

A system of topic mining and dynamic tracking for social texts.

Cong Zhang

University of Tampere
School of Information Sciences
Computer Science
M.Sc. thesis
Supervisor: Jyrki Nummenmaa
January 2017

University of Tampere

School of Information Sciences

Computer Science

Cong Zhang: A system of topic mining and dynamic tracking for social texts

M.Sc. thesis, 71 pages, 1 appendix and index pages

January 2017

A massive amount of information is stored as text in the real world. Classifying the texts according to topics is an approach for people to extract useful information. Social medias generate a mass of texts every day. Topic mining and tracking on social texts are beneficial to both humanity and IT areas.

Although ready-made algorithms for topic mining and evolution tracking exist, existing methods are mostly aimed at static data and only to the mining phase of the topics. There is a lack of a general and entire solution covering all phases of topic mining and tracking of social texts.

This thesis aims to develop an entire and coherent system which can receive social texts from real-time data streams, mine topics from texts and track topic evolution over time. It is based on the existing algorithms. Tests were conducted after the development, including coverage of LDA for social texts, performance of system and presentation of system in the real environment.

According to the experiment results, the system operated smoothly in the real environment. The existing algorithms are effective to social texts. The system successfully covered the whole process of topic mining for social texts as expected. However, there is still room for system improvement. Since the system is a prototype, there may be a need to change it based on requirements of the real application if the system is put into practice and a lot of real tests should be performed in order to guarantee it is functioning well.

Key words and terms: social, platform, text, topic, mining, evolution, tracking, system, prototype, development, Twitter, test.

Table of Contents

1.	Introduction	1
2.	Literature and theory	4
2.1.	Social text and its features	4
2.1.1.	Timely and rapid information dissemination	4
2.1.2.	Vast amount of data	4
2.1.3.	Extensive content	5
2.2.	Algorithms of topic mining.....	5
2.2.1.	Overview	5
2.2.2.	LDA.....	6
2.3.	Algorithms of evolution tracking.....	11
2.3.1.	Overview	11
2.3.2.	Topic evolution method based on association filter.....	12
2.4.	General process of data mining:.....	15
3.	Design.....	17
3.1.	Overview of design	17
3.2.	Related Technology	17
3.2.1.	Java EE Web Application	18
3.2.2.	Restful Webservice	19
3.2.3.	Node.js	20
3.2.4.	Mallet Toolkit.....	20
3.2.5.	Interface design model	20
3.3.	Architecture of system	21
3.4.	Modules of function	25
3.4.1.	Text receive	25
3.4.2.	Management of mining task.....	26
3.4.3.	Execution of a task	30
3.4.4.	Other modules	33
4.	Implementation.....	34
4.1.	Text Receiver	34
4.1.1.	Express framework.....	34
4.1.2.	Input and Output of interface	34
4.2.	Management of Mining Task.....	35
4.2.1.	Related technology.....	35
4.2.2.	Front-End UI pages	36
4.2.3.	Class Diagram	39
4.3.	Task execution process	41
4.3.1.	Quartz framework	41
4.3.2.	Process component.....	41

4.3.3. Text preprocessing	43
4.3.4. Topic mining	44
4.3.5. Topic evolution tracking	45
4.4. Result presentation	46
4.4.1. D3 framework	46
4.4.2. Result visualization of mining and evolution tracking	46
4.5. Exception View	47
5. Test and Evaluation	49
5.1. Test environment	49
5.2. Test runs and results	49
5.2.1. Topic coverage test	49
5.2.2. System test	52
5.2.3. Performance	55
6. Evaluation and discussion	58
6.1. Evaluation of system	58
6.2. Discussion of system development	60
6.3. Limitation and improvement	61
7. Conclusions	63
References	65
Appendix 1	71

Acknowledgements

Firstly, my respect and honest appreciation belong to my supervisor Mr. Jyrki Nummenmaa from the University of Tampere, who guided me and gave me corrections through the process of this master thesis.

I would also like to thank Yichi Zhang and Libin Tan, who were involved in the test of the system for this thesis. Without their passionate participation, the test could not have been successfully conducted.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.

1. Introduction

A massive of information that people obtain are stored as text in the real world. The information is from multiple data sources, such as news articles, thesis papers, books, digital libraries, email and documents from web pages [Han and Kamber, 2006]. With the fast growth of digital text in the cyberspace, the need to automatically acquire useful information from text data becomes a significant issue. Text mining, which is also called Knowledge Discovery in Text, is exactly a way of analysing and categorizing texts by applying technology of data mining to natural language texts.

A useful approach of text mining is to classify the documents by topics. Effective classification of text information via topics is beneficial to readers who intend to search for archives in which they are interested. The use of text categorization via topics also includes news classification, Web page classification, intelligent recommendation of personalized news, spam mail filter, etc [Luo and Li, 2014]. Therefore, today's technology-oriented world has a need to classify texts by topic quickly and accurately.

The concept of topic detection and tracking was put forward by Defense Advanced Research Projects Agency of USA at the earliest in 1996, which aimed at discovering different news data streams automatically without manual work. Nowadays, the application of topic detection and tracking is not limited to original news medias but extended to cyber medias in large quantities. The basic tasks of topic detection and tracking are to identify topics from samples, detect new emerging topics and track evolution of old topics. Meanwhile, the popularity of topic is also considered in this area, which is usually evaluated by some measures, such as user attention, participation and timeliness.

Topic detection and tracking are to a large extent based on topic modelling. Topic models can be regarded as a mathematical probabilistic formalism underlying part of the topic mining technology. It has drawn extensive attention by mining latent information effectively. Topic modelling can represent documents with a smaller effective dimensionality by modelling the generative process of documents, word co-occurrence statistics, extracting semantics-similar topics [Xu and Wang, 2011]. Among all of them, LDA [Blei, Ng and Jordan, 2003] is a popular hierarchical probabilistic model which regards that texts are formed by topics with certain probability distribution and topics are formed by words with certain probability distribution. LDA model is able to detect meaningful information and specify semantic content in text data [Griffiths and Steyvers, 2004].

In real application areas, most of the text data change with time. As a result, text topics vary from different time periods. Therefore, evolution models based on topic models have been proposed to discover topic evolution and trends. Many methods of topic evolution detection are also based on LDA. There are also some practical systems to demonstrate topic evolution visually, such as Text-flow graph [Cui et al., 2011]. In general, topic mining and evolution have drawn more and more attention and they have been applied widely in research areas.

In the last decade, social platforms become popular with incredible speed and play more and more important roles in the life of the public. Social Web-based applications, known as social networking websites create opportunities to establish interaction among people leading to mutual learning and sharing of valuable knowledge, such as chat, comments, and discussion boards [Sorensen, 2009]. At the same time, the social network is a carrier of massive text information. According to the statistics of Twitter, 500 million Tweets are sent per day by 140 million active users. As a platform for creating lightweight blogs, Tumblr has 110 million users and creates more than 10 million blogs every day. Facebook, the biggest social network site over the world, generates 4.7 billion texts per day. As to Sina Weibo, which is a major social media like Twitter in China, it also generates above 100 million tweets per day.

Texts created from social websites or platforms can be regarded as social text. Social texts are meaningful to many areas from view of big data analysis. Many commercial organizations use the social media data to understand the needs and behaviour of their customers [Thiel and Kötter, 2012]. Topics about entertainment and leisure will be propagated and spread, such as new fashions [Zhou, 2013]. Social medias provide a direct communication channel between the public and governments. Public figures gain more popularity and level of interest. They have more influence by social medias. Social network will have powerful broadcast ability when emergencies happen, usually faster than traditional medias [Hu, 2014]. Nevertheless, social medias and the content created in them are affecting minds, behaviours and life of people, even the society's culture, gradually. Therefore, methods of text mining from social media have drawn attention of scholars and specialists in both humanities and IT areas.

Even though algorithms of topic mining, evolution tracking are ready-made, existing methods mostly aim at static data mining. The data are largely from history data. There is a lack of dynamic tracking and monitoring which is able to address real-time data stream, such as new topic detection, extinction of old topics and evolution of topics. Meanwhile, the focus of existing research is only on the mining phase of text topics. There is a lack of a general and entire solution from the start of receiving data to the key

phase of mining, then to the tracking of topic evolution. Therefore, there is a requirement to propose a general system covering the entire process. In addition, the system needs to have potential to be adapted to different topic models and algorithms because there is a need to improve and change it to apply to various situations.

In general, existing algorithms are effective in mining and grouping of topics, but there is more room for timely and dynamic mining of text topics and also lack of practical work on comprehensive solutions.

This thesis aims at creating an entire and coherent system which receives social text from real-time data stream, mine topics from texts and track topic evolution over time. Algorithms of topic mining and evolution tracking will be chosen from existing research achievements. Then, tests will be conducted to validate functions of the system. Finally, the system will be evaluated based on its applicability, followed by final discussions.

2. Literature and theory

This chapter describes the current situation of related research and introduces theory and algorithms for topic mining and evolution tracking. A general process for data mining systems will also be presented.

2.1. Social text and its features

As mentioned previously, social texts can be regarded as texts generated from social platforms. Therefore, features of social texts are highly related to social network platforms. 4A (Anytime, Anywhere, Anyone, Anything) can be a concise summary for dissemination features of social medias [Liu, 2010]. According to research resources about social media, three main features of social texts can be concluded.

2.1.1. Timely and rapid information dissemination

The most significant feature of social texts is their fast dissemination [Mai, 2012]. Most of social networks have limitations on character amount. Short texts can be generated very quickly by users. Meanwhile, users can access social network to send texts anywhere and anytime by mobile applications. Users can receive the latest information if they follow other accounts and comments can be seen by authors at the first moment.

Due to the timeliness of social texts they can be used to predict the progress of emergency situations. Savage regarded that twitter is a reflection of social fact events and detection of social groups can be done by it [Savage, 2011]. Social networks have features of medias. Moreover, their abilities of fast dissemination and timeliness can be more powerful than in traditional medias. To put it precisely, social texts are carriers for dissemination of information.

2.1.2. Vast amount of data

The amount of Internet users over the world has massively increased in the last decade. According to report by eMarketer, the amount of global Internet users is over 3 billion by the end of 2015 [Ren, 2015]. On average an Internet user has 3.8 social network accounts. The data they generated are also growing explosively. A significant part of them are in text format.

With trend of Web 2.0, the top-down way of content post is gradually replaced by the bottom-up way [Cui, 2013]. Actually, the information generated by websites is replaced by UGC (User-Generated Content). Nowadays, over 10 thousand tweet-like items of text

information are generated in one second and that will absolutely grow in the future [Xie, 2013]. Especially, there are frequent peaks of data amount when emergencies happen [Cui, 2013]. Low threshold and high speed of posting social texts lead to people being flooded in massive information [Hu, 2014].

2.1.3. Extensive content

Social networks have an all-inclusive breadth of content. Similarly, social texts have extensive contents. Every user posts different information due to their various works, hobbies and life environment. Most people find resources they need either on their own or by communicating with others. Therefore, the social network becomes an important database and information source [Zhou, 2013]. Due to popularity and extensive coverage of users, social networks have also turned into a type of public information exchange platform [Ren, 2015]. Thus, the content of social text also naturally has quite equivalent diversity.

2.2. Algorithms of topic mining

2.2.1. Overview

The topic mining of text have attracted much attention from researchers in recent years. Many achievements have been done, mainly in the areas of application and improvement of existing algorithms. Among the most popular algorithms are PLSA, LDA, LCSCS, HMETIS, Multi Topic Distribution Model, etc.

Probabilistic Latent Semantic Analysis (PLSA) is a technique from the category of topic models. PLSA was developed in 1999 by Thomas Hofmann [Hofmann, 1999] and it was initially used for text-based applications (such as indexing, retrieval, clustering). Usage of it shortly spread in other fields: such as computer vision [Lienhart, Romberg and Horster, 2009] [Monay and Gatica-Perez, 2004] [Sivic, Russell, Efros, Zis-serman and Freeman, 2005] or audio processing [Hoffman, Blei and Cook, 2009].

The LDA (Latent Dirichlet Allocation) is a type of algorithms being used in the text topic mining frequently. Created as a variant of PLSA, it is an effective tool directed at modeling huge document corpora, introduced by Blei in 2003. PLSA can be seen as a frequentist statistical approach while LDA is based on a Bayesian approach. In both PLSA and LDA, the aim is to convert a text document to a vector containing words and frequencies, ignoring the order of the words, and to represent the content of the vector as a mixture of topics [Zhao and Zhang, 2012]. It is able to facilitate the transformation from text information to digital information [Zhao and Zhang, 2012], which contributes to

analysing text similarity. Several applications of the LDA approach have been done, including applications to blogs [Nallapati and Cohen, 2008], academic documents [Wang and Land, 2011], short texts and advertisements [Wei, 2006].

In addition, the paper by Chris Clifton, Robert Cooley and Jason Rennie presents a novel method for identifying related items based on traditional data mining techniques. In the paper, frequent itemsets and HMETIS clustering are applied successfully in identifying topics in collections of news articles. The frequent itemsets are generated from the groups of items, followed by clusters formed with a hyper-graph partitioning scheme [Clifton, Cooley and Rennie, 2004].

Besides fundamental algorithms for text mining, there have been already some useful studies on improved topic searching and diverse variants.

In the paper by Zheng and Han, Multi Topic Distribution Model was introduced to mine topics. According to features of tweets, their model not only efficiently discover topics, but also is able to indicate which topics are interesting to a user and which topics are hot issues of the Twitter community [Zheng and Han, 2013]. An article, by Wang, Peng and Wang, makes the similarity calculated between texts by the linear combination of TF-IDF model and LDA model, which enables more accurate cluster analysis [Wang, Peng and Wang, 2014].

The paper by Qin, Dai and Li presented a data mining system for hot topics on the web based on the scale-free topology of the complex network [Qin, Dai and Li, 2006]. A paper, by Li, Dai, Lai and Dai, presents a statistic approach for hot topic detection in Chinese web forum by longest common segmented consecutive subsequence (LCSCS) and other techniques to overcome the basic obstacles of Chinese web data-mining: new words, non-standard syntax and Chinese word segmentation [Li, Dai, Lai and Dai, 2011]. The time and space complexity of the algorithm is acceptable.

2.2.2. LDA

LDA (Latent Dirichlet Allocation) is a mainstream algorithm in the area of text topic mining, first introduced by Blei in 2003 firstly. It originates from PLSA but produces lower perplexity and suffers less from overfitting [Blei, Ng and Jordan, 2003] [Minka and Lafferty, 2002]. Although PLSA has good performance on known training documents, LDA is better able to handle previously unseen documents [Gimpel, 2006]. Furthermore, for mining results, the model of LDA has better consistency and the running

speed of LDA is faster than PLSA [Kakkonen, Myller and Sutinen, 2006]. It has been widely applied to topic mining process.

The LDA model is described in two parts. One is how the topic model represent documents. The other one is the way to solve topics in documents.

In the topic model, a topic represents a set of words which is related to this topic. A topic can be expressed as a probability distribution over possible words that can be used while talking about that topic, so that the order or words is not assumed to matter, but some of the words have bigger probability to occur in the topic than others. In each topic, the top words that have high probability to occur will be different: sometimes, topics are summarized by listing their (unordered) high-probability top words as below.

Topic 1: Nba Final Read Watch Time Live Curry Steph Pro Beat
Free Director Nice Edition Media Cares Sports Guys Super International

Topic 2: Game LeBron Coming Gonna Bad James Lol Feel Fucking
Plan Warriors Full Lost Interesting Funny Hard Players Buy Production
Hope

Topic 3: Year Top June Day Days Case People Years Month Shit
Book Places Left Call Face Open True Ago Piece Reason

Topic 4: Fiction Science Time Life Real Art John Lot Point Hate
Understand Making High Japan Word Richard Film Dont Pretty Offers

Figure 2.1. Topic constitution by words

As a result, texts that discuss some particular topics will have a higher probability to contain top words from those topics. Firstly, the topic model describes how to generate texts. The process can be stated as below. In the original version of LDA [Blei, Ng and Jordan, 2003], the word “document” is used to represent “text”.

