Towards Identifying Personalized Twitter Trending Topics using the Twitter Client RSS Feeds

Jinan Fiaidhi, Sabah Mohammed
Department of Computer Science
Lakehead University
Thunder Bay, Ontario P7B 5E1, Canada
{jfiaidhi,mohammed}@lakeheadu.ca

Aminul Islam
Department of Computer Science
Lakehead University
Thunder Bay, Ontario P7B 5E1, Canada
maislam@lakeheadu.ca

***Abstract*—We are currently witnessing an information explosion with aid of many micro-blogging toolkits like the Twitter. Although Twitter provides a list of most popular topics people tweet about known as Trending Topics in real time, it is often hard to understand what these trending topics are about where most of these trending topics are far away from the personal preferences of the twitter user. In this article, we pay attention to the issue of personalizing the search for trending topics via enabling the twitter user to provide RSS feeds that include the personal preferences along with a twitter client that can filter personalized tweets and trending topics according to a sound algorithm for capturing the trending information. The algorithms used are the Latent Dirichlet allocation (LDA) along with the Levenshtein Distance. Our experimentations show that the developed prototype for personalized trending topics (T3C) finds more interesting trending topics that match the Twitter user list of preferences than traditional techniques without RSS personalization.**

***Index Terms*—component; Trending topics; Twitter Streaming; Classification, ADL**

I. INTRODUCTION

Twitter is a popular social networking service with over 100 million users. Twitter monitors the millions and billions of 140-character bits of wisdom that travel the Twitter verse and lists out the top 10 hottest trends (also known as “trending topics”) [1]. With such social networking, streams have become the main source of information for sharing and analyzing information as it comes into the system. Streams are central concepts in most momentous Twitter applications. Streams become so important that they even replaces search engines as a starting point of Web browsing – now a typical Web session consists in reading Twitter streams and following links found in these streams instead of starting with Web search. One of the central applications of using streams with Twitter is mine in real-time trending topics. To develop such application one need to use one of the three Twitter Application Programming Interfaces (APIs). The first API is the REpresentational State Transfer (REST) which covers the basic Twitter functions (e.g. send direct messages, retweets, manipulate your lists). The second is the Twitter search API which can do everything that the Twitter Advanced Search can do. The third API is the streaming API which give developers low latency access to Twitter's global stream of Tweet data. In particular the streaming API gives the developer the ability to create a long-standing connection to Twitter that receives “push” updates when new tweets matching certain criteria arrive, obviating the need to constantly poll for updates. For this reason the use of the streaming APIs becomes more common practice related to twitter applications like finding trending topics. In such approach the user subscribes to followers (e.g. FriendFeed) and read the stream made up of posts from the followers. However, the problem with this approach is that there is always a compromise with the number of followers that the user would like to read and the amount of information he/she is able to consume. Twitter users share variety of comments regarding a wide range of topics. Some researchers recommended a streaming approach that identifies interesting tweets based on their density, negativity, trending and influence characteristics [2, 3]. However, mining this content to define user interests is a challenge that requires an effective solution. Certainly identifying a personalized stream that contains only a moderate number of posts that are potentially interesting for the user can be used for the customization and personalization of a variety of commercial and non-commercial applications like product marketing and recommendation. Recently Twitter introduced local trending topics that contribute to the solution of this problem through improving the *Discover Tab* to show what users in your geography are tweeting about. But this service fail short in providing many personalized trending like displaying trends only from those you follow or whom they follow. There are many current other attempts to fill this gap like the Cadmus[1] and KeyTweet[2], however, there is comprehensive solution that can provide wider range of personalization venues for the Twitter users. This article introduces our investigation on developing personalized trending topics over stream of tweets.

