}{\log_2(i+1)}$$

where $best_{DCG}$ is the maximum possible gain (given the ten tweet per day limit). Tweets are judged as:

- Not relevant tweets receive a gain of 0.
- Relevant tweets receive a gain of 0.5.
- Highly-relevant tweets receive a gain of 1.0

Once a tweet from a cluster is retrieved, all other tweets from the same cluster automatically become not relevant. This penalizes systems for returning redundant information.

Your results will be evaluated with an adapted metric:

- **nDCG@10-0**: for a silent day, all systems receive a gain of zero no matter what they do. For days when there are relevant Tweets, the metric falls back to the regular **α-nDCG@10** metric.

You are provided with the Python script rts2016-batchB-eval.py to evaluate your runs. The script requires the following parameters:

```
rts2016-batchB-eval.py -q <qrels file> -c <clusters> -t
<tweetsdayepoch> -r <your file>
```

PROJECT IMPLEMENTATION

Your implementation can be in Java or Python. The search index can be supported by either Lucene, SOLR or Whoosh search engines, although you are advised to follow up on your previous checkpoint implementations. Your project and report must answer these requirements:

1. **Daily digest:**

    a. Implement an incremental indexing where you store a separate index for each day. Each index contains the Tweets of that day and all previous days.

    b. **Baseline 1 [5%]:** Build the daily email digest as a retrieval task with the LMD retrieval model. Calibrate the μ parameter.

    c. **Baseline 2 [5%]:** Build the daily email digest as a retrieval task with the BM25 retrieval model. Calibrate the b and k_1 parameters.

    d. **Baseline 3 [20%]:** Use pseudo-relevance feedback to expand the initial query and improve the daily summaries. Justify the selection of the retrieval models. Calibrate the parameters.

    Evaluation [5%]: Compare and analyse the performance of the different retrieval models.

2. **Diverse daily digests:**

    a. **Baseline 4 [15%]**: To address the problem of information diversity, i.e. groups of tweets that are too similar in content, implement a strategy with KMeans to remove redundant tweets from your search results.

    b. **Baseline 5 [15%]**: Following the MinHash algorithm, implement a near-duplicate detection method with the Jaccard coefficient to remove redundant tweets from your search results.

    Evaluation [15%]: Do a success and failure analysis of your results. Identify the elements in your system that contribute positively and negatively. Discuss your insights.

3. **Rank fusion:**

    a. **Combining multiple fields [10%]:** using the reciprocal rank fusion method, implement a search model to combine multiple ranking models.

    Evaluation [10%]: Compare and analyse the performance of your system with multiple fields.

4. **Discussion of possible improvements [5%]**

All the files you need are available in this link. The tweets in this file, are a subset of the free Twitter 1% sample. It contains only the Tweets retrieved by all 2016 competitor systems.

REPORT

Your project report is limited to 8 pages, it must follow the ACM conference template (the Word document *ACM_SigConf.docx* or the Latex file *sample_sigconf.tex*), and it must be structured as follows:

1. Introduction
2. Methods and algorithms
3. Implementation: What are your ideas? What makes your project unique?
4. Evaluation
    a. Dataset description
    b. Baselines
    c. Results and discussion
5. Possible improvements
6. References

NOTES AND ADVICES

- You are advised to re-use the methods that you implemented in the checkpoints.
- Your experimental setup and protocol cannot use future evidence.
- Your experimental setup and protocol cannot use relevance judgments of the same day of the daily email digest that you are computing.
- However, you can, if you wish to, use the relevance judgments of the previous days as relevance feedback.

Please, feel free to discuss these ideas (as well as other ideas you may have). A good advice before you start your implementation, can save you a lot of time.