- Pick one topic from a given probability distribution over possible topics
- Pick one word from the word-distribution of the selected topic, and add the word to the document.

Thus, a document can be generated by repeating the process above. The probability that a particular word is generated into a document can be represented as conditional probability below.

$$p(\text{word} | \text{document}) = \sum_{\text{topic}_i}^{i=1 \text{ to } n} p(\text{word} | \text{topic}_i) p(\text{topic}_i | \text{document})$$

where topic is the i :th topic in a set of n available topics. It also can be represented as a matrix form as

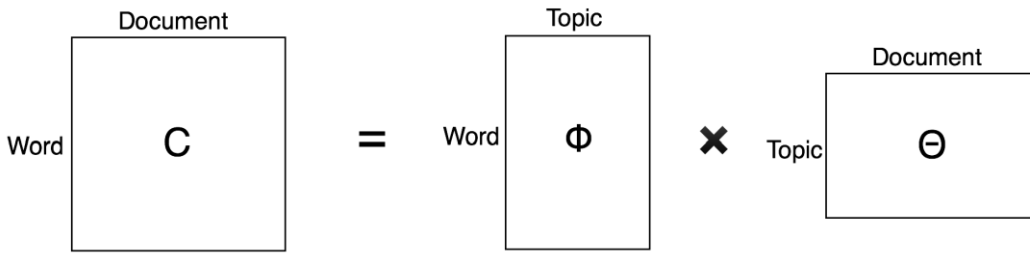


Figure 2.2. The matrix expression of topic model

The Document-Word matrix on the left side of equation represents probability of every word emerging in every document; the Topic-Word matrix represents probability of every word emerging in every topic; the Document-Topic matrix represents probability of every topic emerging in every document.

Assume that there are a batch of documents, we can easily fetch the Document-Word matrix on the left side of equation by word segmentation, counting frequency of words. Thus, the topic model can solve these two matrixes on the right side by learning and training upon the matrix on the left side.

LDA is a hierarchical Bayesian model with three levels. It allows documents to contain multiple topics. We assume the following generative process for each document w in a corpus D [Blei, Ng and Jordan, 2003]:

- Choose random variable $\theta \sim$ Dirichlet distribution p is a topic vector ($\theta | \alpha$); θ is a vector of topic probabilities for the document. α is a vector parameter for $p(\theta)$.
- For each word w_n in the document, where n goes from 1 to the number of words N in the document:
 - a) Choose a topic $z_n \sim$ Multinomial distribution $p(z | \theta)$; θ is the parameter of $p(z | \theta)$.
 - b) Choose a word w_n from $p(w_n | z_n, \beta)$, a multinomial probability conditional on the topic z_n . β is a matrix parameter for $p(w | z)$, which represents word probability distribution in every topic.

Here α is a parameter of the Dirichlet distribution (prior over topic proportions) and β is a parameter of the multinomial word choice distribution. The process can be described as graph below:

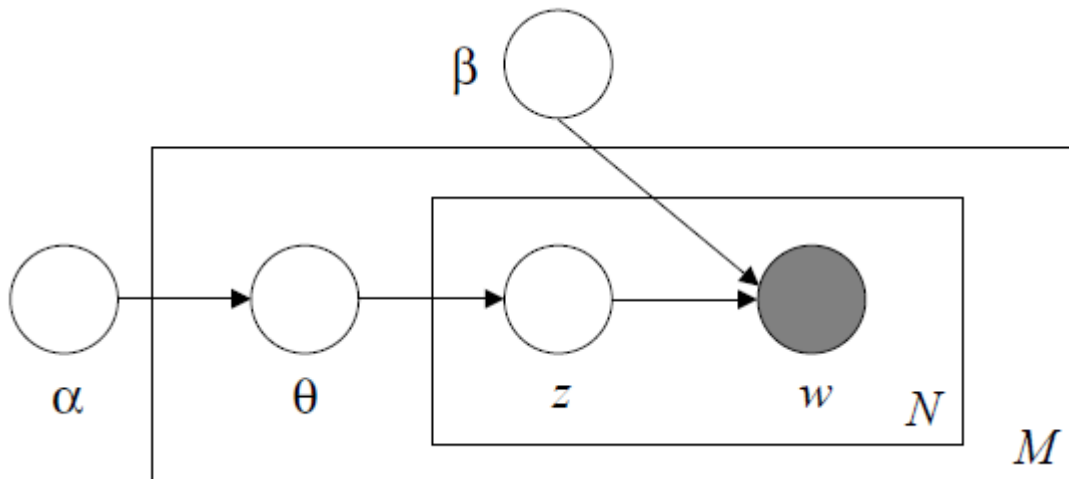


Figure 2.3. Graphical model representation of LDA. The boxes are “plates” representing replicates. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document. [Blei, Ng and Jordan, 2003]

Therefore, the joint probability of the whole process of LDA can be represented as below:

$$p(\theta, \mathbf{z}, \mathbf{w} | \alpha, \beta) = p(\theta | \alpha) \prod_{n=1}^N p(z_n | \theta) p(w_n | z_n, \beta)$$

We can interpret the equation in accordance with the graph in Figure 2.4.

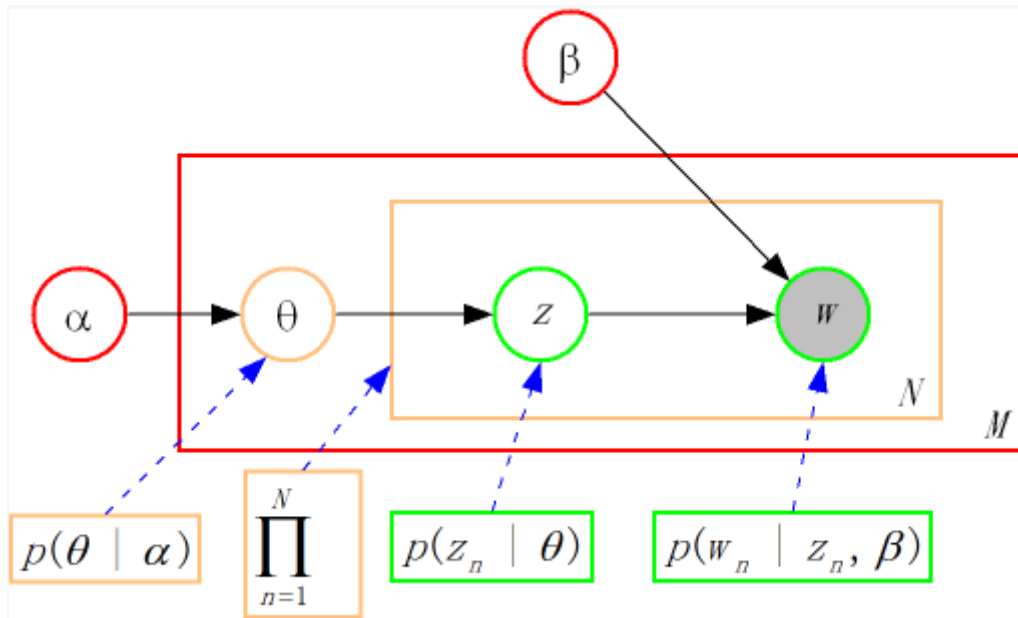


Figure 2.4. Comparison of LDA process graph and joint probability [Huagong, 2012]

The three layers of LDA are represented as above:

- Corpus-level (red): α and β are corpus-level parameters which are unique for the whole generating process.
- Document-level (orange): θ is a document-level parameter. Each of document \mathbf{w} has one θ which means there are different probability of topics for every document.
- Word-level (green): z and w are word-level parameters. z is generated on the basis of θ . w is generated on the basis of z and β . A word belongs to one topic at most.

The process of LDA generating documents is presented as above. However, documents are usually ready-made in the practical application situation. What we want to detect is topics in documents. Therefore, the actual algorithms of detecting topics are the reverse process of LDA. The key parameters we want to know are α and β . As previously mentioned, α is a vector parameter for $p(\theta)$ which generates topic vectors; β is a matrix parameter which represents $p(w|z)$, word probability distribution in every topic

In Figure 2.3 and 2.4, words achieved from documents (\mathbf{w} in grey circle) is an explicit variable. θ and z are regarded as latent variables. In practice, the variational inference (E-M) algorithm [Blei, Ng and Jordan, 2003] and Gibbs sampling [Steyvers and Griffiths, 2006] are usual methods to solve α and β by the way of learning process upon ready-made documents. Which method suits better depends on the specific topic model. Advantages and disadvantages of both two methods are discussed in the article of

Asuncion, et al [Asuncion, Welling, Smyth and Teh, 2009]. The matrices Φ and Θ in Figure 2.2 can be solved by given α and β . Thus, topics can be extracted from documents.

2.3. Algorithms of evolution tracking

2.3.1. Overview

Although the studies directly related to evolution tracking are not so abundant as topic mining, some articles and resources exist. For example, Shan summarizes three methods in LDA-based topic evolution detection according to the time sequence: joining the time to LDA model, post-discretizing or pre-discretizing methods [Shan and Li, 2010]. Elshamy analysed topic evolution model by time continuity and online progress support [Elshamy, 2013].

In order to solve the topic evolution problem, the paper by Hu and Chen uses an approach of dynamic topic modelling based on LDA [Hu and Chen, 2014]. The sets of text are split by the time and extracted to topics. Thus, the analysis of evolution can be conducted in this way. An improved online LDA(IOLDA) model was presented based on OLDA in order to solve the problem of topic mixing and untimely detection of new topics in the traditional OLDA. Meanwhile, a new method was introduced to evaluate topic intensity [He, et al., 2015]. These proposed methods are proved to be efficient for analysing topic evolution online.

Qin presented a simple but effective algorithm to detect topic evolution in terms of the birth, extinction, development, merge and split of topics within the literature in a certain field [Qin and Le, 2015]. The method divides time periods in accordance with publication dates of literatures. LDA model is applied to extract topics from each time window automatically. By topic association filter rules, evolution relationships are detected between topics in adjacent time windows. Different types of topic evolution could be detected with high accuracy according to the result.

There are also some practical systems to demonstrate topic evolution visually, such as Text-flow graph [Cui, et al., 2011] which is a coherent visualization for conveying complex relationships of topic evolution. In this way, the topic mining, evolution and visualization can communicate with each other to help users refine analysis results and gain insights into the data. D-VITA, a novel interactive visual text analysis system based on dynamic topic modelling is designed to support users exploring and interacting with large numbers of documents [Gunnemann, 2013]. This is a relatively complete system which can extract topics hidden in the documents and highlight the evolution of selected

topics. However, as a disadvantage it only works on prepared and ready-made data. It lacks the ability to process data dynamically.

2.3.2. Topic evolution method based on association filter

The way introduced in the paper of Qin [Qin and Le, 2015] is simple, effective and easy to implement. Meanwhile, it is also based on LDA model. In experiments it has shown advantages over four common baseline methods of detecting topic association [Qin and Le, 2015]. It can analyse topic evolution by distinguishing whether there is strong association between topics. The original method in the paper is mainly used in topic evolution tracking within books and within literature collections. However, it can be introduced to other fields. Hereon, we choose it as the method we implement in the system.

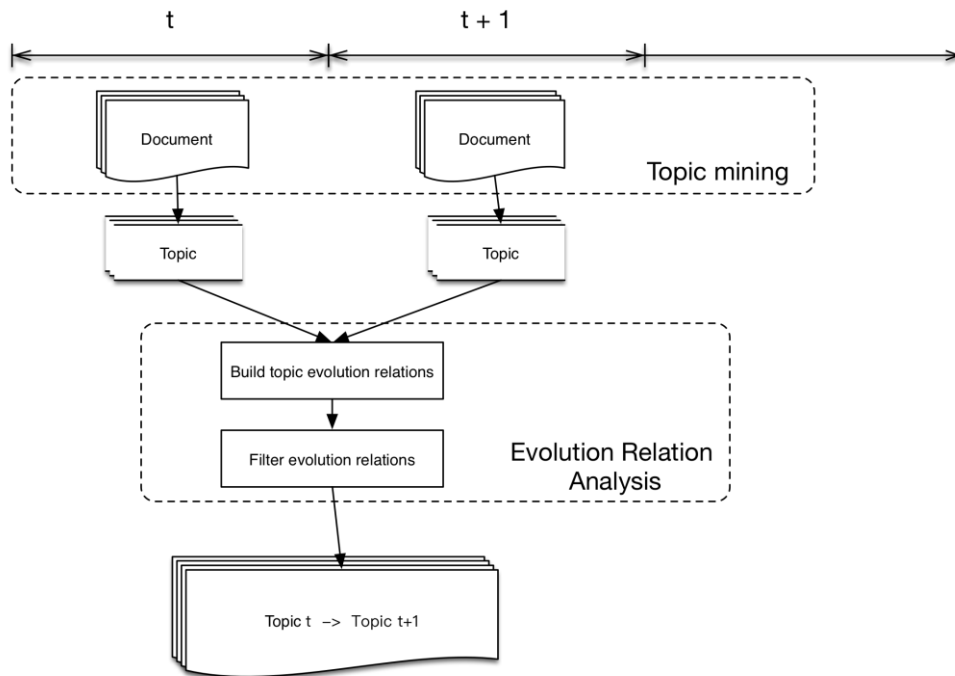


Figure 2.5. Framework of topic evolution [Qin and Le, 2015]

The basic idea of the method can be illustrated as above. It includes several steps:

1. Divide documents by same time intervals. Documents will be classified by creation time. Then, for each interval a LDA model is trained to detect the topics in those documents.
2. Calculate similarity between topics in two groups of adjacent time intervals. In particular, every topic T_i^t in the previous time interval t and every topic T_j^{t+1} in the

next time interval $t+1$ will be computed an association similarity, and this is repeated for all intervals t .

3. Filter to discover significant topic associations.
4. Infer and judge types of evolution relations according to time sequence. Finally, results will be presented.

The topic can be expressed as a vector of probabilities over words. Every dimension of a topic vector is a word. The value of a dimension is a probability that a particular keyword will be generated into a document when the topic is chosen. For instances, a topic can be represented as:

$$T = \{(w_1, p_1), (w_2, p_2), \dots (w_i, p_i), \dots (w_n, p_n)\}$$

where w_i is a word and p_i is its probability in the topic. Topic evolution includes both continuity and changes of content. This means topics between two adjacent time intervals have varying amounts of similarity in terms of their content. We can calculate the similarity to measure the continuity and build topic associations. Since different topics are represented as vectors of probabilities over the same set of possible words, cosine similarity is a practical and easy way to calculate the similarity:

$$Sim(T_i^t, T_j^{t+1}) = \frac{T_i^t \cdot T_j^{t+1}}{\sqrt{T_i^{t^2}} \times \sqrt{T_j^{t+1^2}}}$$

Post topic: for a topic T_i^t in time interval t , rank topics in next time interval $t+1$ by similarity to T_i^t descending. If there is a topic T_j^{t+1} holding maximal similarity, we refer to T_j^{t+1} as the post topic of T_i^t .

Prior topic: for a topic T_j^{t+1} in time interval $t+1$, rank topics in previous time interval t by similarity to T_j^{t+1} descending. If there is a topic T_i^t holding maximal similarity, we refer to T_i^t as the prior topic of T_j^{t+1} .

In order to improve accuracy of topic evolution analysis, the method adopts three filter rules to remove invalid topic association.

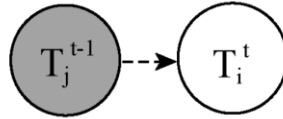
1. Set a minimal threshold of similarity ϵ . If topic similarity $Sim(T_i^t, T_j^{t+1}) < \epsilon$, then this association between two topics is invalid.
2. Assume that a topic T_j^{t+1} in time $t+1$ is the post topic of T_i^t in time t . Rank all topics in time t by the similarity to T_j^{t+1} in a descending way. If there are any topic in time

t which meet the conditions that its similarity to T_j^{t+1} is higher than T_i^t and its post topic is not T_j^{t+1} , then the association between T_i^t and T_j^{t+1} is invalid.