II. Related Reseach

Research on topic identification within a textual data is either related to information retrieval, data mining or a hybrid of both. Information retrieval research provides searching techniques that can identify the main concepts in a given text based on structural elements available within the provided text (e.g. by identifying noun phrases as good topic markers [5]). This is a multi-stage process that starts by identifying key concepts within a document, then grouping these to find topics, and finally mapping the topics back to documents and using the mapping to find higher-level groupings. Information retrieval research utilizes computational linguists and natural language techniques to predict important terms in document using methods like coreference, anaphora resolution or discourse center [6]. However, using linguistic techniques in identifying important terms do not necessarily correspond to the subject or the theme. Predicting important terms involves numerical weighting of terms in document. Terms with top weights are judged important and representative of document. In this direction terms extraction methods like TF-IDF [7] (term frequency-inverse document frequency (TF–IDF) generally extracts from a text keywords which represent topics within the text). However, TF-IDF does not conduct segmentation). A segmentation method (e.g., TextTiling [8]) generally segments a text into blocks (paragraphs) in accord with topic changes within the text, but it does not identify (or label) by itself the topics discussed in each of the blocks. While both techniques (i.e. TF-IDF and Segmentation) have some appealing features—notably in its basic identification of sets of words that are discriminative for documents in the collection—these approaches also provides a relatively small amount of reduction in description length and reveals little in the way of inter- or intra-document statistical structure. To address these shortcomings, IR researchers have proposed several other dimensionality reduction techniques and topic identification techniques (e.g. LSI (latent semantic indexing), LDA (Latent Dirichlet allocation)) [9]. On the other hand, data mining tries to analyze text and predict frequent itemsets, or groups of named entities that commonly appear together from a training dataset and use these associations to predict topics in future given documents [10]. This approach assumes a previously available datasets and it not suitable for streaming and dynamically changing topics as the one associated with Twitter. For this reason we consider this approach is out of the scope of this article.

III. Developing a Streaming Client for Identifying Trending Topics

It is a simple task to start developing a Twitter streaming client especially with the availability of variety of Twitter streaming APIs (e.g. Twitter4J[3], JavaTwitter[4], JTwitter[5]). However, modifying this client to search for trending topics and adapting to the user preferences is another issue that can add higher programming complexities. The advantage of using trending topics is to reduce messaging overload that each active user receives each day. Without classifying the incoming tweets users are forced the Twitter to march through a chronologically-ordered morass to find tweets of interest. By finding personalized trending topics and grouping tweets according to coherently clustered trending topics for more directed exploration will simplify searching and identifying tweets of interest. In this section we are presenting a Twitter client that enables the client to group tweets according to the user preferences into topics mentioned explicitly or implicitly, which users can then be browsed for items of interest. To implement this topic clustering, we have developed a revised LDA (Latent Dirichlet allocation) algorithm for discovering trending topics. Figure 1 illustrates the structure of our Trending Topics Twitter Client (T3C).

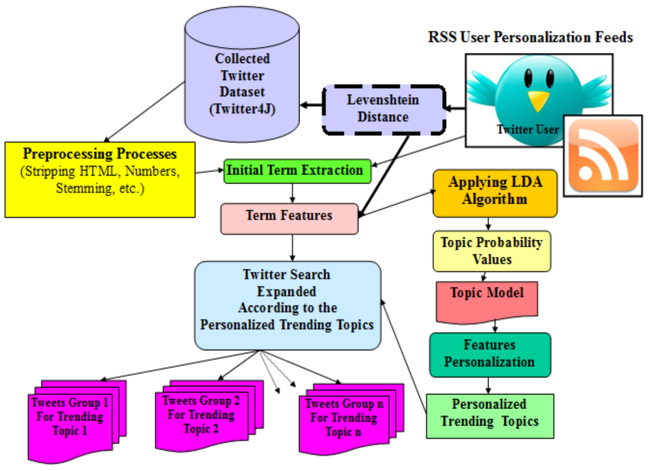

Collected Twitter Dataset (Twitter4J)
RSS User Personalization Feeds
Levenshtein Distance
Twitter User
Preprocessing Processes (Stripping HTML, Numbers, Stemming, etc.)
Initial Term Extraction
Applying LDA Algorithm
Term Features
Topic Probability Values
Twitter Search Expanded According to the Personalized Trending Topics
Topic Model
Features Personalization
Tweets Group 1 For Trending Topic 1
Tweets Group 2 For Trending Topic 2
Tweets Group n For Trending Topic n
Personalized Trending Topics

Figure1. The Structure of the T3C Twitter Client.

