Real-time summarization of social-media information streams

Web Search 2016/2017

This project guide is largely based on the TREC Real-Time Summarization Task.
The reports of all participant systems are available online.

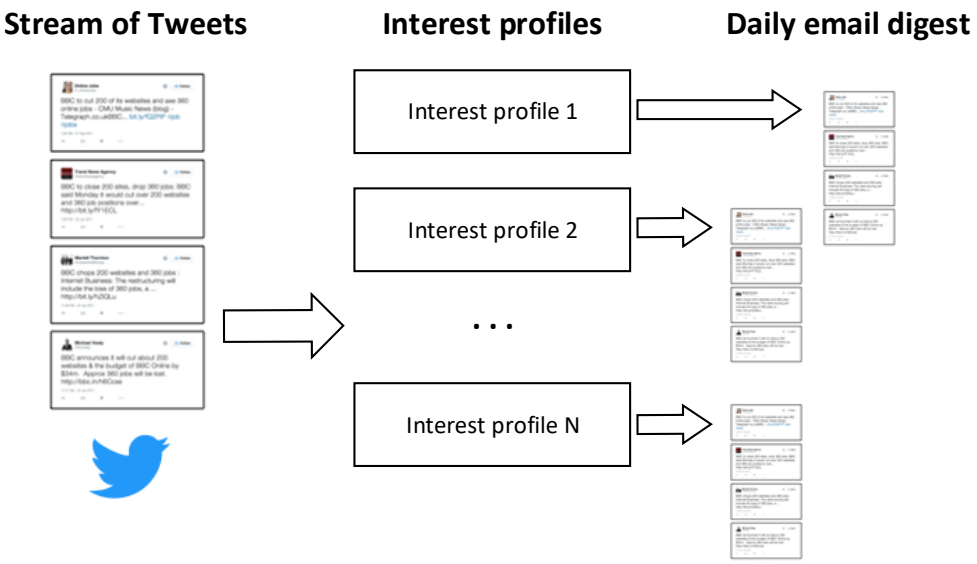
Introduction

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, provided that the tweets were all posted on the previous day.

Experimental 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.

Twitter stream dataset

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.

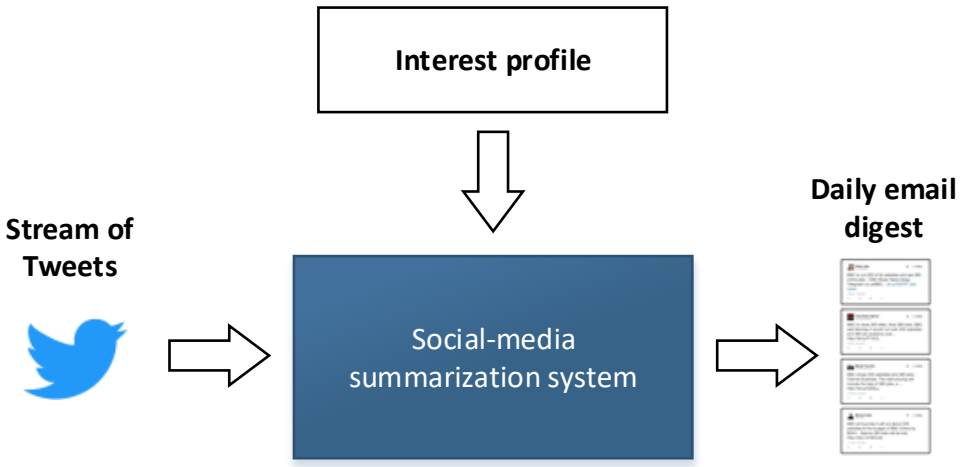
Interest Profiles

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."
}
```

Task

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.



Evaluation metrics

To evaluate your system runs, we compute the nDCG@10 score for each day for each interest profile, and then average across them. We will measure your results in two ways:

- **nDCG@10-1**: on a "silent day" (same definition as above), the system receives a score of one (i.e., perfect score) if it does not return any tweets (i.e., does not submit a ranked list for that interest profile), or zero otherwise
- **nDCG@10-0**: for a silent day, all systems receive a gain of zero no matter what they do.

nDCG@10-1 rewards a system for recognizing when there are no relevant tweets, whereas for nDCG@10-0, it never hurts to return tweets, even if there are no relevant tweets for that day.

You are also provided with a Python script to evaluate your runs. Please, read the corresponding documentation.

Discussion

1. Consider an implementation where you index the full collection at the same time, and to build the daily email digest, you filter the Tweets by their dates. Discuss this solution.
2. Describe how you indexed incrementally, the stream of Tweets.
3. What types of Web data does your system process and consider to create the daily email digests?
4. Build the daily email digest as a retrieval task with the three studied retrieval models (tf-idf, BM25 and LMD. Compare the performance of the different models.
5. How does your solution consider the temporal evidence of the data? Compare the performance of your system with and without temporal evidence.
6. Implement a strategy to remove redundant tweets (i.e., tweets that are too similar in content). Compare the performance of your system with and without the removal of redundant information.

Qualitative evaluation (optional)

1. Create one interest profile covering an event in Lisbon (or other place in Portugal).
2. Create a Web page to visualize the event summaries of each day.
3. Implement a strategy to summarize the full set of days by selecting the highlights of each day.
4. Create a Web page to visualize the summary of the whole event (covering the full set of days).

Notes and advices

- All the files you need are available on CLIP.
- You are allowed to re-use the methods that you implemented in the first project.
- You can compute document similarity metrics with cosine-TF-IDF.
- 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.
- You can use clustering algorithms to remove redundant information.
- You can crawl other Web sources (e.g. Wikipedia) to mine further sources of evidence.

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.

Implementation, report and grading

Your implementation can be in either Java or Python. The search index can be supported by either Lucene or SOLR.

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. Off-the shelf algorithms
3. Implementation: What are your ideas? What makes your project unique?
4. Evaluation
    a. Dataset description
    b. Baselines
    c. Results
5. Critical discussion
6. References

The project grading is divided into four items:

- Implementation originality: 25%
- Code "cleanness": 25%
- Critical discussion: 25%
- Report: 25%