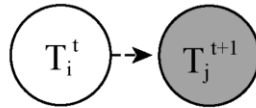
3. Set a minimal proportion threshold μ . Assume that T_j^{t+1} is a topic in time $t+1$. Rank all topics in time t by the similarity to T_j^{t+1} in a descending way. The maximal similarity is M . For any topic T_r^t in time t , if $\text{Sim}(T_r^t, T_j^{t+1}) < \mu \times M$, then the association between T_r^t and T_j^{t+1} is invalid.

After inference and judgement of topic associations filtered, results of topic evolution can be concluded as five types:

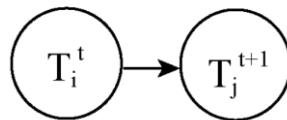
1. **Creation** (if there is no prior topic for T_i^t , this topic is created at time t .)



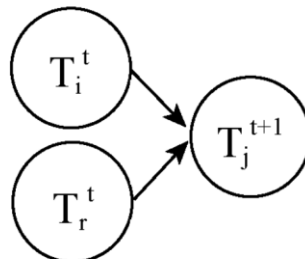
2. **Extinction** (if there is no post topic for T_i^t , this topic is extinct from $t+1$.)



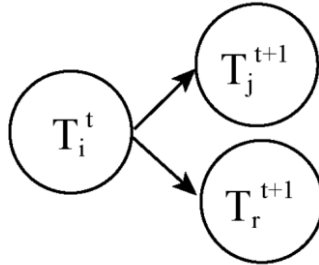
3. **Inheritance** (T_i^t is the prior topic of T_j^{t+1} ; T_j^{t+1} is the post topic of T_i^t . This can be regarded as continuity of same topics.)



4. **Merge** (there are several topics having the same post topic T_j^{t+1} . T_j^{t+1} is merged from topics in last time interval.)



5. **Split** (there are several topics having the same prior topic T_i^t . These topics are splitted from the same topic.)



The end results can be presented as example below:

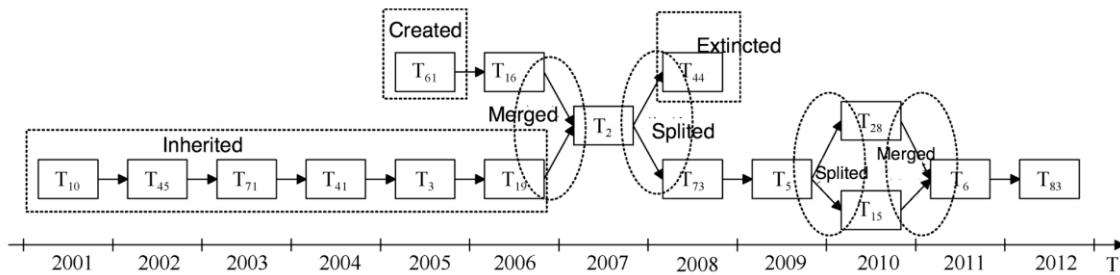


Figure 2.6. A sample of topic evolution relations [Qin and Le, 2015]

2.4. General process of data mining:

Data mining is the process of discovering interesting patterns and knowledge from large amounts of data [Han and Kamber, 2006]. The data sources can include databases, data warehouses, the Web, other information repositories, or data that are streamed into the system dynamically [Han and Kamber, 2006]. In practice, data mining usually refers to the entire process of data analysis which include some general steps: data selection, data cleaning and preparation, data mining, visualization of the results, and how to evaluate patterns discovered.

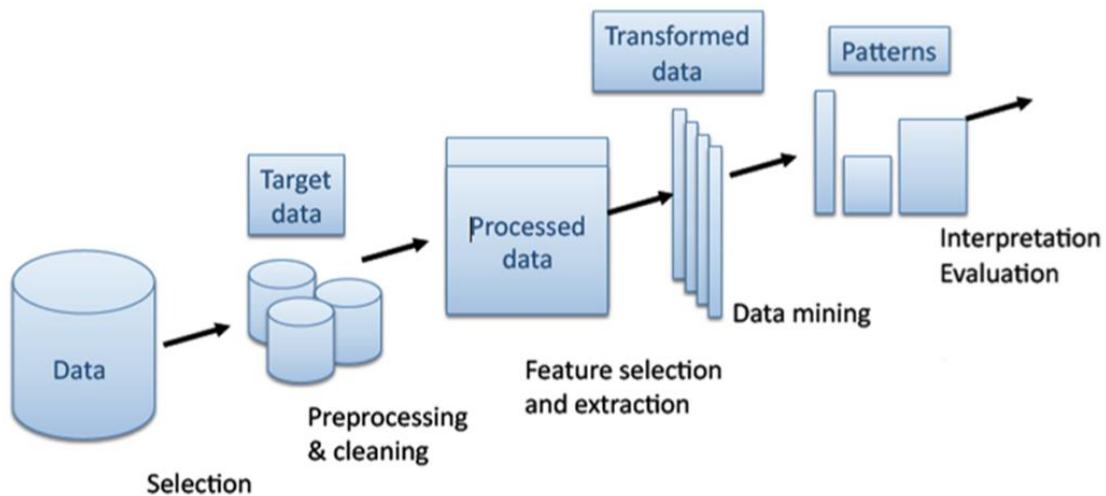


Figure 2.7. General process of data mining [Indarto, 2013]

The image above describes several general steps of data mining:

1. Data selection:

Data are retrieved from the database;
Multiple data sources may be combined.

2. Data cleaning and preprocessing:

Noise or inconsistent data are removed;
Missing data fields are handled;
Time sequence information is handled.

3. Data transformation

Useful features of data are found;
Data are transformed and consolidated into forms appropriate for mining.

4. Data mining

The essential process where intelligent methods are applied to extract data patterns.

5. Pattern evaluation

The truly interesting patterns representing knowledge are identified.

6. Knowledge presentation

The visualization of mined knowledge is represented to users

These steps are universal for most of data mining processes. Different applications of data mining may contain all or part of the steps according to practical need.

3. Design

This chapter describes design thoughts and related technology used. Architecture of system will be illustrated. Then, specific function design and modules will be drawn in accordance with system requirements.

3.1. Overview of design

As mentioned in Section 2, there is lack of a ready-made system which covers the aims of receiving real time data, topic mining and topic evolution tracking entirely at the moment. What we focus on is to create a complete and coherent system satisfying the above aims. The system can benefit commercial organizations, government organizations or media workers by helping them understand text information from social platforms and track popular trends the public are interested in, especially hot topics being discussed in social network.

According to requirements above and features of social text we mentioned in Section 2.1, fundamental demands for the system design should include:

1. Receiving social media text and storing them continuously and dynamically.
2. Preprocessing raw text for the mining task in the next step, such as removing stop-words and meaningless symbols.
3. Mining topics from social texts, based on existing algorithms.
4. Tracing topic evolution, including topic creation, elimination, merge, and split, which are mainly aimed at data divided into time periods.
5. Managing and monitoring mining processes which are configurable and working automatically. For instance, time-scheduled processes for mining topics and tracking their evolution.
6. Presenting results visually. So user can view the result as graphs and charts.
7. Core mining and tracking functions can be extended in order to consider existing different algorithms.

3.2. Related Technology

On the basis of requirements we mentioned above, several common technologies are chosen to support development of the system.

3.2.1. Java EE Web Application

Java is a type of programming language being applied in many areas since its creation in the 1990s. It is well known for being Object-Oriented, platform independent and architecture neutral. There are a large number of third-party frameworks, libraries, toolkits based on Java. Most of data mining and machine learning platforms are implemented by Java or have their Java versions, which helps us to develop a system based on these toolkits.

Java Platform, Enterprise Edition (Java EE) is the standard in community-driven enterprise software [Java EE, 2016]. It provides a technical standards and interface for enterprise application development. Java EE is developed with contributions from industry experts, commercial and open source organizations, Java User Groups, and countless individuals [Java EE, 2016].

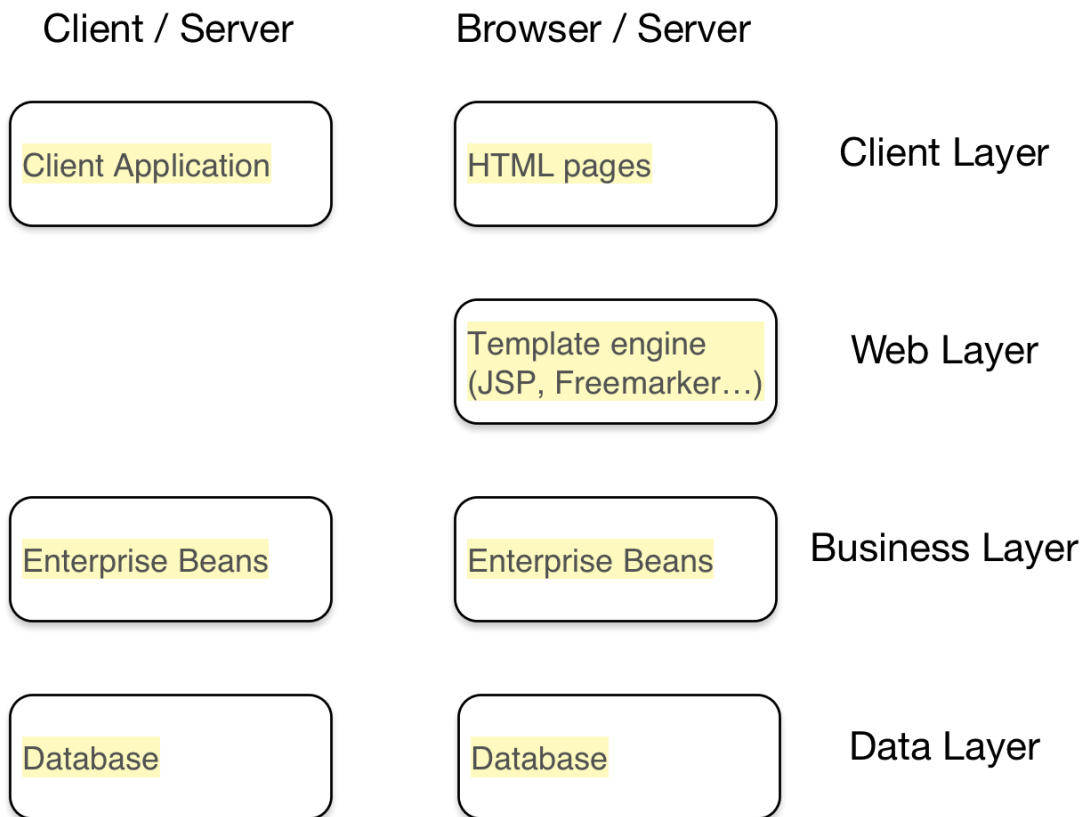


Figure 3.1. C/S and B/S Layer Architecture of Java EE

Java EE has two types of layered architectures, B/S and C/S. Nowadays, most of Java EE applications are B/S structure. Advantages of B/S structure are that it is cross-platform and distributed. Users have no need to install client programs and can easily access applications anywhere with the browser. The end results of text mining should be

presented to users using graphics. HTML and Javascript are applied in a wide range of GUIs in B/S systems, which is the base of visual presentation of topic mining results.

Storing text data is an indispensable precondition for text mining. Therefore, a database is necessary for the system. Java has native support to database operations. JDBC (Java Database Connectivity) is in common use for applications accessing databases. Nearly all types of databases are supported by JDBC. Some useful big data tools are also based on Java, such as the Hadoop framework, which is famous for having an essential role in big data processing nowadays.

3.2.2. Restful Webservice

The system needs an interface running continuously to receive text data from other applications collecting social texts, which means we need a relatively common and easy-to-use interface standard. Meanwhile, inner subsystems of the whole system also need a common communication mechanism. Webservices are a practical choice.

Webservices are platform independent, low coupling and self-describing. They are based on HTTP protocol to transfer data and could be described by XML. Webservices enable programs running on different machines or domains to transfer data without any other third-party software, which provides a common mechanism for integration between various heterogeneous systems.

Restful Webservices is one mainstream of Webservices. They directly use stateless HTTP protocol for transferring and adopting default HTTP methods (Get, Post, Put, Delete) to operate resources. They are independent of programming languages. Deployment and maintenance are very convenient.

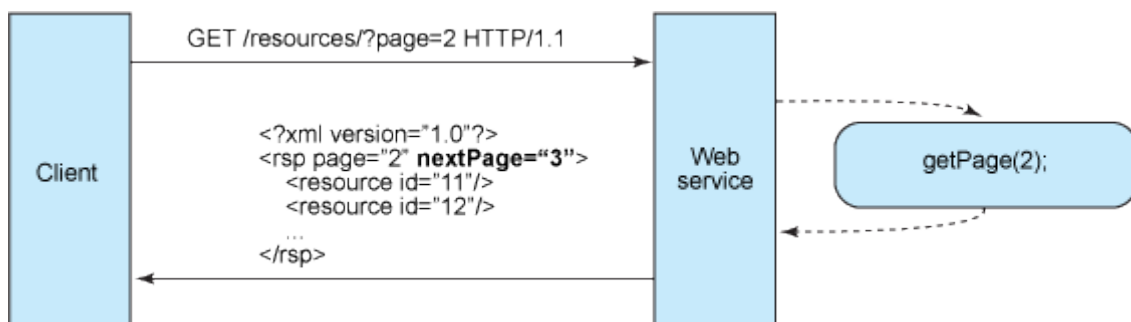


Figure 3.2. Sample of typical REST Webservice invoke operation [Rodriguez, 2008]

A typical invoking process of REST Webservice is shown above. A Client requests to get resource from a Webservice server and a server responses with XML data. Requests and Responses are transferred by HTTP.

3.2.3. Node.js

The system needs to accept text data continuously. Therefore, the interface of receiving social text data should be running continuously. Meanwhile, the amount of data being received may increase sharply at a time when some critical social events happen. Thus, the interface program is required to undertake high load. Node.js is a proper solution for it.

Node.js is an open-source and cross-platform JavaScript running environment which can be used for server and web applications. Nowadays, it has been widely used to build data-intensive applications because it is convenient to develop quick responding and easy to extend web applications. Due to it being based on JavaScript, its event mechanism reduces complexity of development and improve performance at the same time. Node.js can optimize throughput and scaling of an application. These features make it useful in real-time programs and it is also the first choice to build a REST Webservice program.

3.2.4. Mallet Toolkit

Text topic models have relatively mature theories since they were proposed in the late of 1990s. It has many implementation of different programming languages so far. Basing on existing toolkits will vastly increase our developing efficiency.

MALLET is an open source Java-based toolkit for statistical natural language processing, which includes text classification, clustering, topic modelling, information extraction, and other machine learning applications for text. It is developed by University of Massachusetts and it has become a relatively popular text mining tool in recent years. The MALLET topic modelling toolkit contains efficient, sampling-based implementations of Latent Dirichlet Allocation, Pachinko Allocation, and Hierarchical LDA [McCallum and Kachites, 2002]. The MALLET topic model package includes an extremely fast and highly scalable implementation of Gibbs sampling, an efficient method for document-topic models [McCallum and Kachites, 2002].

3.2.5. Interface design model

Topic mining is a mature area as we mentioned in Section 3.2.4. There are several mainstream algorithms we have used, such as LDA and PLSA. There are also improved

versions of these algorithms, such as the LF-LDA which uses training corpus to improve accuracy and DMM which aims at short text [Nguyen, Billingsley, Du and Johnson, 2015]. These algorithms have their features to work in different situations. They are able to extend scope of application of our system. The design model should be considered to be fit for it and make the system extendable.

Interface Design Model is a design model packaging concrete service providers. A problem that invokers often face is to use a service provided by another instance, but we are not able to determine which class the instance belong to. The common solution is to abstract the instance to an interface which can be called Service Provider. Thus, the invoker can use the interface instance to get service. The degree of the system coupling will be reduced because invoker class does not rely on any concrete service provider. Meanwhile, the independence of the interface enables changing the concrete service providers.

