Technical Report for SF1 Topic Detection and Tracking

Jia Guo

June 7, 2011

Document History		
Date	Author	Content
2011-05-30	Jia Guo	Initial version.
2011-06-07	Jia Guo	Revised. Add "Related Works", "API", "Evaluation", "Future works" sections. Add reference links. Add more formulas

Abstract

Detecting and characterizing emerging topics of discussion and consumer trends through analysis of Internet data is of great interest to businesses. This report considers the problem of monitoring the Web to spot emerging events distinctive phrases which act as "tracers" for topics as a means of early detection of new topics and trends. We also analyze the linking relationship among all topics.

Contents

1 Introduction

Topic detection and tracking (TDT) is an area of information retrieval research the focus of which revolves around news events. The problems TDT deals with including [4, 7, 6]

1. **Segmentation** Splitting a continuous stream of news into stories that are about a single topic.
2. **Detection** Gathering stories into groups that each discuss a single topic.
3. **First story detection** Identifying the onset of a new topic in the news.
4. **Tracking** Keep track of stories similar to a set of example stories.
5. **Link Detection** Detect whether or not two stories are topically linked.

In this work, we explore the n-grams based technique which we had shown in *traditional KPE* [2] to represent news documents in order to detect new events and track their development. In the next section, we review the related works on TDT area .Section 3 reviews the n-gram based approach in *traditional KPE* and modify it to generate candidate phrases for topics' representation. We describe the burst detection algorithm in Section 4. Section 5 describes the link analysis process to identify similar or same topics. In Section 6, we will show the API and usage examples. We do evaluation in Section 7. In Section 8 we will make some future improvement plan.

2 Related Work

There have been a large number of systems built for the purpose of browsing the information with time. [9] used *Expected Mutual Information Measure* to measure associations among topics. [10] extracted topical words by LDA-based approach. [5] use patterns to identify topics then computing the similarity by their morphology. [3] tracked the burst topics by derive momentum, acceleration, and force from these. [1] built a relation graph on temporal information to detect the relationship. [7] uses unigram language model combine with semantic features to detect the topic and also measuring the links. [11] uses keywords in events to track the specific topics. [4] built a IR liked relevance model and then use KL-divergence to measure the similarity of two events. In [5, 11, 4], a topic or event must be represented as a whole sentence.

3 Traditional KPE

In *traditional KPE* [2], we have three steps to extract key-phrases:

1. Generating all n-grams.
2. Computing some statistics for all n-grams and do filtering by thresholds.
3. Do filtering by the prefix and suffix statistics from Wikipedia concepts.

In step 1, we extract the n-grams together with their left and right terms. So each continous n-gram is represented as {[left term]+ , n-gram terms, [right term]+}.

In step 2, some statistic approaches were used, such as 1) mutual information; 2) Log-Likelihood; 3) Baised contextual entropy. After this filtering work, most of the key-phrases are important statistically. However, some of them are not good for human beings.

In step 3, we further filtered the ones judged by Wikipedia concepts.

4 Temporal KPE

4.1 Introduction

We made some changes on traditional KPE to do the temporal topic detection. In temporal topic detection, each document is tagged by time to indicate when this news or blogs had been post.

In temporal KPE, we

1. Do not use Wikipedia concepts statistics, so it means step 3 in traditional KPE was ignored.
2. Detect the burst events by momentum and acceleration of their frequency on time series [3].

4.2 Burst Detection Algorithm

By using the momentum and acceleration of temporal value, it looks like the stock technical analysis.

Let's denote

1. $x(t)$ as the frequency of one topic on time t. So $x(t)$ can make the time series.
2. $m(t)$ as the importance of that topic on time t, like the reply number for blogs or forum's threads.
3. $v(t) = dx(t)/dt$: the velocity of the change.
4. $a(t) = dv(t)/dt$: the acceleration of velocity changes.

$v(t)$ could be measured by the difference of moving average(http://en.wikipedia.org/wiki/Moving_average) as they're smoothed value of different time range.

$$MACD = EMA(x(t), n_1) - EMA(x(t), n_2)(n_1 < n_2)$$

So $a(t)$ should also be measured by the difference of $v(t)$. As our definition of $v(t)$ is the difference of EMA, so for consistent $a(t)$ also must be defined as the difference of $EMA(v(t))$.

$$\begin{aligned} a(t) &= EMA(v(t), n_x) - EMA(v(t), n_y) \\ &= EMA(MACD, n_x) - EMA(MACD, n_y) \end{aligned}$$

Here we set $n_x = 1$ and denote n_y as n_3 for coherence. Then

$$\begin{aligned} a(t) &= EMA(MACD, 1) - EMA(MACD, n_3) \\ &= MACD - EMA(MACD, n_3) \end{aligned}$$

Actually this is just the definition of *MACD histogram.*

$$\begin{aligned} signal(n_1, n_2, n_3) &= EMA(MACD, n_3) \\ &= EMA(n_3)[MACD(n_1, n_2)] \end{aligned}$$

$$\begin{aligned} histogram(n_1, n_2, n_3) &= MACD(n_1, n_2) \\ &- signal(n_1, n_2, n_3) \end{aligned}$$

These functions are all linear ones, we can calculate the coefficients for both EMA and MACD at the start.