Data was collected using the Twitter streaming API[6], with the filter tweet stream providing the input data and the trends/location stream providing the list of terms identified by Twitter as trending topics. The filter

Identify applicable sponsor/s here. If no sponsors, delete this text box. (sponsors)

[1] http://thecadmus.com/
[2] http://keytweet.com/

[3] http://repo1.maven.org/maven2/net/homeip/yusuke/twitter4j/
[4] http://www.javacodegeeks.com/2011/10/java-twitter-client-with-twitter4j.html
[5] http://www.winterwell.com/software/jtwitter.php
[6] http://twitter.com

streaming API is a limited stream returns public statuses that match one or more filter predicates. The United States (New York) and Canada (Toronto) was used as the location for evaluation. Google Geocoding API has been used to get location wise Twitter data[7]. The streaming data was collected automatically using the Twitter4j API. The streaming data was stored in a tabular CSV formatted file. Data has been collected with different time interval for same city and topic. We collected different topics of dataset from different city with different time interval. We collected Canada gas price topics from Thunder Bay and Toronto on 25th April 2012 and 26th April 2012 about 200 tweets. We collected sport (Basketball) related topics from New York and Toronto on 8th May 2012 and 9th may 2012 about 28000 tweets. We collected health (flu) related topics from Toronto and Vancouver on 8th May 2012 and 9th May 2012 about 600 tweets. We collected political (election) topics from Los Angeles and Toronto on 9th May 2012 and 10th May 2012 about 6000 tweets. We collected education (engineering school) related topics from Toronto and New York on 9th May 2012 and 10th May 2012 about 2000 tweets. Additionally we collected large set of data from USA and Canada between 25th June 2012 and 30th June 2012. We collected total 2736048 (economy 1795211, education 89455, health 390801, politics 60265, sports 400316) tweets. We ran our client to automatically collect the data. We used multiple twitter account to collect data concurrently[8]. Next the tweets were preprocessed to remove URL's, Unicode characters, usernames, and punctuation, html, etc. A stop word file containing common English stop words was used to filter out tweets from common words.[9] The T3C client collects tweets and filters those that match the user preferences according the user feeds sent by the T3C user via the RSS protocol.

IV. T3C Trending Topics Personalization

In this section, we describe an improved Twitter personalization mechanism by incorporating the user personalization RSS feeds. The user of our T3C Twitter client can feed his or her own personalization feeds via the RSS protocol or the user can directly upload the personalize data list from a given URL. Using RSS, the T3C reads these feeds an XMLEventReader method that reads all the available feeds and store them in a personalize data list. Figure 2 illustrates the filtering of personalized tweets through the streaming process.

While the Twitter API collects tweet to form a dataset, the tweets that are related to the user personalization feeds are filtered out using string similarity method that is based on the Levenshtein Distance algorithm[10]. The Levenshtein distance is a measure between strings: the minimum cost of transforming one string into another through a sequence of edit operations. In our T3C the use of this measure can be illustrated using the following code snippet.

[7] http://developers.google.com/maps/geocoding
[8] http://flash.lakeheadu.ca/~maislam/Data
[9] http://flash.lakeheadu.ca/~maislam/Data/stopwords.txt
[10] http://en.wikipedia.org/wiki/Levenshtein_distance

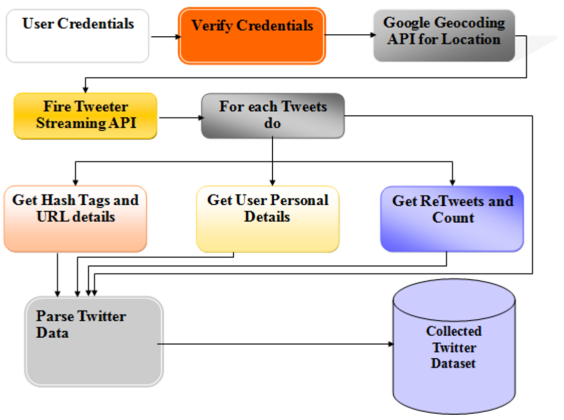

User Credentials
Verify Credentials
Google Geocoding API for Location
Fire Tweeter Streaming API
For each Tweets do
Get Hash Tags and URL details
Get User Personal Details
Get ReTweets and Count
Parse Twitter Data
Collected Twitter Dataset

Figure2. Filtering Personalized Tweets During Streaming.

```
while((line = input.readLine()) != null){
 line=cleanup(line);
double Distance = 80;
if(personalize.size()>0)
     Distance=4000;
for (int j = 0; j < personalize.size(); ++j )  {
 String comparisionTweet = personalize.get(j);
 Int thisDistance;

thisDistance=Util.computeLevenshteinDistance(comparisionTweet,
line);
  if (Distance > thisDistance) {    Distance = thisDistance;}
 }
if(Distance<=80)
         articleTextList.add(line);
}
```