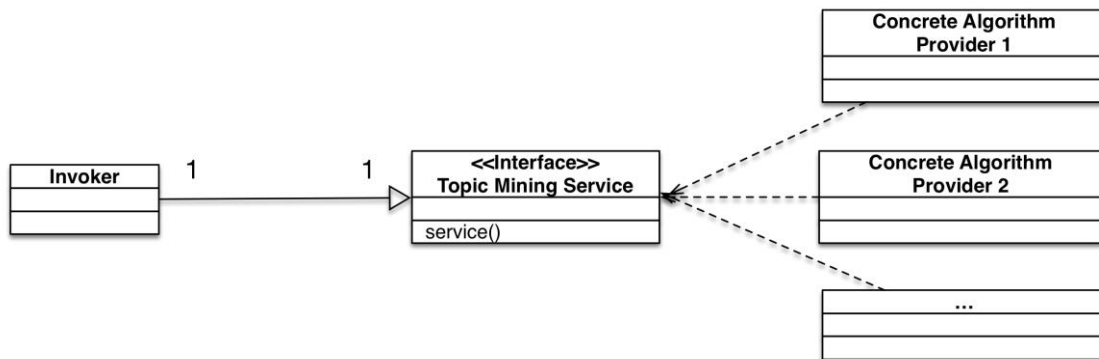


Figure 3.3. Interface Design Model

A sample diagram shown above presents that the invoker only need to interact with topic mining service interface, concrete classes of mining algorithms can be chosen by different situations or option parameters. As to our system, there are three main processes, text preprocessing, topic mining and evolution tracking. Every process should be extendable. For example, there may be requirements to deal with text from different language in text preprocessing. Technology of removing stop-words are different between English and Chinese. Besides, the topic mining process can have diverse algorithms to support various types of texts, such as short text and long text. These situations demand system design considering scalability, maintainability and possibility of the secondary development.

3.3. Architecture of system

According to targets of the system and technology standards we mentioned before, a brief diagram of module layer is shown as below.

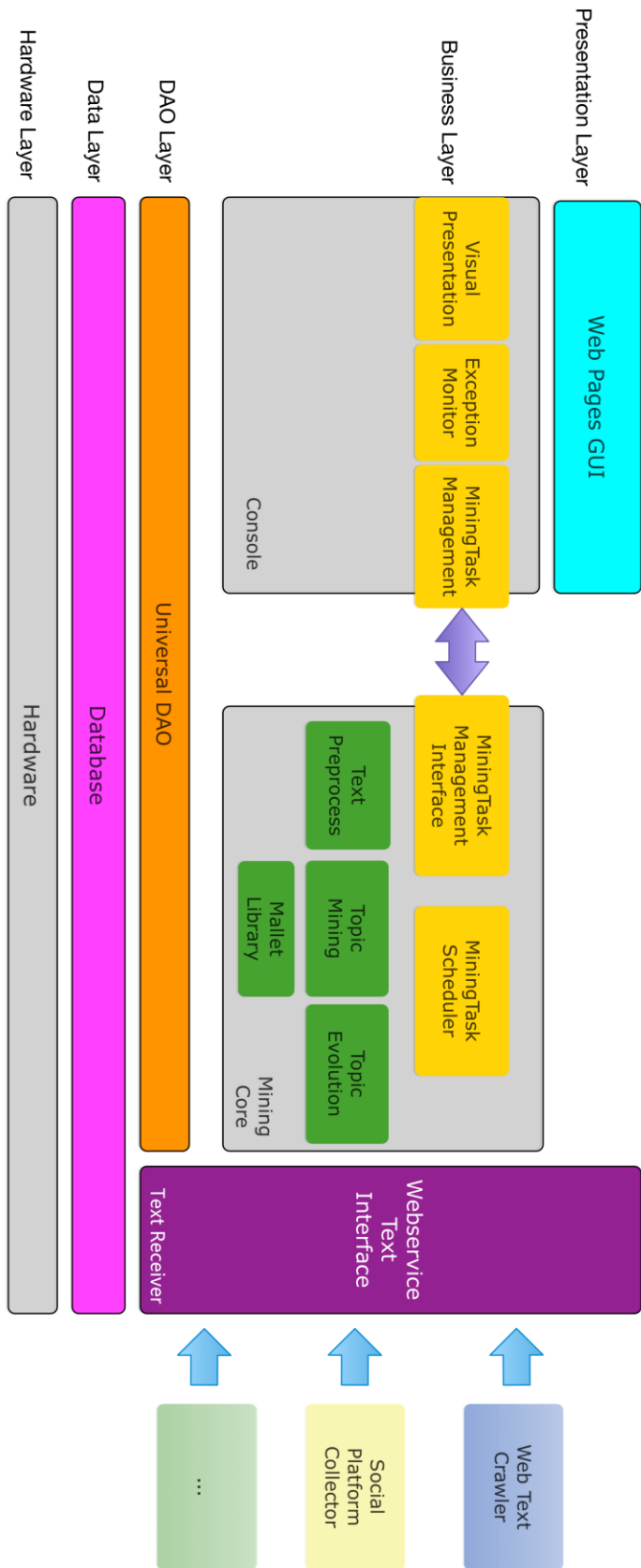


Figure 3.4. Diagram of system module layer

The whole system contains three subsystems, console, text mining core and text Webservice interface.

The main function of Text Receiver is to receive social texts from other sources. The sources can be some sort of web crawlers which crawl texts from pages of social websites or from social platform API, such as Twitter API [Twitter, 2016]. The Text Interface will be running continuously and uninterruptedly, which can insure receiving any text from any sources around the clock. The only need to send text data is to adhere to the format of REST Webservice of Text Interface.

Console consists functions for mining task management, visual presentation of results and exception monitoring in mining and evolution tracking process. The Mining Core module undertakes mining and tracking tasks. The interface of Mining Core receives commands from Console and creates tasks of text mining and tracking. The task scheduler is responsible of running tasks of mining and tracking regularly. The mining function is based on LDA algorithm mentioned in Section 2.2.2 and the evolution tracking function is based on algorithm mentioned in Section 2.3.2.

Results of topic mining and tracking will be persisted in database. The Console will also access these data and present them in graphic ways. In addition, any errors or exceptions that happened in mining or tracking processes will be recorded in database. There is a view in Console to check the information.

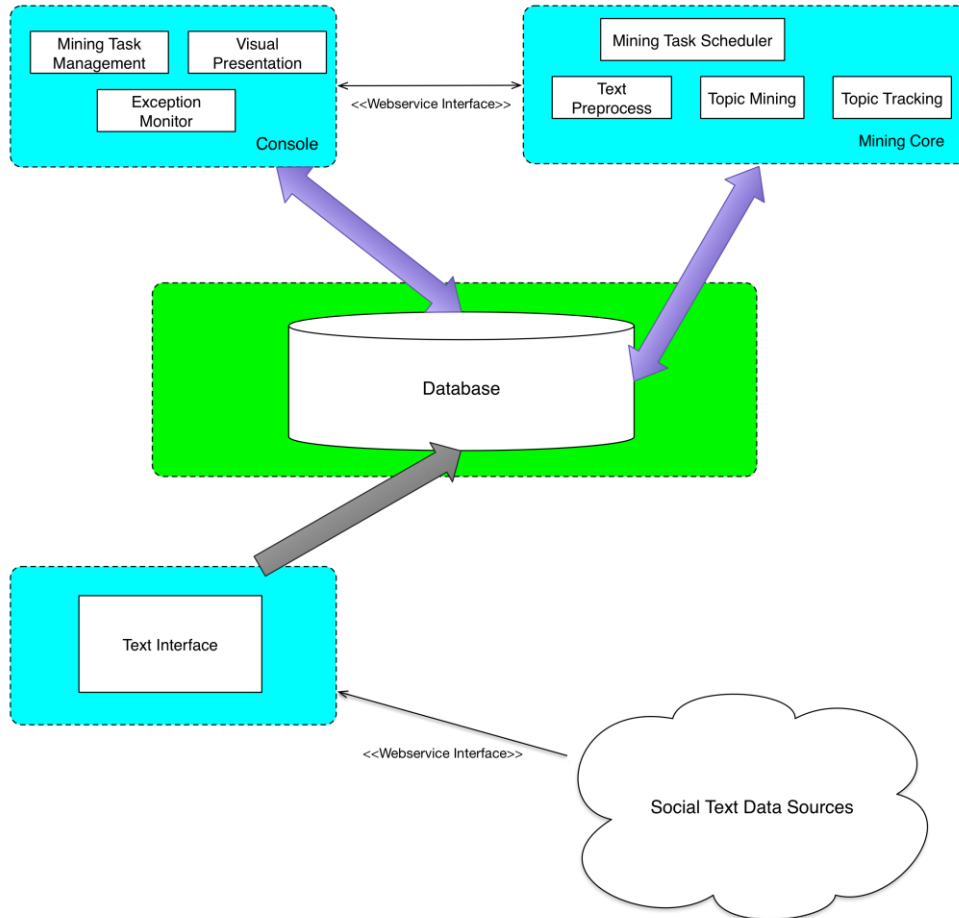


Figure 3.5. Diagram of system tasks

Three subsystem have each own duty and communicate with each other by Webservice interface. The blue dashed line areas mean different domains. Three subsystems will run in separated server domains because their tasks are relatively independent. Text interface only undertakes functions of receiving social text and persistence in database directly, which has no relationship with the others. Meanwhile, text mining and evolution tracking are highly resource-consuming tasks. They may affect other functions working if they run in the same server.

3.4. Modules of function

Functions of the system mainly include text receiving, text preprocessing, topic mining and evolution tracking. After that, users can view results by visualization module. In this section, we present main functions of the system by program flow diagrams.

3.4.1. Text receive

The task of text receiving is conducted in Text Receiver which is implemented in Node.js. It will be running as an independent server. Text Receiver receives social texts from other

programs invoking. Then, abnormal characters will be removed, such as £ (pound), \$ (dollar) and some character emotes. Later, raw texts will be stored in database and prepared for next step process.

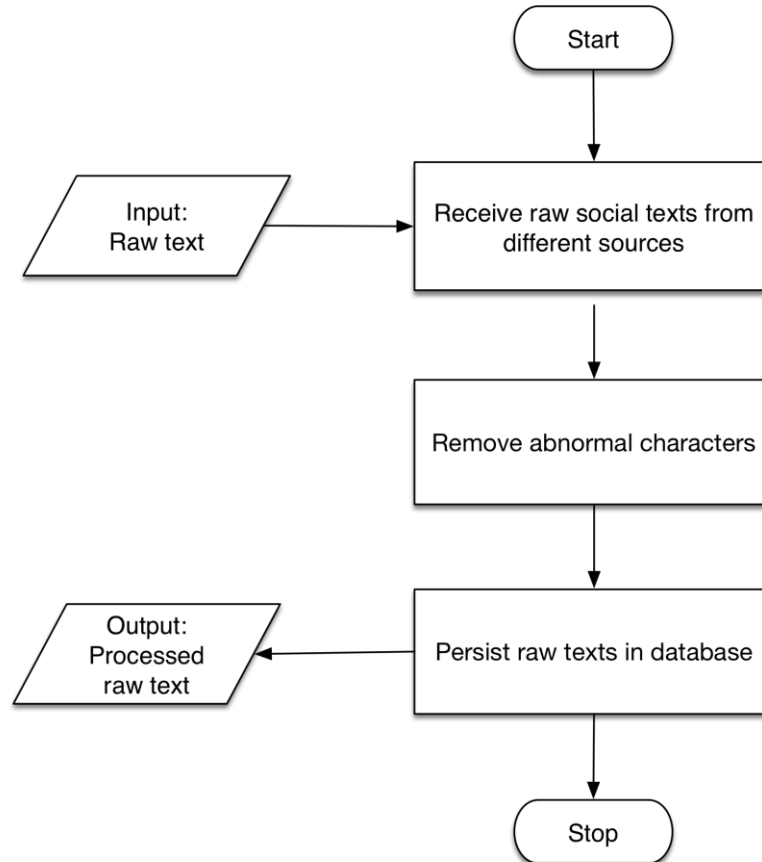


Figure 3.6. Flow diagram of social text receive via interface.

3.4.2. Management of mining task

Execution of mining text topics and tracking evolution is performed by the mining and tracking task (hereafter referred to as task). A task can have four types of status: non-started, running, stopped and completed. The status of a task is non-started before running the first time. Once a task is created and running normally, the work of topic mining and tracking will be executed at regular times set by user. A task can be stopped when running. Results of mining and tracking will still be saved after a task is stopped.

The diagram below illustrates relationships between task, topic and evolution:

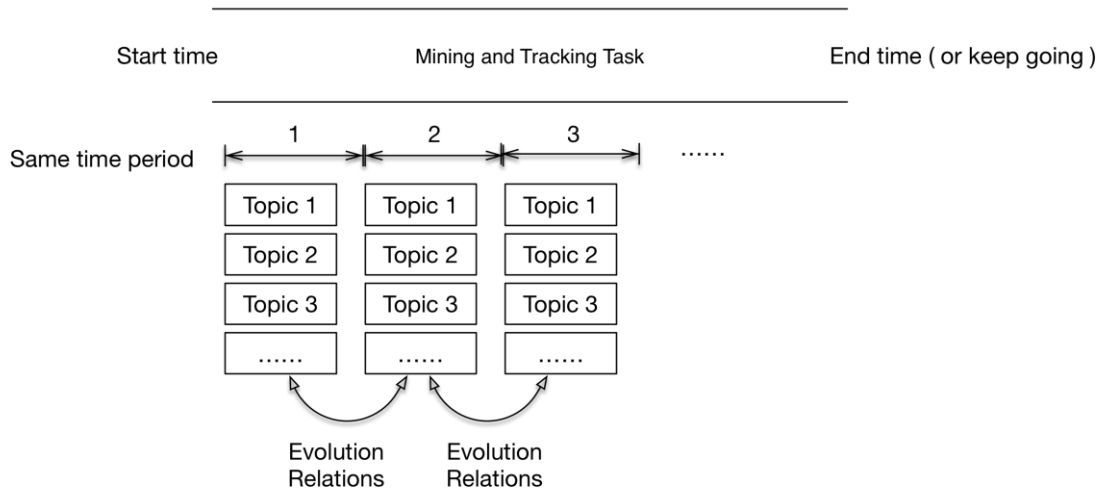


Figure 3.7. Diagram of task, topic and evolution relations.

Normally, start time and end time will be set for the task. In the time period of a running task, the work of topic mining and tracking will be executed at the same time gap. The same number of topics will be generated in every time of execution. Every topic contains a certain number of key words of which number is set by users. Mined texts come from raw texts collected in this time period. At the same time, topic evolution relations will be tracked between now and the prior period.

Create new mining task:

Users create a new task with Console. Input parameters mainly include number of topics in mining, number of key words, time gap of mining, and other related parameters. Creation command will be sent to the Mining Core subsystem. Mining Core will set a time schedule for the task. Once task has started, it is under monitoring. The first time of running the task is after the first time gap.

After Mining Core returns successful results, Console will persist the task in the database. The exception information will be shown if there is any error.