- **EMA**: For a variable $x = x(t)$ which has a corresponding discrete time series $\mathbf{x} = \{x_t | t = 0, 1, ..., \}$, the n-day EMA with smoothing factor α

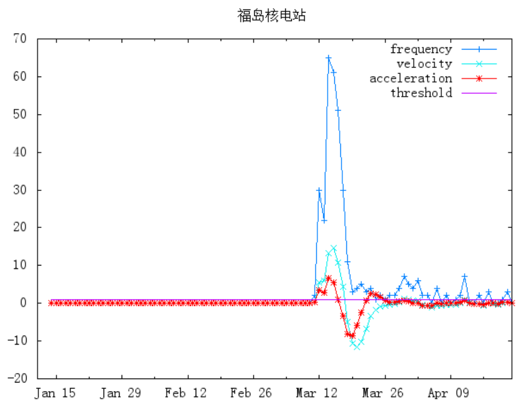
$$\begin{aligned} EMA(n)[x]_t &= \alpha x_t + (1-\alpha) EMA(n-1)[x]_{t-1} \\ &= \sum_{k=0}^{n} \alpha (1-\alpha)^k x_{t-k} \end{aligned}$$

  In this case the factor α is often taken to be $\alpha_n = 2/(n+1)$. It is obviously that the n-day EMA is a linear formula.

- **MACD**: the $MACD$, the $MACD$ *signal* and $MACD$ *histogram* are all linear functions of a time series - as they're all defined by a linear combination of EMA. For example, if $\alpha_n = 2/(n+1)$ as indicated above, and set $n_1 = 4, n_2 = 8, n_3 = 5$.

$$\begin{aligned} & MACD\_higtogram(4, 8, 5) \\ &= (1-\alpha_5)(\alpha_4 - \alpha_8) x_t \\ &+ (1-\alpha_5)(\alpha_4 - \alpha_8)(1 - \alpha_5 - \alpha_4 - \alpha_8) x_{t-1} \\ &+ ... \\ &+ \alpha_5 \alpha_8 (1-\alpha_5)^4 (1-\alpha_8)^7 x_{t-11} \\ &= \frac{16}{135} x_t + \frac{32}{6075} x_{t-1} - \frac{9536}{273375} x_{t-2} + ... + \frac{26353376}{10460353203} x_{t-11} \end{aligned}$$

Based on the histogram formula, we define the *burst* as: if the histogram value keeps on top of predefined threshold for at least two days, this topic will be considered as a *burst* on this time period. For example in the picture:



5 Link Detection

5.1 Two types of similarity

There're two types of similarity on topics. One is to find the topics which actually discuss the same thing. The other is to measure the similarity between two topics on time series. In first version we all use the vector space model - random indexing [8], for these two tasks.

5.2 Topics Sameness

Compare with [5, 11, 4], their similarity measuring approaches was based on the sentences. So lexical analysis was helpful in that situation.

However, the original output topics in our algorithm are continous n-gram - even sometimes consist of two or three characters, any lexical based approach will not help a lot.

For two key-phrases which actually discussing the same topic, they should always co-occurrence with each other in documents. So to solve this problem, the document level similarity could be applied. We can construct the vector space model by phrases-documents matrix then calculating the similarity by *Random Indexing*. For example, if we find topic A and B are the same, then a new topic will be generated - "A,B", instead of A and B independently.

5.3 Temporal Similarity

Many approaches exist for computing the similarity for two time series. Here we apply a vector space model approach on smoothed time series value.

From our definition in 4.2, $x(t)$ is the frequency of one topic on time t. Further, $s(t)$ is the EMA smoothed frequecy on time t . $s(t) = EMA(n_a)$. Then we construct the vector space model by $s(t)$ directly: the vector representation of topic A is $EMA(t_1, n_a), EMA(t_2, n_a), ..., EMA(t_m, n_a)$.