The similarity detection loop continues until the end of the dataset. For each tweets we remove URL's, Unicode characters, usernames, and punctuation, html, stop words, etc. Then similarity loop iterates over the user personalize RSS data list to get the minimum Levenshtein distance value. In our implementation we have set an average distance value 80 as a good value to catch most related personalize tweets. We also found that using the Levenshtein distance we can remove duplicate tweets if the distance is zero indicating that the two tweets are identical [11]. After filtering the personalized tweets the Latent Dirichlet allocation (LDA) algorithm is used to generate trending topics model. The basic idea of LDA is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words [4]. LDA makes the assumption that document generation can be explained in terms of these distributions, which are assumed to have a *Dirichlet prior*. First a topic distribution is chosen for the document, and then each word in the document is generated by randomly selecting a topic from the topic distribution and randomly selecting a word from the chosen topic. Given a set of documents, the main challenge is to infer the word distributions and topic mixtures that best explain the observed data. This inference is computationally intractable, but an approximate answer can be found using a Gibbs sampling approach. The LingPipe LDA implementation[11] was used in our Twitter client prototype. In this LDA implementation, a topic is nothing more than a discrete probability distribution over words. That is, given a topic, each word has a probability of occurring, and the sum of all word probabilities in a topic must be one. For the purposes of LDA, a document is modeled as a sequence of tokens. We use the tokenizer factory and symbol table to convert the text to a sequence of token identifiers in the symbol table, using the static utility method built into LingPipe LDA[12]. Using the LingPipe LDA API we can report topics according to the following code snippet:

```
for (int topic = 0; topic < numTopics; ++topic) {
        int topicCount = sample.topicCount(topic);
        ObjectToCounterMap<Integer> counter = new
ObjectToCounterMap<Integer>();
for (int wordId = 0; wordId < numWords; ++wordId)
{
        String word = mSymbolTable.idToSymbol(wordId);
         double Distance = 4;
          if(personalize.size()>0)
               Distance=4000;
           for (int j = 0; j < personalize.size(); ++j )
            {
                  String comparisionTweet = personalize.get(j);
thisDistance=Util.computeLevenshteinDistance(comparisionTweet,wor
d);
             if (Distance > thisDistance) {
                                    Distance = thisDistance;
                                  }
                              }
                              if(Distance<=4 ){
counter.set(Integer.valueOf(wordId),
sample.topicWordCount(topic, wordId));
     }
 }
 List<Integer> topWords = counter.keysOrderedByCountList();
 }
```

The iterative process of identifying trending topics maps the word identifiers to their counts in the current topic. The resulting mapping is sorted for each identifier based on their counts, from high to low, and assigned to a list of integers. Then trending topics are ranked according to the Z score by testing binomial hypothesis of word frequency in personalized topic against the word frequency in the corpus [11]. Table 1 and 2 illustrates running T3C with or without personalization with initial search around football and basketball.

TABLE I.
TRENDING TOPICS WITHOUT PERSONALIZATION

Trending Topic	Count	Probability
basketball	12479	0.161
play	2471	0.032
watch	1277	0.016
school	1271	0.016
game	1153	0.015
bballproblemz	1109	0.014
#basketballproblem	1082	0.014
basketballproblem	1079	0.014
asleep	1063	0.014
love	874	0.011
player	853	0.011
football	647	0.008

TABLE II.
TRENDING TOPICS WITH PERSONALIZATION

Trending Topic	Count	Probability
basketball	3660	0.298
watch	372	0.030
love	322	0.026
play	322	0.026
game	289	0.024
football	204	0.017
player	181	0.015
team	82	0.007
don	72	0.007
season	72	0.007
baseball	59	0.005
short	57	0.005