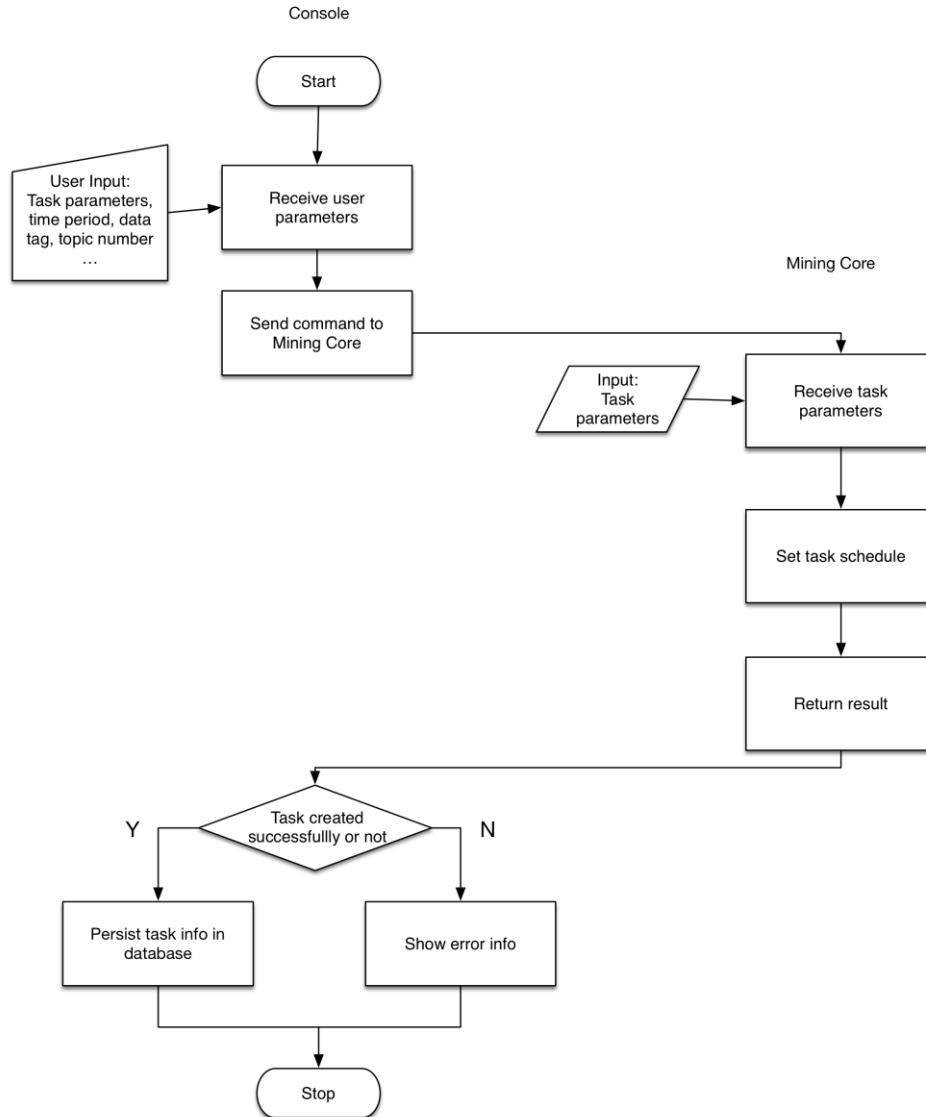


Figure 3.8. Flow Diagram of Create New Task

Stop mining task:

Stop command will be sent to Mining Core after users choose to stop a task. The schedule of the task will be stopped and never be restarted. The task status will be changed to Stopped after Mining Core returns a result. Topics which have been generated will be saved. Users can still view results of the past.

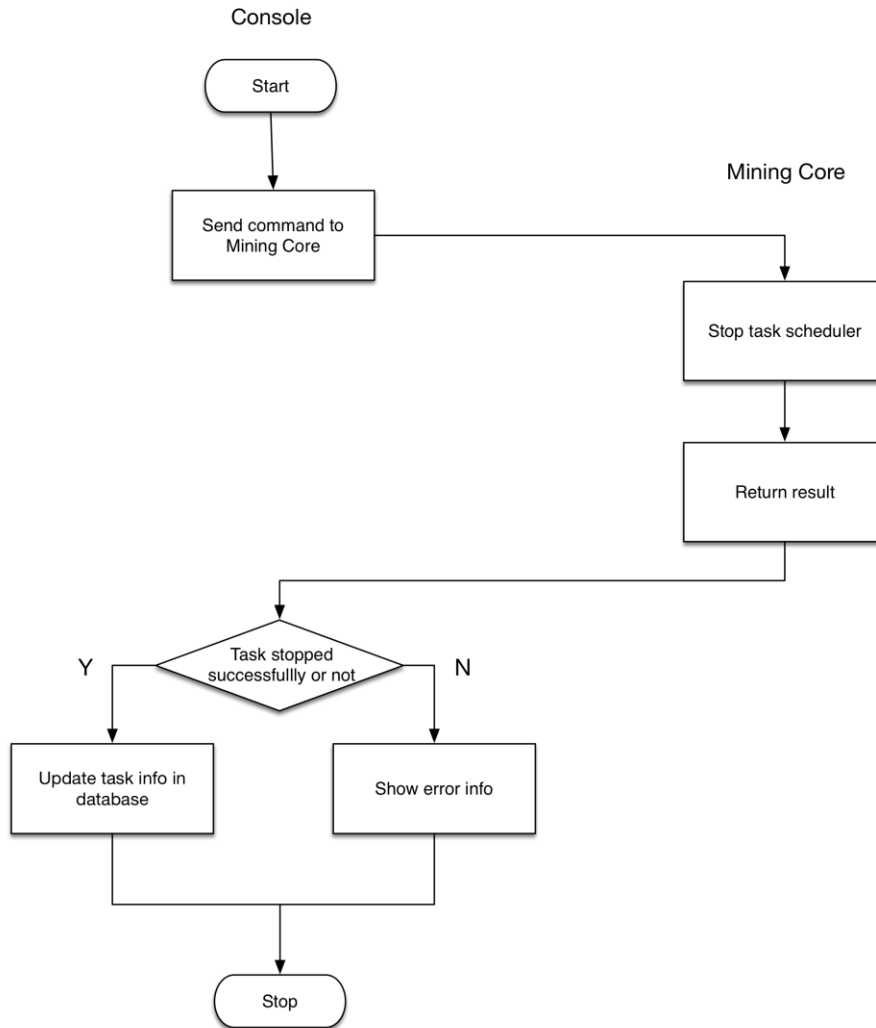


Figure 3.9. Flow Diagram of Stop Task

Delete mining task:

Delete command will be sent to Mining Core after a user choose to delete a task. Mining Core will stop the schedule of the task. Then, result will be returned to Console. After that, the task will be deleted and the related topics and evolution relations will also be deleted. All information about this task will be purged. This operation is unrecoverable.

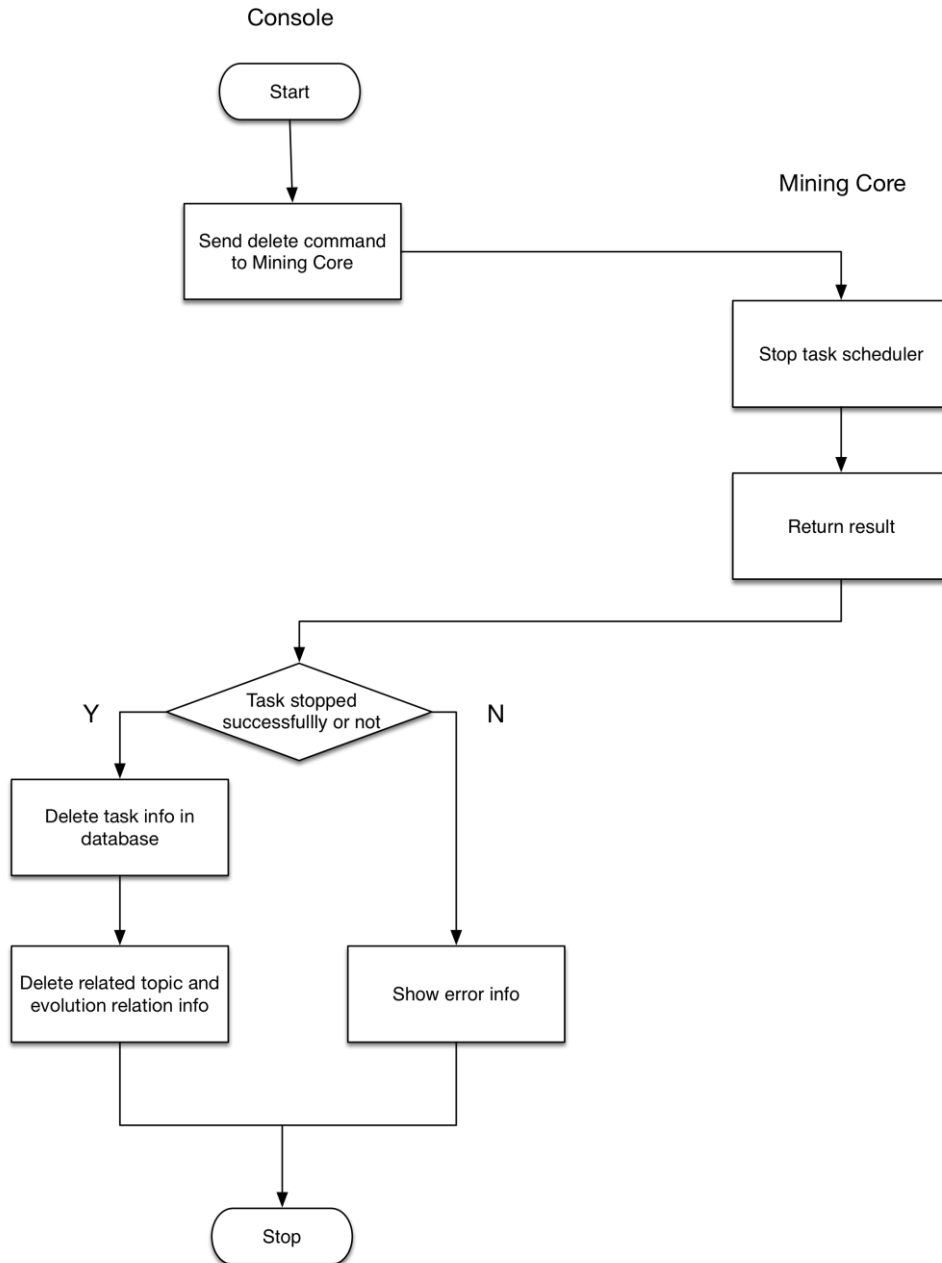


Figure 3.10. Flow Diagram of Delete Task

3.4.3. Execution of a task

The execution of a task of topic mining and tracking is conducted every time gap. The whole execution includes three main parts: text preprocessing, text topic mining and topic evolution tracking, which is similar to general data mining process mentioned in Section 2.4.

Firstly, the text preprocessing will be conducted in order to transform raw texts into formats we need for mining. Hereon, word segmentation and stop-word removing are executed in a sequence.

Word segmentation and identifying word boundaries in continuous speech or text, is a fundamental problem in Natural Language Processing (NLP) [Chen, Xu and Chang, 2011]. Simply, word segmentation will split a complete sentence into words, which is fundamental to text mining as LDA is a model presenting text by set of unordered words. For English, this process can be implemented by identifying spaces easily. There are many ready-made libraries for word segmentation of other languages.

Sometimes, extremely common words appear to be of little value in text mining. These words are called stop-words. Existence of stop-words will largely interfere results of text mining. They need to be excluded from the vocabulary entirely. A common way to remove stop-words is using stop-list to filter them before mining process.

a an and are as at be by for from
has he in is it its of on that the
to was were will with

Figure 3.11. An example list of 25 common English stop-words [Manning, Raghavan and Schütze, 2008]

After text preprocessing, raw texts are transformed into texts which consist of unordered single words and have no stop-words. These texts will be persisted in a database and used in the next step.

Secondly, topic mining will run right after the preprocessing. LDA algorithm is the base of mining as mentioned in Section 2.2.2. Mallet will be used as a ready-made library of text mining. Topics being generated will be saved in database, organized by time sequence and provided to usage in next step of evolution tracking.

Finally, evolution tracking will be operated between current topics and topics of the prior period according to the algorithm mentioned in Section 2.3.2. If there are topics mined in the prior period, evolution tracking will be conducted. Evolution relationships will also be recorded in the database.

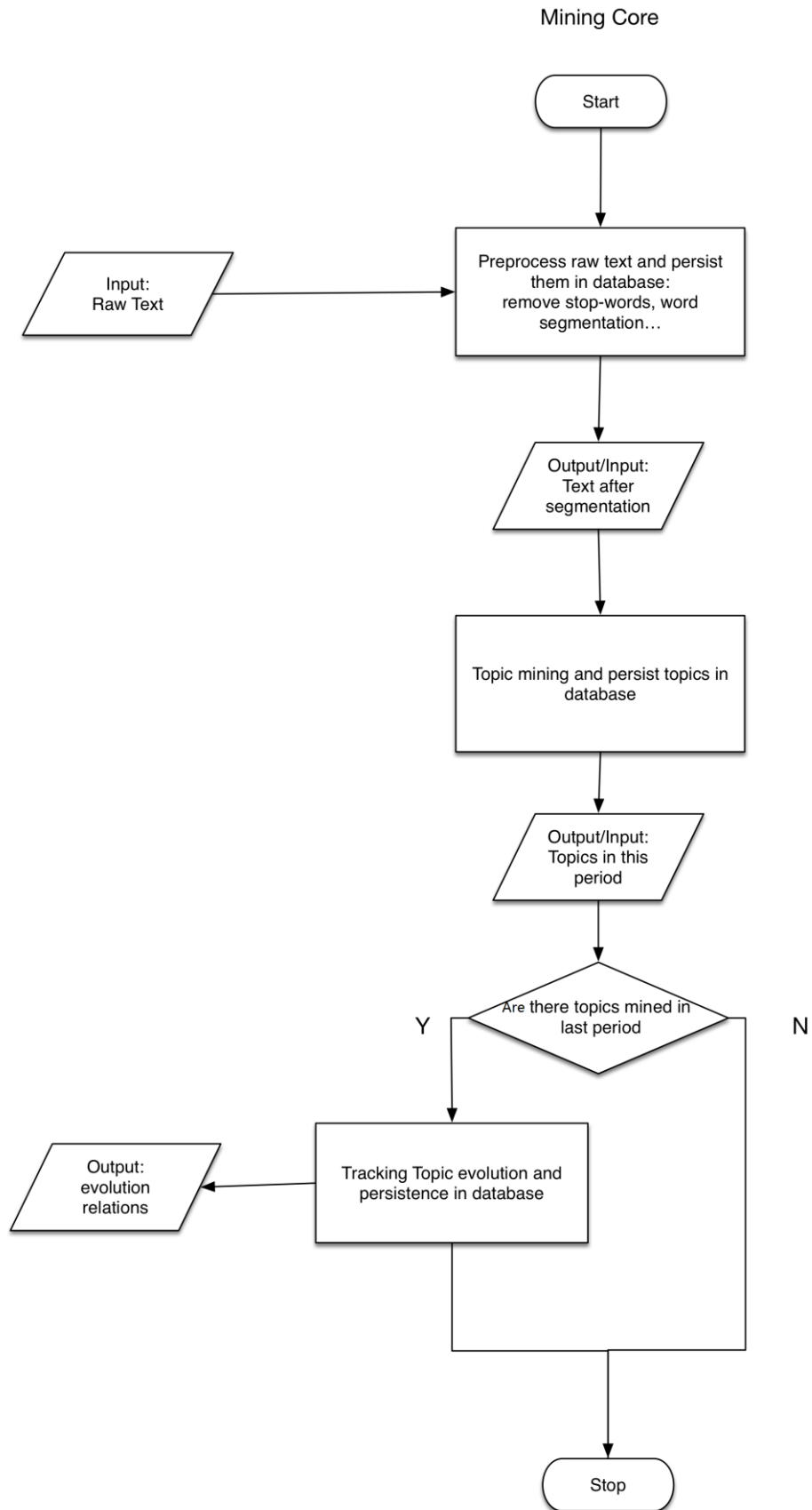


Figure 3.12. Flow Diagram of execution of Mining and Tracking Task

3.4.4. Other modules

Visual presentation:

Results of topic mining and tracking will be presented by a graph or a diagram. Topics will be shown as nodes and there are lines between previous and next topics to indicate evolution relations. Meanwhile, there is way to show heat of topics by adjusting the size of node. Users can access a view in Console to see results.

Exception monitoring:

Error or exception in every task will be recorded in database. All of raw exception information, occurrence time and stack information will be saved. Users can monitor exceptions happened from Console view.

4. Implementation

This chapter describes inner logics of each module and technology details. Meanwhile, frameworks and tools used in the system are introduced. User interfaces and running results are shown as well.

4.1. Text Receiver

Text receiver is responsible to receiving social texts from various data sources. For examples, web crawlers of news medias or invokers of social platform API can be adopted to achieve raw texts and send texts to Text Receiver. Text Receiver is an independent interface which is deployed on server and running around all the time. Invoke and result return of interface are operated by REST Webservice. Format of message transported is XML which is common and widely popular in data transmission.

4.1.1. Express framework

Express is a simple and flexible development framework based on Node.js. It extends basic functions of Node.js and, meanwhile, provides a series of powerful features to help users create various web and mobile apps. For example, request and response component for HTTP, route control and template parsing. These common features in web development enable users to construct a complete web app quickly.