The reason why we need to smooth the original frequecy shows below. Consider we have two topics A and B. A made a burst on 2011-01-01 and B bursts on 2011-01-02, the frequencies of A and B on their burst date are both 100. And on other date they never appears. $X_A(*) = 0; X_A(20110101) = 100; X_B(*) = 0; X_B(20110102) = 100;$

If we use the original frequency, the cosine similarity of A and B is obviously zero. If we use the EMA smoothed frequency with $n = 5$, then $S_A(20110101) = 0.333{*}100 = 33.3$; $S_A(20110102) = 0.333{*}0{+}0.222{*}100 = 22.2$ and $S_B(20110101) = 0$; $S_B(20110102) = 0.33 * 100 = 33.3$. So the cosine similarity of S_A and S_B will be 0.55 instead of zero.

6 API

Temporal KPE processes time tagged documents. To process documents:

```
//we want to extract topics and track them on a collection which contains the news in
        year 2011.
//specify the valid date range
DateRange date_range;
date_range.start = boost::gregorian::from_string("2011-01-01");
date_range.end = boost::gregorian::from_string("2011-12-31");

//max processing documents number
uint32_t max_documents = 1000000;
//init the temporal kpe instance, analyzer is used to tokenize chinese sentences into
        characters, rig_resource_path is the path where we can load pre-generated
        random indexing vector.
TemporalKpe* kpe = new TemporalKpe(working_path, analyzer, date_range,
        rig_resource_path, max_documents);
//load kpe resources
kpe->Load(kpe_resource_path);
//set storage instance, it saves all output results which will used for browsing and
        searching. If not set, temporal kpe will output all results on console.
Storage* storage = new Storage(storage_dir);
kpe->SetStorage(storage);
//documents have four properties in temporal kpe: docid, title , content and date.
for (..;..;..)
{
  //got documents from SCD files or else

  //notice that now content is not used.
  kpe->Insert(date, docid, title, content);
}

//after inserting all documents, close it and some calculation will start.
kpe->Close();
```

To browse and search the detected topics:

```
//for example we want to get the topics burst in January
boost::gregorian::date start = boost::gregorian::from_string("2011-01-01");
boost::gregorian::date end = boost::gregorian::from_string("2011-01-31");
std::vector<UString> topic_list;
storage->GetTopicsInTimeRange(start, end, topic_list);

//Get info for specific topic
for(std::size_t i=0;i<topic_list.size();i++)
{
  //TopicInfoType contains the original time series data, burst date and also all of the
        similar topics.
  TopicInfoType topic_info;
  storage->GetTopicInfo( topic_list[i], topic_info);
  //do sth. on topic_info
}
```

7 Evaluation

7.1 Topics Detection Evaluation

TODO

7.2 Topics Linking Evalution

Compare Euclidean distance ,cosine similarity and SimilB with each other, do manually evaluation. **TODO**

8 Future Works

Some improvement points based on current architecture:

1. Generate sentence level description for each topic. Now all topics are represented only by continous n-gram which maybe not enough for some users. We can construct a sentence-gragh which contains the topic then use summarization liked algorithm to extract top ranked sentence as its description.

2. Choose better time series similarity algorithm.

3. More supporting on incremental.

References

[1] A. Das Sarma, A. Jain, and C. Yu. Dynamic relationship and event discovery. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 207–216. ACM, 2011.

[2] J. Guo. Sf1 keyphrases extraction technical report. `https://ssl.izenesoft.cn/projects/sf1r-engine/repository/revisions/master/raw/docs/pdf/sf1-r-owl-kpe-tr.pdf`.

[3] D. He and D. Parker. Topic dynamics: an alternative model of bursts in streams of topics. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 443–452. ACM, 2010.

[4] V. Lavrenko, J. Allan, E. DeGuzman, D. LaFlamme, V. Pollard, and S. Thomas. Relevance models for topic detection and tracking. In *Proceedings of the second international conference on Human Language Technology Research*, pages 115–121. Morgan Kaufmann Publishers Inc., 2002.

[5] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 497–506. Citeseer, 2009.

[6] J. Makkonen. Semantic classes in topic detection and tracking. 2009.

[7] R. Nallapati. Semantic language models for topic detection and tracking. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Proceedings of the HLT-NAACL 2003 student research workshop-Volume 3*, pages 1–6. Association for Computational Linguistics, 2003.

[8] M. Sahlgren. An introduction to random indexing. In *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005*. Citeseer, 2005.

[9] R. Swan and J. Allan. Automatic generation of overview timelines. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 49–56. ACM, 2000.

[10] X. Zhao and J. Jiang. An empirical comparison of topics in twitter and traditional media. *Technical Paper Series, Singapore Management University School of Information Systems*.

[11] W. Zheng, Y. Zhang, Y. Hong, J. Fan, and T. Liu. Topic tracking based on keywords dependency profile. *Information Retrieval Technology*, pages 129–140, 2008.