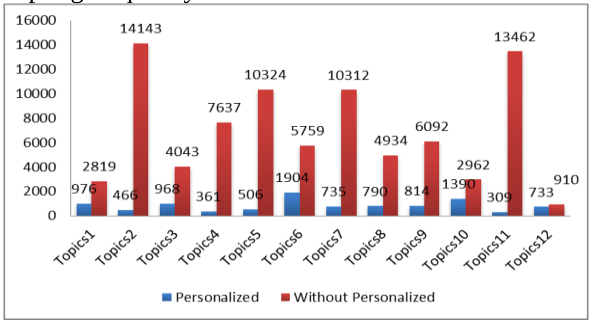
V. EXPERIMENTATION RELATED TO THE IDENIFICATION OF TRENDING TOPICS

Our experimentation starts by collecting a reasonable tweets samples on general topics like health, education, sports, ecomomy and politics. For this purpose, we run our T3C client to find trending topics by applying certain personalization feeds/queries as well as without any personalization. To demonstrate the effects of our RSS-Based personalization, we conducted for experiments. For the first experiment, we have collected 3,90,801 Tweets related to the *health* topic and applied our personalization mechanism by uploading medical feeds related to cancer/oncology research from MedicalNewsToday[13]. We used the same sample and filter trending topics without personalization feeds for comparison purposes using the Twitter Filter API and the LDA algorithm. For personalization purpose, we apply the Twitter Filter API first to get general helath related Tweets followed by calling our rss reader to read the client personalization feeds and after that we apply the Levenshtein Distance algorithm (between user personalization feeds and health related tweets) followed by the LDA algorithm to finally finding the personalized

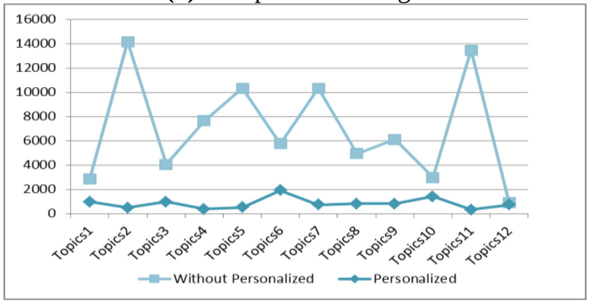
[11] http://alias-i.com/lingpipe/docs/api/com/aliasi/cluster/LatentDirichletAllocation.html
[12] http://alias-i.com/lingpipe/demos/tutorial/cluster/read-me.html

[13] http://www.medicalnewstoday.com/rss/cancer-oncology.xml

trending topics. Figures 3a and 3b illustrate the comparison between finding trending topics with personalization and without personalization. For this expirement, we used fixed some variables like: Dirichlet priors to be .01 for η , .01 for α, number of topics to be 12 for, samples to be 2000, 200 for buring period and 5 sampling frequency[14].



(a) Comparison Histogram

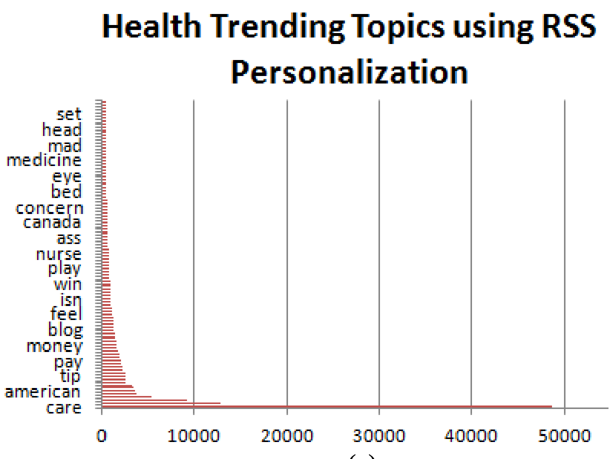


(b) Comparison Graph

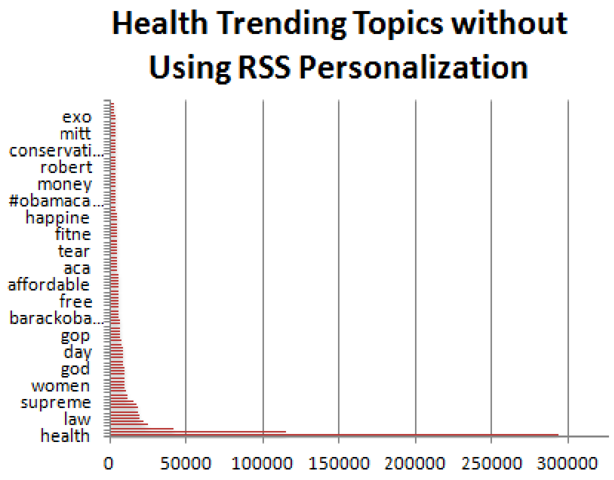
Figure 3. Comparing Health Related Trending Topics with RSS Personalization and without Personalization.

Figures 4.a and 4.b are showing the most frequent health related trending topics word for both personalize and non-personalize topics.

