Blog to Microblog

Report Checkpoint 1 - Project CSCE - 670

Vandana Bachani
March 8th, 2012

Blog2Microblog is a tool that can summarize articles, based on the social context of the article. It will be developed as a simple Web UI. Given a link for an article (from the test collection), it will generate a summary (5 important sentences from the article) and a tweet text. The tool would also give the statistics and reasoning as to why it feels the sentences chosen are important (the support). As mentioned in the proposal, I look at the problem from the perspective of a recommender system which can recommend the 5 most important (popular) sentences in the article/blog.

1. Data Collection

The data for the tool comprises of:

1. Links of blogs or articles from mined tweets
2. The whole text of blogs or articles
3. Related Tweets and Comments
4. Authors' Social Features, Authors' Followers Recent Activity

I selected 8 popular twitter blogs from different fields such as technology, news, nature, fun, etc. in terms of number of followers and follower activity. (May add a few more for diversity reasons)

[engadget, techcrunch, nytimes, mashable, businessinsider, fakingnews, espn, treehugger, huffingtonpost]

Data is collected as follows:

Step 1: Links of blogs or articles

I use the twitter search api to get the recent tweets by the above mentioned twitter handles, which contain urls.

```
query = 'from:techcrunch'  #example
http://search.twitter.com/search.json?rpp=100&q={0}&page={1}&include_entities=true&result_type=mixed'.format(urllib.quote_plus(query), k)
```

The following table shows the number of blog links per handle that were collected.

Twitter handle	No. of Blog Links
businessinsider	639
mashable	447
huffingtonpost	415
engadget	371
techcrunch	320
treehugger	121
espn	50
fakingnews	11

Step 2: The whole text of the blogs

The link tweets collected in Step 1 are used to crawl and get the content of the blogs. "nytimes" provided its api which was used to get the content, for the rest of the handles had to make http requests to get the html content. http://longurl.org/ api service was used to expand the short twitter url for "nytimes".

"BeautifulSoup" library was used to parse the html and extract the content from the blogs. The comments were also parsed and saved in mongo db along with the blog content, if any were returned as part of the blog html. The complete html was also stored for later use in compressed tar.gz files.

Step 3: Related Tweets and Comments

The link tweets data collected in Step 1 is used to get tweets related to the blog/article, using the twitter search api.

The heuristics implied to collect "related" tweets are: tweet similarity to title, no retweets, replies to the blog link tweet and tweets mentioning hashtag as the blogger (eg. #techcrunch).

"nltk stopwords corpus" was used to remove the stop words to get article topic similar tweets.

Relevant tweets were stored in mongo db.

```
#example queries and logic – search api (same as in Step 1)
query = 'to:techcrunch' and filter by twt['in_reply_to_status_id_str'] = link_tweet['id_str']
query = '#techcrunch' and filter by twt['in_reply_to_status_id_str'] = link_tweet['id_str']
query = 'bp plantiffs settlement gulf  oil spill case' and filter all retweets
```

For initial data analysis and understanding the user reply pattern on blogs, 1336 blogs were successfully parsed (content retrieved) and 23305 related tweets have been collected.

Some graphs to assess blog popularity and tweets per blog:

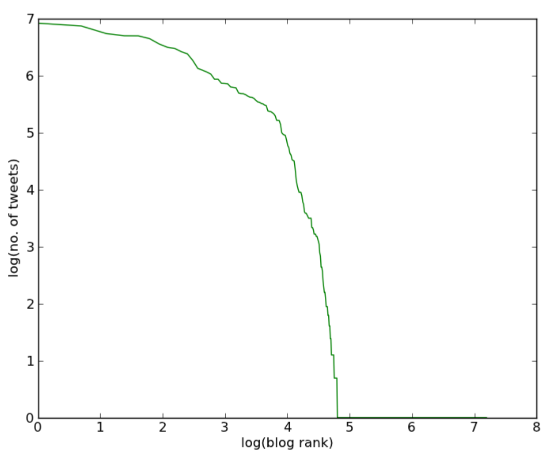


Figure 1

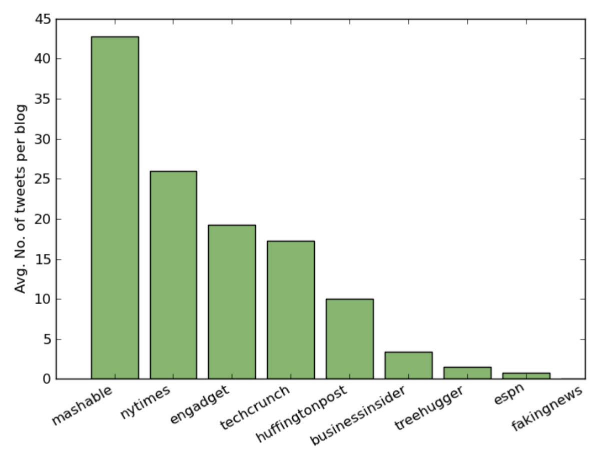


Figure 2

The tweets distribution by blog rank in the collected sample does not follow power law strictly as can be seen in Figure 1 (or maybe the corpus is small). These graphs give a starting point in terms of which blogs to start working with as more users tweet on certain blogs. I still need to do an analysis on the comments in the blogs. (Not all blogs return comments as part of the html).

Step 4: Authors' Social Features, Authors' Followers Recent Activity

Authors' social features – manual (already know which blogger is topic expert in what field)

Authors' Followers Recent Activity – need to use twitter follower rest api to get followers and then recent tweets of those ids using the users/lookup api. We filter useful tweets using blog content data from Step 2. The noisiest dataset (w.r.t. what I am looking for) which requires lot of filtering to get meaningful recent tweets or their weight would be 0 in the ratings learner (next section).

```
list = http://api.twitter.com/1/followers/ids.json?cursor=-1&screen_name=techcrunch
http://api.twitter.com/1/users/lookup.json?user_id=list&include_entitites=true
```

2. System Design and Architecture

System Design

Blog2Microblog is a different kind of recommender system in which the concept of users and items is different from the traditional recommender. I am developing this tool as a content-based recommendation system with dynamic utility matrix and machine learned ratings.

The utility matrix for this recommender system is small enough to fit in memory (unless we have an article which runs into pages and pages and/or there are millions of users who commented on it).

The Utility Matrix comprises of:

1. Items: The sentences in the article.
2. Users: The twitter users and blog commenters, who comment on the article and 3 hypothetical users which represent the other features of the article (refer the diagram below). The 3 hypothetical users are the author, the content of the blog and an aggregate rater. These hypothetical users represent the minimalistic social and document context of the article, which is present when the article is generated and no users have commented on the article.
3. User Ratings: The users' comments are symbolic of the ratings for the various items (sentences) of the article. Objective numerical ratings are calculated for each item as a weighted sum of some features (described below) of the comment with respect to the article. For the hypothetical users also the ratings are a weighted sum of different set of features (described below).
4. Given a document as a set of sentences (labeled as important or unimportant on a scale of 0 to 1 or 1 to 5 (yet to decide)), a list of user comments, the utility matrix is filled with ratings which are a weighted combination of features and the weights of the features need to be learned from a training set. The items which are not assigned comment user ratings (no comments from any users), are given default minimum ratings to fill the matrix.

Once the utility matrix is constructed i.e. ratings assigned based on the learned feature weights, the system can recommend 5 most important/popular sentences from the document.

The ratings and features defining the ratings:

Ratings are calculated as a function of various features of different users with the help of the weights learned as part of training the system.

1. Author ratings:

   The features which define the ratings of the author, include, the recent activity of authors' followers and whether author is an expert in the topic for that particular item.

   The recent activity of authors' followers is measured as: recent tweet by authors' followers and the cosine similarity of those tweets with the items.

   Example: @mashable publishes a blog about Samsung Galaxy Tab being launched and its features. Compare with recent tweets by followers of @mashable who may be interested in this article (cosine similarity of high tf-idf terms of recent tweets of followers) and come up with an interest score.

   Ra = wa0 + wa1(interest score) + wa2(expert score)
2. Content ratings:

The features which define the ratings of the Content include, the sentence position in the document, the similarity of sentence with the topic and the title, etc. (will be extended to include the good content features as have been studied by previous summarization papers).

Rc = wc0 + wc1(sentence position) + wc2(similarity with topic) + ….

3. User ratings:

   The features which define the users' ratings, include, the similarity of user tweet with the item, sentiment of the user tweet as compared to the item, tweet similarity w.r.t. hashtags, etc.

   Ru = wu0 + wu1(tweet similarity with item) + wu2(user sentiment wrt item) + wu3(hashtag similarity) + …

4. Aggregate rating:

   The feature is the number of tweets supporting the item.