With the help of Express framework, we can create a Text Receiver interface fast without support of other web containers or servers.

4.1.2. Input and Output of interface

Input message contains four parts: title, text, textCreatetime and tag. Among them, title is not a compulsory XML node. It can be created by invoker of interface to identify each piece of text. Text node is the main part of input message. Generally, it is raw text from social media. The third node is creation time of this social text, which is important to timely execution of mining task. Commonly, it can be acquired from social platform APIs. The last node is used to specify data used by every task. All of raw texts being received will be stored in the same table. Tags indicate which data is used by which mining tasks. A mining task has tag attribute which enables it to collect raw texts with same tag for topic mining.

```
<?xml version="1.0" encoding="utf-8"?>
<message>
```

```

<title>ForbesTech</title>
<text>This new iOS feature means never unlocking your phone to text again
https://t.co/IroSqcdvaV https://t.co/97BZum1jUH</text>
<textCreatetime>2016-09-18 14:12:00</textCreatetime>
<tag>unique_twitter_source</tag>
</message>

```

After receiving the requests of texts, a basic process will be conducted to ensure that the text can be stored in database, removing pound, dollar or other abnormal characters. Successful output will be sent back to invokers if text has been stored into database successfully. A sample is shown below.

```

<?xml version="1.0" encoding="utf-8"?>
<message>
  <result>success</result>
  <info>created successfully!</info>
</message>

```

If there are any errors or exceptions occurs, the value of the result node will be fail and error info will be recorded in the info node. A sample output is shown below.

```

<?xml version="1.0" encoding="utf-8"?>
<message>
  <result>fail</result>
  <info>specific error info or stack trace info</info>
</message>

```

4.2. Management of Mining Task

As mentioned in Section 3.4.2, the task is the core part of topic mining and evolution tracking in this system. User need to have interfaces to manage and monitor mining tasks conveniently, viewing results as well. The function of Console in the system is basically around the management (CRUD of task) and result visualization. Here we show the UI implementation of task management and related technology we adopt.

4.2.1. Related technology

MySQL:

MySQL is a famous and relatively popular relation database, member of Oracle, which is widely applied into various types of websites, small or personal apps. With it maturing gradually, more and more large websites have adopted it as database, such as Wiki, Facebook. MySQL has features of small size, fast execution speed and open-source and its community version is free for all developers. Therefore, it has been welcome broadly by personal or small companies since it reduces development cost largely.

Spring:

Spring is an open-source Java framework which is found by Pivotal Software. It aims at reducing complexity of enterprise application development. It provides light-weight IOC (Inversion of Control) and AOP (Aspect Oriented Programming) features which benefit any Java program development. Spring is committed to blend exist framework and provide many built-in support for web development, such as JDBC DAO, MVC framework and general transaction management.

Freemarker:

FreeMarker is a free Java-based Template Engine, originally focusing on dynamic web page generation with MVC software architecture. It's become a general purpose template engine so far, with no dependency on servlets or HTTP or HTML. It is also often used for generating source code, configuration files or e-mails. It is often used in cooperation with MVC framework, Struts, Spring, etc. Templates are written in the FreeMarker Template Language (FTL), which is a simple, specialized language (not a full-blown programming language like PHP) [Freemarker, 2016]. Freemarker is easy to learn and has good performance. Meanwhile, its built-in functions are powerful to developers to use conveniently.

Bootstrap:

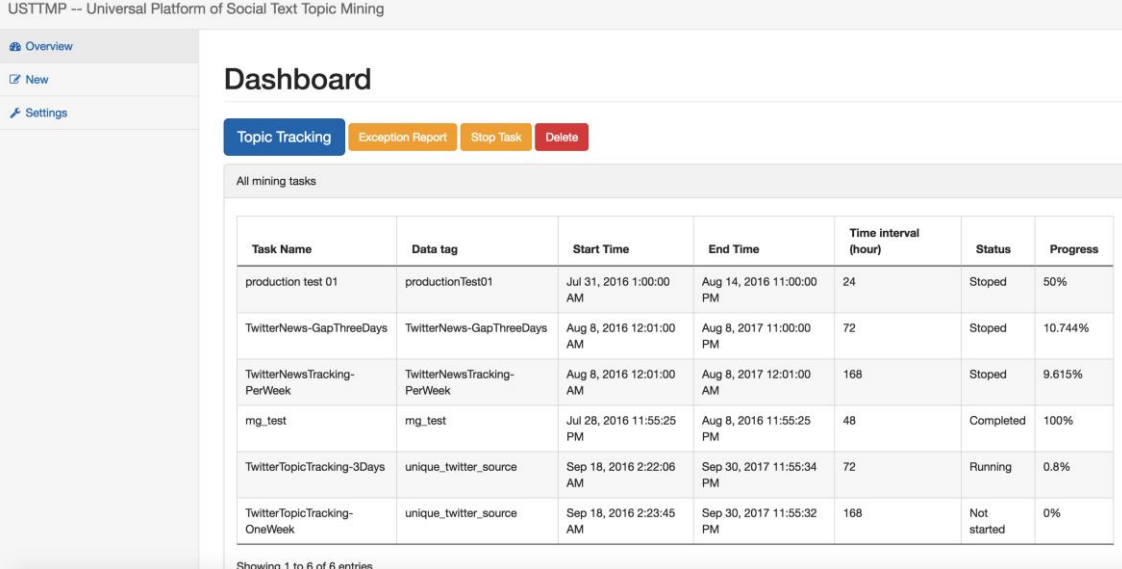
Bootstrap is a popular Front-End framework based on HTML, CSS, JavaScript. It is often used to develop web apps rapidly. Mobile-friendly and supporting most of browsers are big features of it. It provides unique layout and web components for fitting browsers in different device platforms. Meanwhile, Bootstrap is open-source and convenient to be customised. There are many extensions of Bootstrap being created since it is a very hot project in Github. In addition, its compatibility to JQuery is also perfect.

4.2.2. Front-End UI pages

The system provides a main view for management of tasks, which is also the main interface of this system. Dashboard style is adopted to present the whole main interface.

The list of tasks shows basic information of a task, such as task name, data tag (mentioned in Section 4.1, used to collect corresponding data), time interval, start and end time. If a task has not been executed yet, status should be not-started. If a task has been executed once, but not completed, status should be running. Stopped status means this task has been stopped ever. Completed status will be shown if the task has executed all required time of mining process.

There are some entrances for other functions in the main view, including buttons for topic tracking result presentation, exception report, stopping and deleting tasks. Side bar menu provides navigation of the whole system, which links Dashboard, Creating A New Task and System Settings.



The screenshot shows the main dashboard of the USTTMP system. It features a sidebar with navigation options: Overview, New, and Settings. The main content area is titled 'Dashboard' and includes several action buttons: Topic Tracking (blue), Exception Report (orange), Stop Task (yellow), and Delete (red). Below these buttons is a table titled 'All mining tasks' with the following data:

Task Name	Data tag	Start Time	End Time	Time interval (hour)	Status	Progress
production test 01	productionTest01	Jul 31, 2016 1:00:00 AM	Aug 14, 2016 11:00:00 PM	24	Stoped	50%
TwitterNews-GapThreeDays	TwitterNews-GapThreeDays	Aug 8, 2016 12:01:00 AM	Aug 8, 2017 11:00:00 PM	72	Stoped	10.744%
TwitterNewsTracking-PerWeek	TwitterNewsTracking-PerWeek	Aug 8, 2016 12:01:00 AM	Aug 8, 2017 12:01:00 AM	168	Stoped	9.615%
mg_test	mg_test	Jul 28, 2016 11:55:25 PM	Aug 8, 2016 11:55:25 PM	48	Completed	100%
TwitterTopicTracking-3Days	unique_twitter_source	Sep 18, 2016 2:22:06 AM	Sep 30, 2017 11:55:34 PM	72	Running	0.8%
TwitterTopicTracking-OneWeek	unique_twitter_source	Sep 18, 2016 2:23:45 AM	Sep 30, 2017 11:55:32 PM	168	Not started	0%

Showing 1 to 6 of 6 entries

Figure 4.1. Main page of the system

The page for creating new tasks is shown in Figure 4.2. Since the system aims at collecting dynamic data from real-time data stream and mining and tracking process will make process progress with time passing, start and end time should be future time. Total number of execution of certain task will be decided by time length between start time and end time, along with a time interval set by user.

Topic number defines number of mined topics in each of task execution process. Key word number means how many key words are contained in one topic. Alpha and beta are specific parameters for LDA topic model. They have been introduced in Section 2.2.2. Data tag is a compulsory parameter for collecting respective texts, which should be same with that in the message format (Section 4.1).

As mentioned in Section 3.4.3, the execution process of a task consists of three parts: preprocessing, topic mining and topic evolution tracking. The three parts of a process can be configured and extended due to the system supporting secondary development for not only LDA but various types of topic models. The system allows different algorithms or the methods to implement three parts of task execution. Specific implementations for each of the parts are integrated in source code and respective options will be configured in web page. Among them, the mining component is necessary because topic mining is the necessary part to be executed. The detailed implementation of three processes will be explained in Section 4.3.

After filling the form for a new task, a request will be sent to the Mining Core (Section 3.3). The new tasks will be shown in the list of main page if the creation has been successful by Mining Core.

USTTMP -- Universal Platform of Social Text Topic Mining

Overview
New
Settings

New Task

Task name
test_task

Start time
2016-09-21 03:05:29

End time
2016-12-01 21:50:18

Mining interval (hour)
24

Topic Number
20
Topic number in one-time mining.

Key Word Number
20
Key word number in one topic.

Alpha
5
Just leave it as default :)

Beta
0.01
Just leave it as default :)

Data Tag
test_task
Data Tag is necessary to be used for achieving specific data you want.

Preprocess component
English_Twitter_Preprocess

Mining component
Usttmp_Default_Topic_Mining

Evolution tracking component
Association_Filter_Evolution_Tracking

Create

Figure 4.2. The page for creating a new task

The user can choose to stop execution of a task from the main page. After doing so, the execution of task will be stopped forever and the status of task will be changed but data, including topics and evolution relations, will not be deleted. User still can view tracking

results. A stop request will be sent to the Mining Core subsystem. Results will be returned after Mining Core has completely stopped the task.

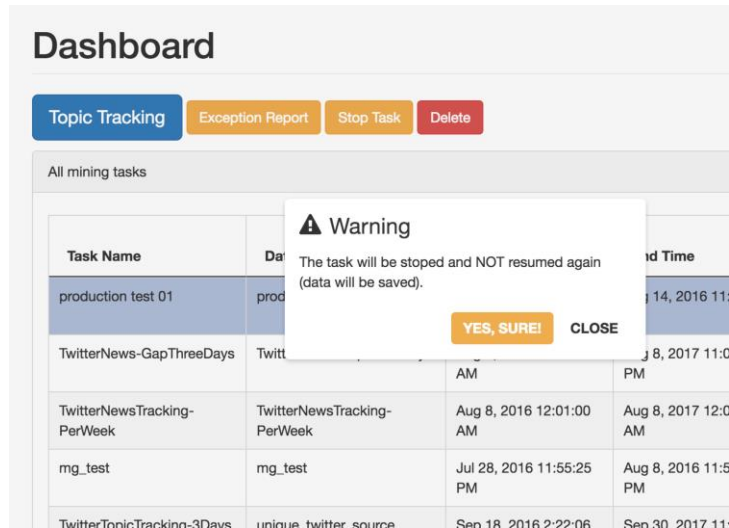


Figure 4.3. Stop a task and warning

If a task has been deleted, all data related to this task will be removed and is not recoverable, except raw texts that the task uses for mining which need to be cleared from database level manually. The delete request also will be sent to the Mining Core subsystem.

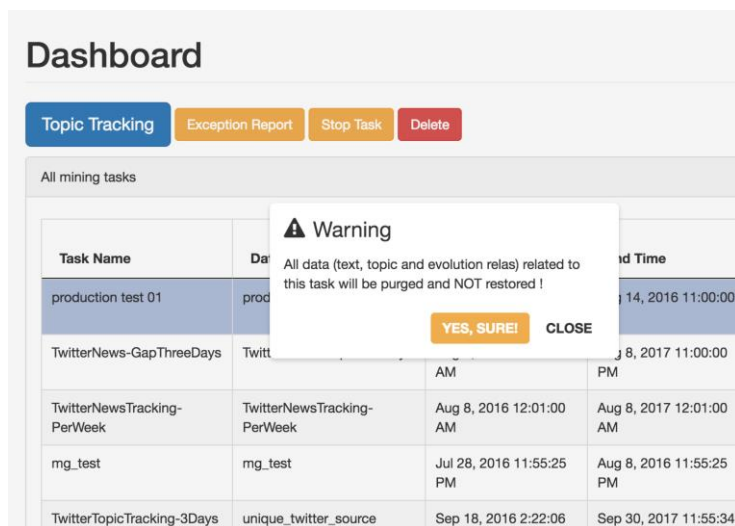


Figure 4.4. Delete a task and warning

4.2.3. Class Diagram

Structure of classes in Mining Core which handle requests of task management are shown in Figure 4.5. The design of the classes adopts Interface model. MiningTaskService is an interface which provides method definitions for managing tasks for invoking.

MiningTaskServiceImpl is the actual implementation class of MiningTaskService interface.

The addMiningTask method responds to requests of creating a new task. After receiving a request, the method invokes a checkDuplTask method to check whether there is a task with the same name. If so, an exception is thrown. Otherwise, a new task information will be created in database by doCreateSingleMiningTask method. After that, scheduleJob method will be invoked to schedule a new and timely job by Quartz framework.

The stopMiningTask method is responsible to handle requests for stopping tasks. After receiving a request, the timing job which has been scheduled in the Quartz framework will be deleted. If there is a job running currently, the job will be interrupted. Finally, the status of task will be changed by updateMiningTaskStatus method.

The deleteMiningTask method responds to requests for deleting task. After a request received, the timing job will be interrupted and deleted as stopping task operation. Then, topics and evolution relations which have been mined will be deleted. Finally, information of the task will be purged.

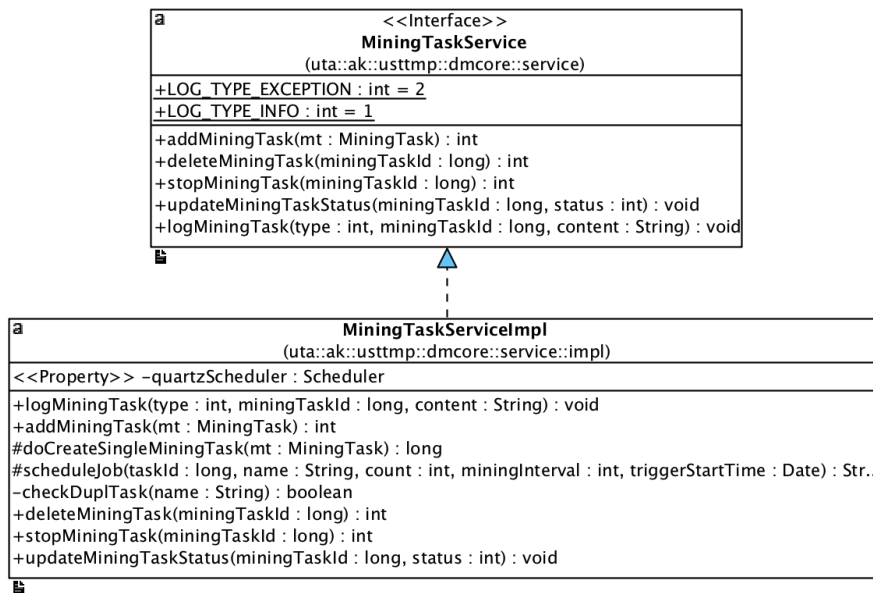


Figure 4.5. Class diagram of MiningTaskService

4.3. Task execution process

4.3.1. Quartz framework

Quartz is a richly featured, open source job scheduling library by OpenSymphony. It can be integrated within virtually any Java application - from the smallest stand-alone application to the largest e-commerce system [Quartz, 2016]. As a job scheduling system, Quartz not only can be integrated into other systems but it can also be running alone. It has light-weight and flexible features which enable users to use it by simple installation and configuration. Quartz has fault tolerance and persistence of scheduled job. Jobs can be resumed even after server crash or restart.