Moreover, we conducted similar experiments using other general topics. For economy and finance, we collected 17,95,211 and used the Economist Banking RSS feeds[15]. For education we collected 89455 tweets and used the CBC Technology Feeds[16]. For politics we collected 60265 tweets and use the CBC Politics Feeds[17]. Finally for sports we collected 400316 tweets and used the CBC Sports Feeds[18]. We publish all the results of these experiments on our Lakehead University Flash server[19]. Our experiments shows clearly that our RSS-Based personalization mechanism finds trending topics that matches the user perferences through the provided user feeds.



(a)



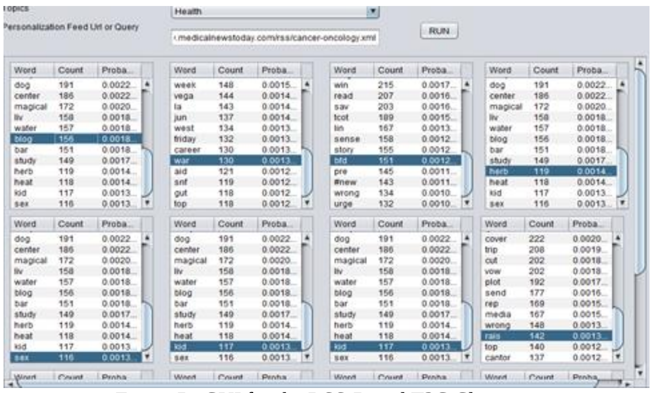
(b)

Figure 4. Frequency Counts for Trending Topics with or without RSS Personalization.

VI. CONCLUSIONS

While numerous volumes of Tweets users receive daily, certain popular issues tend to capture their attention. Such trending topics are of great interest not only to the Twitter micro-bloggers but also to advertisers, marketers, journalists and many others. An examination of the state of the art in this area reveals progress that lags its importance [14]. In this article, we have introduced a new method for identifying trending topics using RSS feeds. In this method we used two algorithms to identify tweets that are similar to the RSS Levenshtein Distance algorithm and the LDA. Although LDA is a popular information retrieval algorithm that have been used also for finding trending topics [12], no attempt that we know have used the RSS feed for personalization. Figure 5 shows a screenshot of GUI of our RSS-Based Personalization Twitter Client (T3C).

[14] http://alias-i.com/lingpipe/docs/api/index.html
[15] http://www.economist.com/topics/banking/index.xml
[16] http://rss.cbc.ca/lineup/technology.xml.
[17] http://rss.cbc.ca/lineup/politics.xml.
[18] http://rss.cbc.ca/lineup/sports.xml
[19] http://flash.lakeheadu.ca/~maislam/TestSample


opics: Health

Personalization Feed Url or Query: .medicalnewstoday.com/rss/cancer-oncology.xml

RUN

Word	Count	Proba...
dog	191	0.0022...
center	186	0.0022...
magical	172	0.0020...
liv	158	0.0018...
water	157	0.0018...
blog	156	0.0018...
bar	151	0.0018...
study	149	0.0017...
herb	119	0.0014...
heat	118	0.0014...
kid	117	0.0013...
sex	116	0.0013...

Word	Count	Proba...
week	148	0.0015...
vega	144	0.0014...
la	143	0.0014...
jun	137	0.0014...
west	134	0.0013...
friday	132	0.0013...
career	130	0.0013...
war	130	0.0013...
aid	121	0.0012...
snf	119	0.0012...
gut	118	0.0012...
top	118	0.0012...

Word	Count	Proba...
win	215	0.0017...
read	207	0.0016...
sav	203	0.0016...
tcot	189	0.0015...
lin	167	0.0013...
sense	158	0.0012...
story	155	0.0012...
bfd	151	0.0012...
pre	145	0.0011...
#new	143	0.0011...
wrong	134	0.0010...
urge	132	0.0010...

Word	Count	Proba...
dog	191	0.0022...
center	186	0.0022...
magical	172	0.0020...
liv	158	0.0018...
water	157	0.0018...
blog	156	0.0018...
bar	151	0.0018...
study	149	0.0017...
herb	119	0.0014...
heat	118	0.0014...
kid	117	0.0013...
sex	116	0.0013...

Word	Count	Proba...
dog	191	0.0022...
center	186	0.0022...
magical	172	0.0020...
liv	158	0.0018...
water	157	0.0018...
blog	156	0.0018...
bar	151	0.0018...
study	149	0.0017...
herb	119	0.0014...
heat	118	0.0014...
kid	117	0.0013...
sex	116	0.0013...