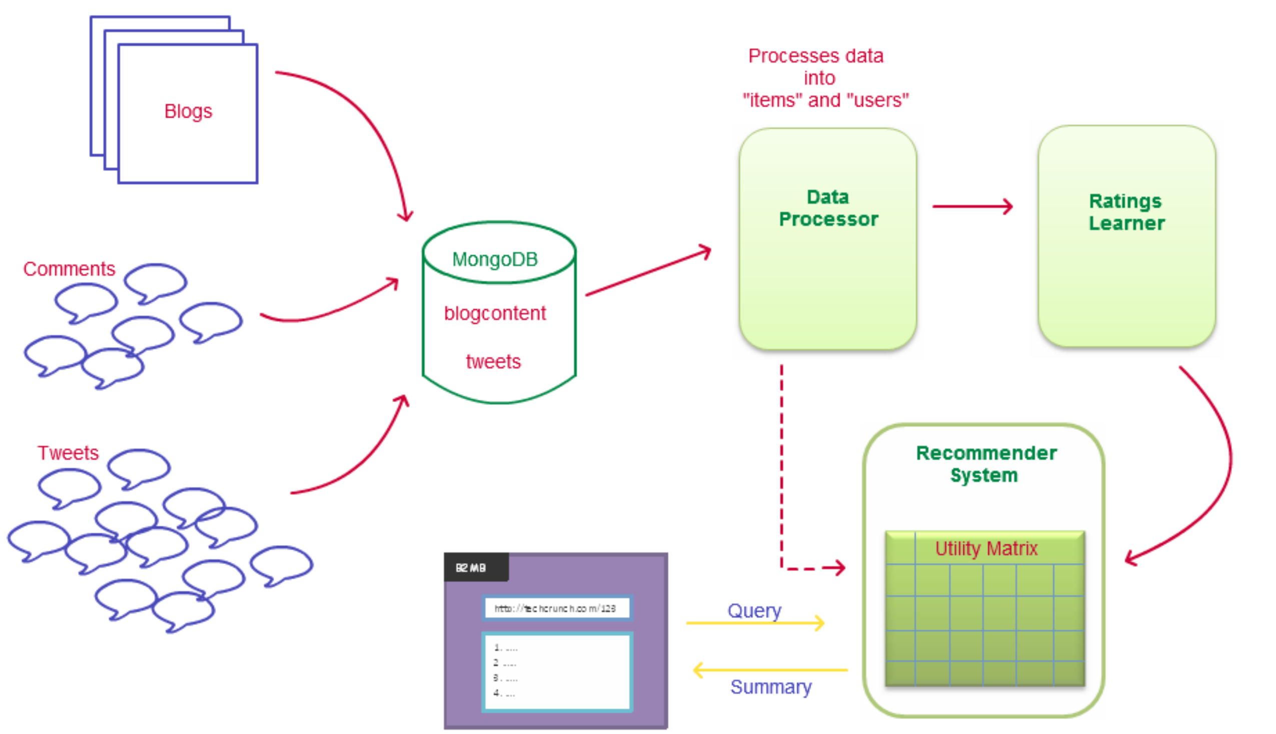
   Rg = wg1(num of tweets about item).

The Utility Matrix:

Users	Item1	Item2	Item3	Item4	Item5	Item6	Item7	Item8	Item9…
Author	Ra1	Ra2	Ra3	..	..	..	..	..	..
Document Content	Rc1	Rc2	Rc3	..	..	..	..	..	..
User1	Ru1			Ru4					Ru9
User2	Ru1		Ru3						
User3									
…									
Aggregate	2	0	2	1	0	0	0	0	1

The above weights are what need to be learned with the help of the training data.
(Might keep the ratings binary or scale of 5 for simplicity). Will use linear regression based model to learn the weights of the features.

Architectural Diagram



3. End-to-End Example

Training:

The training part of the system requires learning the weights for various features as mentioned in last section. For the following example we assume the weights have been learned:

Matrix:

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18
tc	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
content	4	4	4	2	2	2	3	1.5	1	1	1	1	1.5	2	2	2	4	4
james	2	3	0	0	2	3	0	0	0	1	0	1	0	0	0	0	0	1
tom	0	1	3	0	0	0	4	0	0	0	0	0	0	0	0	0	0	3

Summary:

1. A high-ranking digital music executive told The Music Void that Google Music is losing users week after week, despite its preferred access to over 200 million Android installs.
2. Seems its lack of marketing, the missing Warner deal, and competition from iTunes Match and Spotify are taking their toll.
3. It has plenty of ways to promote it but doesn't. It released a mobile web app but nothing native for iOS.
4. If Google Music ever took off, you know that every time their contracts need renegotiating, the labels would reach deep into those deep, search ad-lined pockets.