4.3.2. Process component

As we mentioned in Section 3.2.5, the Interface design model will be adopted to enable the system extensible and support secondary development of different topic algorithms. Therefore, the whole mining executing process is divided into three parts described by three interfaces. There are three interfaces shown in Figure 4.6.

The first one is PreprocessComponent which contains an interface method definition of preprocessing. The method allows two input parameters: MiningTask, representing Java bean of mining task, and rawTextList which is a List value containing RawText Java bean. The method processes incoming raw text list and returns a text list being preprocessed.

The second one is called MiningComponent which is a method definition for topic mining. It allows two incoming parameters: the MiningTask bean and a text list. The output value is a list of topics.

The third one is TrackingComponent which defines a method for topic evolution tracking. The incoming parameters are mining task bean list of topics in prior time intervals and list of topics in post time intervals. It returns a list of calculated topic evolution relations.

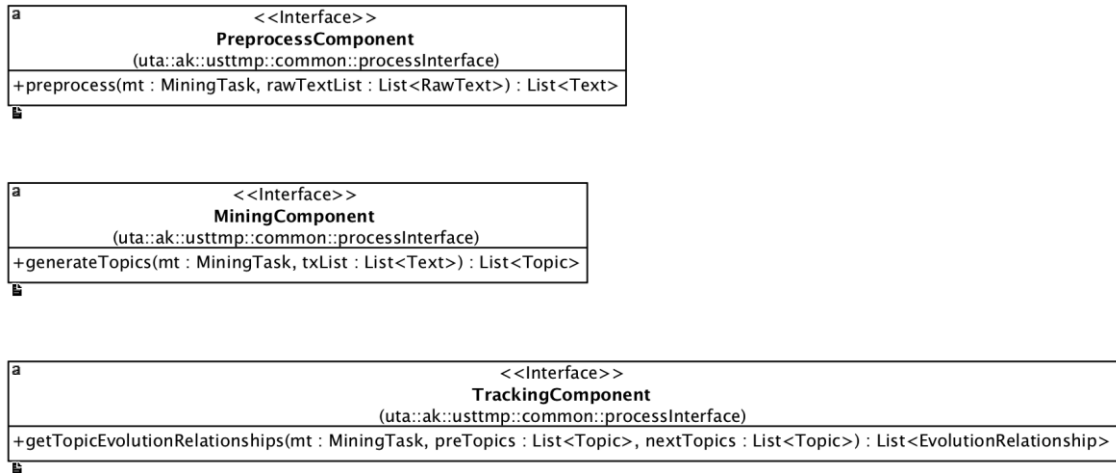


Figure 4.6. Three interfaces of process component

QuartzMiningJob is a class for undertaking task execution, which implements a Job interface in Quartz library. A class implementing the Job interface can be run by the Quartz framework automatically according to the schedule set by the user. The Execute method in the QuartzMiningJob will be run when a task is triggered on time. The method covers the whole process of text preprocessing, topic mining and evolution tracking. MiningTaskService is used to achieve information about the mining task. Three interface components, preprocessing, mining and tracking, will be invoked in a sequence. Results return by every steps will be stored in a database, including a list of text preprocessed, a list of topics and a list of topic evolution relationships.

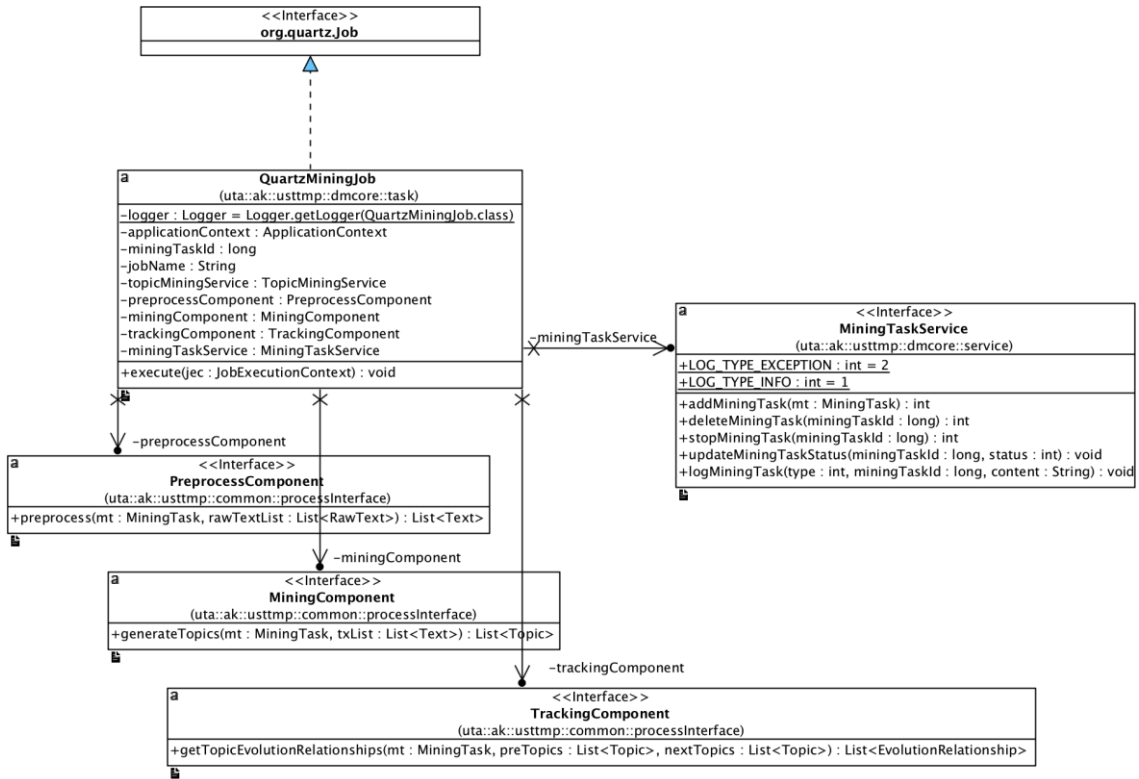


Figure 4.7. Class diagram of QuartzMiningJob

4.3.3. Text preprocessing

The real implementation class of PreprocessComponent is EnglishTwitterPreprocessService since we only adopt English as a language for text mining, which means we only handle English texts in this system. The implementation class processes raw texts by word segmentation and removing stop-words (Section 3.4.3).

Firstly, split texts are retrieved from database according to time and data tag. Secondly, text is split into single words by blank space or punctuation. Then stop-words will be removed according to a stop-word list integrated in the source code. Stop-words includes common English vocabularies (pronoun, preposition, etc.) and common media vocabularies, such as media name, press name and vocabularies about news report. If text is totally made of stop-words, it will be ignored. Finally, a list of processed texts will be returned.

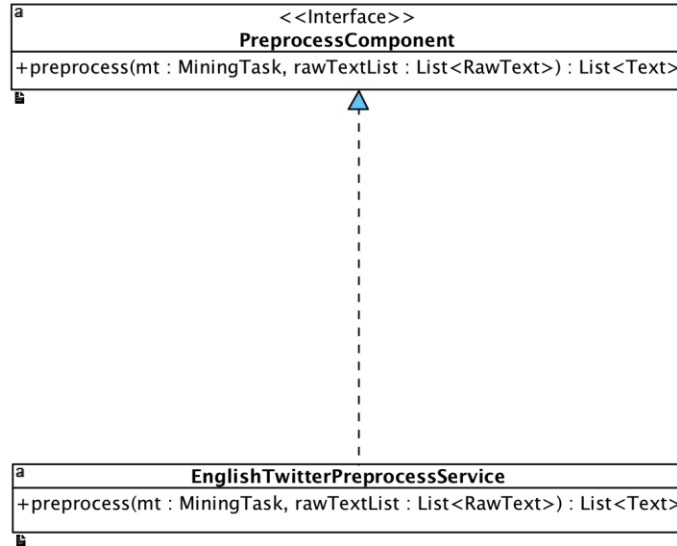


Figure 4.8. Preprocessing interface and implementation

4.3.4. Topic mining

MiningComponent is implemented by UsttmpDefaultTopicMiningService class. The internal logic is the LDA topic model. Firstly, preprocessed texts will be retrieved from data according to time and data. Then, texts will be sent into USTTMPTopicModel class via format transition. Here, the main parameters are topic number in every time interval, number of key words in each of topics and time of texts collected in this time of task execution, all of which are acquired from the mining task. Finally, topics mined will be returned after a series of operations of iteration, regression and calculation.

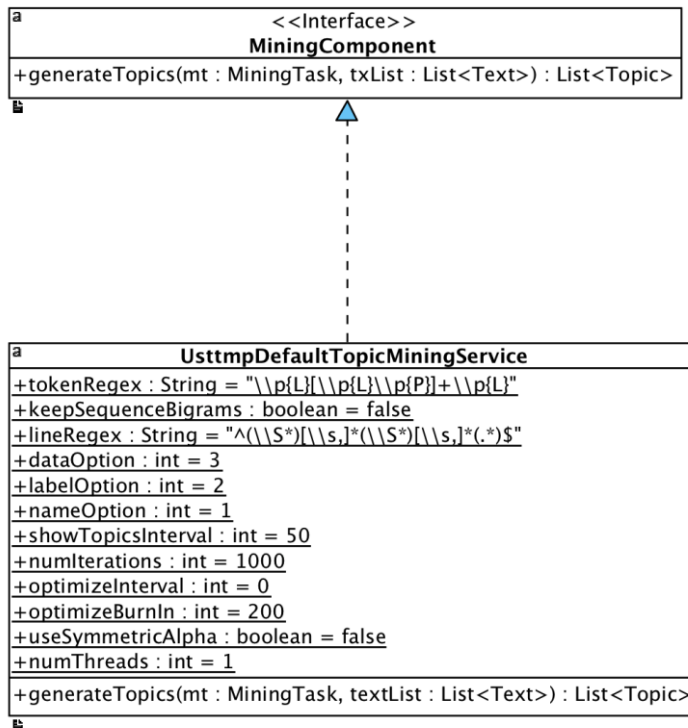


Figure 4.9. Mining interface and implementation

Here we create USTTMPTopicModel class based on ParallelTopicModel which is a class in Mallet library. It provides basic and original LDA methods for text mining. USTTMPTopicModel is instantiated by choosing three parameters: number of topics in one-time interval, alpha and beta. The last two are specific parameters for LDA model, which have an impact on end results of topic mining to some degree.

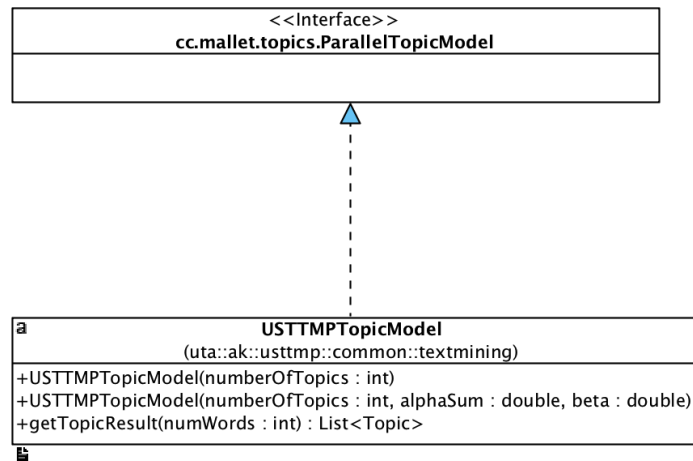


Figure 4.10. Core class of LDA algorithm and extension

4.3.5. Topic evolution tracking

TrackingComponent is implemented by AssociationFilterEvolutionService class based on algorithm in Section 2.3.2. Firstly, prior topics and post topics in the time interval of current task execution will be retrieved. Then, associations will be built pairwise between prior topics and post topics. After that, topic associations will be filtered by three levels of rules according to Section 2.3.2. Valid topic evolution relations will be returned in the end.

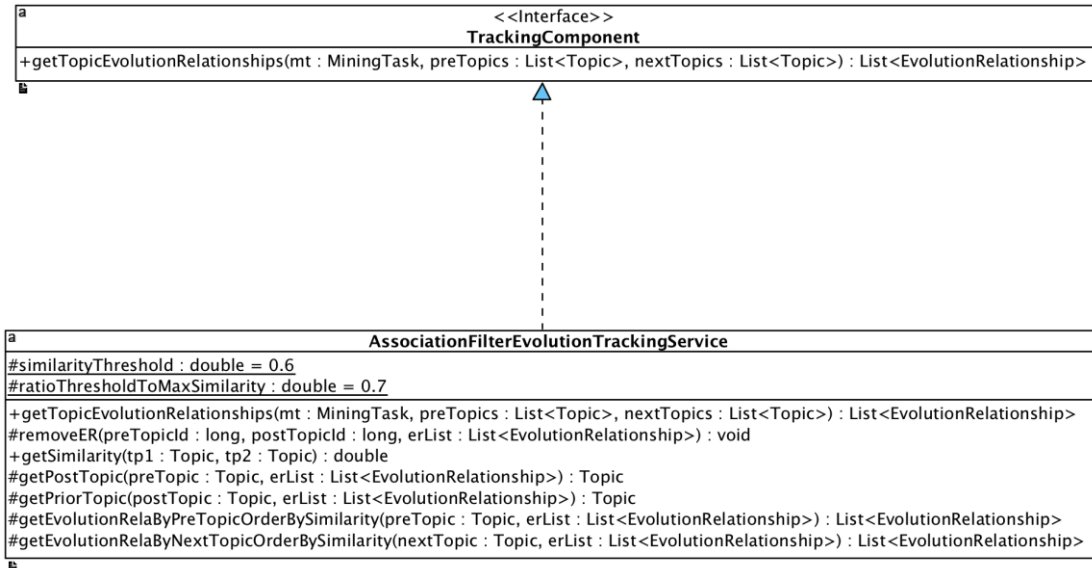


Figure 4.11. Tracking interface and implementation

4.4. Result presentation

4.4.1. D3 framework

D3.js is a data visualization JavaScript library. Compatible with W3C standard, D3.js widely supports HTML, SVG, and CSS. It shields differences between browsers and enables developers to make cool diagrams with simple code. With the help of D3.js tool, we are able to transit result data of text mining to a diagram easily on a web page.

D3 allows you to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document. For example, you can use D3 to generate an HTML table from an array of numbers. Or, you can use the same data to create an interactive SVG bar chart with smooth transitions and interaction. With minimal overhead, D3 is extremely fast, supporting large datasets and dynamic behaviour for interaction and animation. [D3.js, 2016]

4.4.2. Result visualization of mining and evolution tracking

A user can view the result of mining and evolution tracking of a task from main view of the system. Data is visualized as diagram so that user can view results intuitively. The diagram below shows a sample result of topic mining and evolution tracking of social texts from Twitter.