Word	Count	Proba...
dog	191	0.0022...
center	186	0.0022...
magical	172	0.0020...
liv	158	0.0018...
water	157	0.0018...
blog	156	0.0018...
bar	151	0.0018...
study	149	0.0017...
herb	119	0.0014...
heat	118	0.0014...
kid	117	0.0013...
sex	116	0.0013...

Word	Count	Proba...
dog	191	0.0022...
center	186	0.0022...
magical	172	0.0020...
liv	158	0.0018...
water	157	0.0018...
blog	156	0.0018...
bar	151	0.0018...
study	149	0.0017...
herb	119	0.0014...
heat	118	0.0014...
kid	117	0.0013...
sex	116	0.0013...

Word	Count	Proba...
cover	222	0.0020...
trip	208	0.0019...
cut	202	0.0018...
vow	202	0.0018...
plot	192	0.0017...
send	177	0.0016...
rep	169	0.0015...
media	167	0.0015...
wrong	148	0.0013...
rais	142	0.0013...
top	140	0.0012...
cantor	137	0.0012...

Figure 5. GUI for the RSS-Based T3C Client.

We are continuing our attempts to develop more personalization mechanisms that adds more focused identification of personalized trending topics using techniques that utilize machine learning algorithms [13]. The results of these experiments will be the subject of our next article.

ACKNOWLEDGMENT

Dr. J. Fiaidhi would like to acknowledge the support of NSERC for the research conducted in this article.

REFERENCES

[1] James Benhardus, Streaming Trend Detection in Twitter, 2010 UCCS REU FOR ARTIFICIAL INTELLIGENCE, NATURAL LANGUAGE PROCESSING AND INFORMATION RETRIEVAL FINAL REPORT.

[2] Ming Hao et. al, Visual sentiment analysis on twitter data streams,2011 IEEE Conference onVisual Analytics Science and Technology (VAST), 23-28 Oct. 2011, pp277 – 278

[3] Suzumura, T. and Oiki, T., StreamWeb: Real-Time Web Monitoring with Stream Computing, 2011 IEEE International Conference on Web Services (ICWS), 4-9 July 2011, pp620 – 627

[4] Kevin R. Canini, Lei Shi and Thomas L. Griffiths, Online Inference of Topics with Latent Dirichlet Allocation,In Proceedings of the International Conference on Artificial Intelligence and Statistics, 2009, http://cocosci.berkeley.edu/tom/papers/topicpf.pdf

[5] Bendersky, M. and Croft, W.B. Discovering key concepts in verbose queries. SIGIR '08, ACM Press (2008).

[6] Nomoto, Tadashi and Matsumoto, Yuji,EXPLOITING TEXT STRUCTURE FOR TOPIC IDENTIFICATION, Workshop On Very Large Corpora, 1996

[7] Salton, G., & Yang, C. S. (1973). On the specification of term values in automatic indexing. Journal of Documentation, 29(4), 351–372.

[8] Hearst, M. (1997). Texttiling: Segmenting text into multi-paragraph subtopic passages. Computational Linguistics, 23(1), 33–64.

[9] David M. Blei, Andrew Y. Ng and Michael I. Jordan, Latent Dirichlet Allocation, Journal of Machine Learning Research 3 (2003) 993-1022

[10] Alexander Pak and Patrick Paroubek, Twitter as a Corpus for Sentiment Analysis and Opinion Mining, Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)}, May 19-21, 2010,Valletta, Malta.

[11] Alex Hai Wang, Don't Follow me: Spam Detection in Twitter, IEEE Proceedings of the 2010 International Conference on Security and Cryptography (SECRYPT), 26-28 July 2010, http://test.scripts.psu.edu/students/h/x/hxw164/files/SECRYPT2010_Wang.pdf

[12] Daniel Ramage, Susan Dumais, and Dan Liebling, Characterizing Microblogs with Topic Models, in Proc. ICWSM 2010, American Association for Artificial Intelligence , May 2010

[13] Kathy Lee, Diana Palsetia, Ramanathan Narayanan, Md. Mostofa Ali Patwary, Ankit Agrawal, and Alok Choudhary, Twitter Trending Topic Classification, 2011 11th IEEE International Conference on Data Mining Workshops, ICDMW2011,pp.251-258.

[14] Fang Fang and Nargis Pervin, Finding Trending Topics in Twitter in Real Time, NRICH Research, 2010, Available online: http://nrich.comp.nus.edu.sg/research_topic3.html.