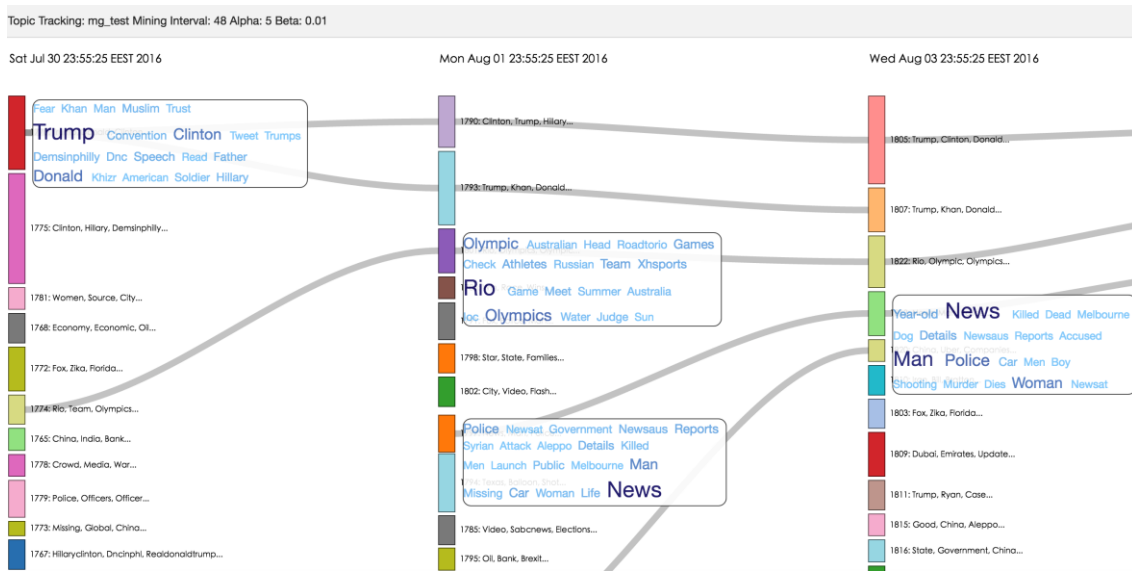


Figure 4.12. General view of mining and tracking results

Vertical bars represent topics mined in a period of time interval. Texts on diagram show basic information of current task. Labels at the top of diagram show task execution time. Length of bar indicates heat of topic. Longer bar means higher appearing probability of topic in this group of topics. White box attached to topic bar shows all of key words that topic contains. Bigger font of words and deeper colour mean higher weight of a word in the group of key words. The user can adjust the position of a topic bar in vertical direction for better viewing effect. Links between topic bars represent topic evolution relations between topics. Relations can be inheritance, extinction, merge and split as mentioned in Section 2.3.2.

4.5. Exception View

Whatever exceptions happen in the process of task execution will be logged into database, including critical information. Users can access exception logs of a task from the main view of the system. The table below shows the view of exception logs of a task. The list shows the occurrence time of logging, the type of log (exception or critical info) and specific information, such as a stack trace or a system error report.

Exceptin list of mining task: TwitterNews-GapThreeDays

Occurrence Time	Type	Info
2016-09-15 04:00:09	Exception	java.lang.RuntimeException: rawTextSet is empty. at uta.ak.usttmp.common.textmining.TextListIterator(TextListIterator.java:38) at uta.ak.usttmp.common.service.impl.TopicMiningServiceImpl.miningTopic(TopicMiningServiceImpl.java:243) at uta.ak.usttmp.common.service.impl.TopicMiningServiceImpl.generateTopics(TopicMiningServiceImpl.java:77) at uta.ak.usttmp.dcore.task.QuartzMiningJob.execute(QuartzMiningJob.java:122) at org.quartz.core.JobRunShell.run(JobRunShell.java:202) at org.quartz.simpl.SimpleThreadPool\$WorkerThread.run(SimpleThreadPool.java:573)
2016-09-15 04:26:36	Exception	java.lang.NullPointerException at uta.ak.usttmp.common.model.Topic.toString(Topic.java:127) at uta.ak.usttmp.dcore.task.QuartzMiningJob.execute(QuartzMiningJob.java:268) at org.quartz.core.JobRunShell.run(JobRunShell.java:202) at org.quartz.simpl.SimpleThreadPool\$WorkerThread.run(SimpleThreadPool.java:573)

Showing 1 to 2 of 2 entries

Figure 4.13. List of exceptions in the task process

5. Test and Evaluation

This chapter describes testing of the system. The effectiveness of LDA for social texts and the real running result of system will be validated. Therefore, some tests will be conducted, including coverage of LDA for social texts, performance of the system and presentation of the system functions in a real environment. Hereon, social texts from Twitter are chosen as the data source.

5.1. Test environment

The platform of the test is on a laptop computer. The hardware and software environment are described below.

CPU	Intel(R) Core(TM) i3-3110M CPU @ 2.40GHz (2 cores)
Graphics Card	Intel GMA HD 3000
Memory	8 GB
Harddisk	500 GB, 5400RPM

Figure 5.1 Hardware environment of the system

Operation system	Ubuntu 14.04.1
Runtime environment	Java 1.8.0_101 Node.js 0.10.25
Web Server	Glassfish 4.1 Tomcat 8.0.36
Development framework	Spring 4.2.5 Express 4.13.4

Figure 5.2 Software environment of the system

The system is Java-based. The Console of the system is continuously running on a Tomcat server. Mining Core is running on Glassfish. Text Receiver interface is running in the environment of Node.js.

5.2. Test runs and results

5.2.1. Topic coverage test

Social texts cover all aspects of our life, entertainment, sport, politic, etc. Topic coverage test aims to validating LDA effectiveness for social texts using extensive contents in social networks.

We collected 20 categories of news texts in different fields from Twitter. Every category contains 300 – 600 pieces of tweet. All data have been preprocessed, including removing abnormal characters, punctuations, stop-words and word segmentation.

	Topic category
Economy theme	G20 Summit
	USA economy
Edu theme	Higher Education
	Remote Education
Entertainment theme	Angry Bird Movie
	Overwatch Game
Health theme	Cancer
	AIDS
Literature theme	Shakespeare
	Japan Literature
Military theme	US Armed Force
	South China Sea
Politics theme	USA Election
	ISIS
Sport theme	Euro Cup 2016
	NBA Final
Tech theme	NASA
	VR
Travel theme	Lonely Planet
	Hiking

Figure 5.3. Preset categories of topics

After preparation of test data, three test persons were invited to attend the test. They were asked to do a few evaluations against mining results. They needed to judge whether topics mined can be categorized in those 20 categories above. The basic principle of the test is their subjective judgements. They are encouraged to distinguish topics by first impression as real viewers of social network.

There were four guidelines for test persons to judge topics but they didn't need to obey these rules very strictly. Subjective first feeling is fundamental.

1. If the test person considers two or more categories covered in a topic, then this topic is invalid.
2. If the test person considers there are many noise words or stop-words occurring in a topic, then this topic is invalid.
3. The test person observes the first half of key words which hold higher weight relatively. If the test person can judge which category it belongs to, then the topic can be categorized even though there is noise data in the last half of key words.
4. If the test person can not judge a category of a topic by the first half of key words, then the whole key words are observed. If the test person still can not judge it, then this topic is invalid.

After recognition, if and only if a topic can be categorized in one single category among those 20 categories, we call the topic a **recognized topic**. Otherwise, the topic is an **invalid topic**.

In one-time task execution, we assume that the number of recognized categories (one topic covers it at least) is c ; the number of invalid topics is u and the total number of topics mined is m . So coverage and validity ratio can be expressed as below.

$$coverage = \frac{c}{20}$$

$$validity\ ratio = \frac{m - u}{m}$$

Tests are conducted by four different groups of parameters which includes topic number in one task and number of key words contained in a topic. Raw test results, coverage and validity ratio are shown as below.

No.	1	2	3	4	5	6
Invalid topic	3	3	5	6	2	1
Uncovered category	3	4	5	7	3	1
Total num of topics	20	20	20	20	20	20
Num of key words	10	10	10	10	10	10
Coverage	85.00%	80.00%	75.00%	65.00%	85.00%	95.00%
Validity Ratio	85.00%	85.00%	75.00%	70.00%	90.00%	95.00%
Average Coverage	80.83%					
Average Validity Ratio	83.33%					

Figure 5.4. Test results of 20 topics and 10 key words

No.	1	2	3	4	5	6
Invalid topic	4	6	4	7	6	9
Uncovered category	1	2	0	0	1	4
Total num of topics	30	30	30	30	30	30
Num of key words	10	10	10	10	10	10
Coverage	96.67%	93.33%	100.00%	100.00%	96.67%	86.67%
Validity Ratio	86.67%	80.00%	86.67%	76.67%	80.00%	70.00%
Average Coverage	95.56%					
Average Validity Ratio	80.00%					

Figure 5.5. Test results of 30 topics and 10 key words

No.	1	2	3	4	5	6
Invalid topic	1	2	9	1	1	3
Uncovered category	3	2	9	3	2	3
Total num of topics	20	20	20	20	20	20
Num of key words	20	20	20	20	20	20
Coverage	85.00%	90.00%	55.00%	85.00%	90.00%	85.00%
Validity Ratio	95.00%	90.00%	55.00%	95.00%	95.00%	85.00%
Average Coverage	81.67%					
Average Validity Ratio	85.83%					

Figure 5.6. Test results of 20 topics and 20 key words

No.	1	2	3	4	5	6
Invalid topic	5	8	5	8	4	7
Uncovered category	1	1	0	3	2	1
Total num of topics	30	30	30	30	30	30
Num of key words	20	20	20	20	20	20
Coverage	96.67%	96.67%	100.00%	90.00%	93.33%	96.67%
Validity Ratio	83.33%	73.33%	83.33%	73.33%	86.67%	76.67%
Average Coverage	95.56%					
Average Validity Ratio	79.44%					

Figure 5.7. Test results of 30 topics and 20 key words

The average coverages and average validity ratios are in the table below.

	20 topics – 10 key words	20 topics – 20 key words	30 topics – 10 key words	30 topics – 20 key words
Average Coverage	80.83%	81.67%	95.56%	95.56%
Average Validity Ratio	83.33%	85.83%	80.00%	79.44%

Figure 5.8. Comparison of average coverages and average validity ratios in 4 groups of parameters

According to Figure 5.8, we can see both coverage and validity are nearly above 80%. It is definitely acceptable since most of preset categories are covered by mining results.

Meanwhile, increasing the number of topics in tasks will significantly raise coverage but validity ratio will decrease to small extent. The number of key words has no obvious effect on results. It also implies that a few of high-weight key words might be meaningful enough to viewers.

5.2.2. System test

The system test will validate three main parts of the system using a real data stream, including receiving social text, topic mining and evolution tracking. It tests whether the system operate well entirely as expectation.

For a data source of real social texts, an independent web crawler will be adopted. It is running continuously, collecting and sending news texts of a whole day to Text Receiver interface by the mid-night of everyday. The crawler gets news from above 100 medias'

Twitter accounts. The language of texts is English. Most of them are from British, American and Chinese news medias.

Two mining tasks are created and their running results are observed. The parameters of every task are shown below. One task executed every three days and the other one executed per week. The LDA parameters, α and β , are set to 5.00 and 0.01.

All mining tasks						
Task Name	Data tag	Start Time	End Time	Time interval (hour)	Status	Progress
TwitterNewsEvolutionTracking-Per3Days	TwitterNews-GapThreeDays	Aug 10, 2016 12:00:09 PM	Sep 10, 2016 12:00:09 PM	72	Completed	100%
TwitterNewsEvolutionTracking-PerWeek	TwitterNews-GapThreeDays	Aug 8, 2016 12:00:06 PM	Sep 18, 2016 12:00:06 PM	168	Completed	100%

Figure 5.9. List of two mining tasks

The two tasks ran smoothly more than one month. Finally, the two tasks were completed successfully. Results are presented below.

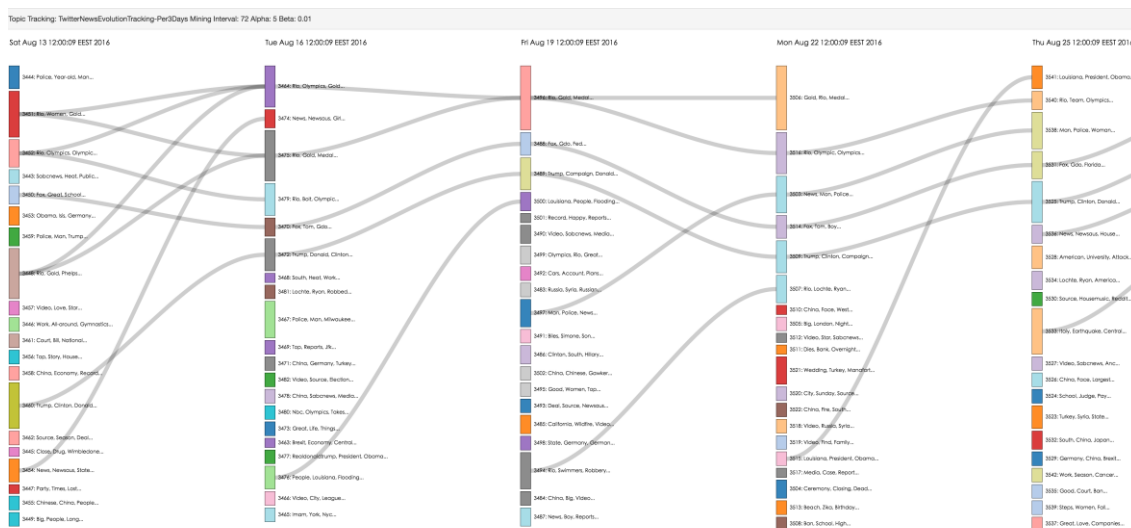


Figure 5.10. General view of the task of Twitter tracking per 3 days

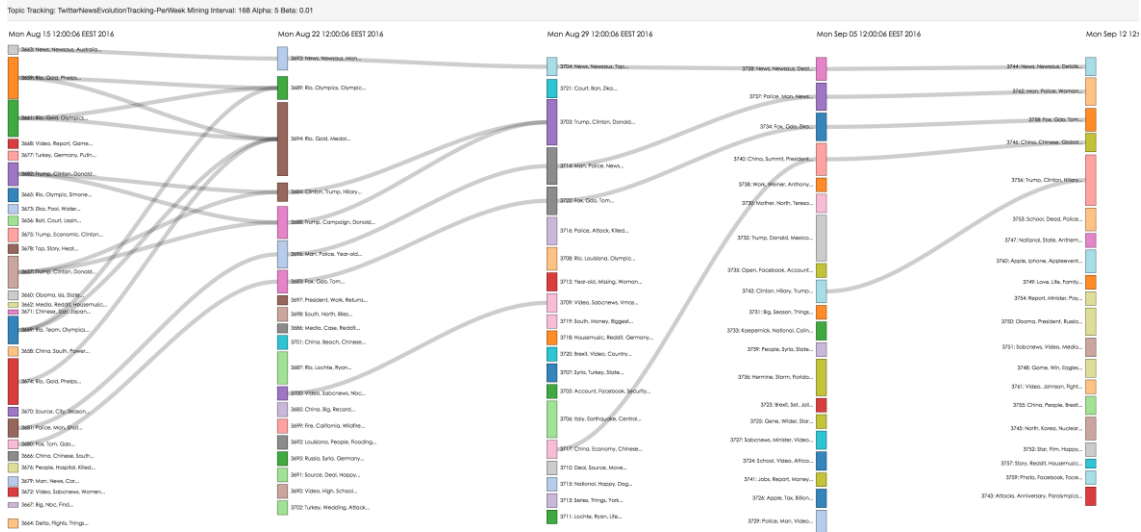


Figure 5.11. General view of the task of Twitter tracking per week

As the diagrams show, the tasks executed every time interval, mined topics from tweet texts and tracked evolution relations. There are obviously more topic evolution relationships in the first time interval. An important factor is the Rio Olympic Games being held in the middle of August. Therefore, many topics about the Olympic Games are mined and evolution relations are complex and massive. This result fits with a realistic situation. Meanwhile, five types of evolution relations have been detected as mentioned in Section 2.3.2



Figure 5.12. Inheritance of topics

In Figure 5.12, topics about US Election were always being talked and kept relatively high heat among all topics. The US Election topics are inherited by similar topics and continue along with timeline.

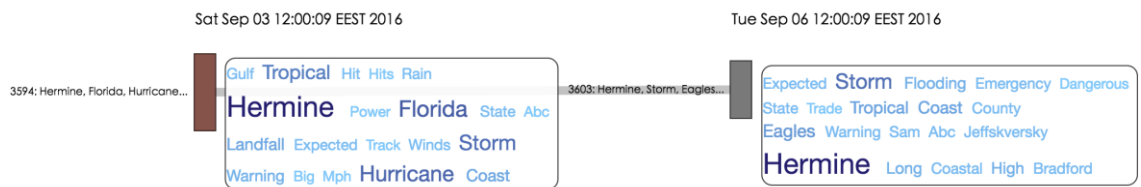


Figure 5.13. Creation and extinction of topics

In Figure 5.13, a topic about Florida hurricane is created at Sep 3rd and became extinct at Sep 6th because of no post topic was following it.

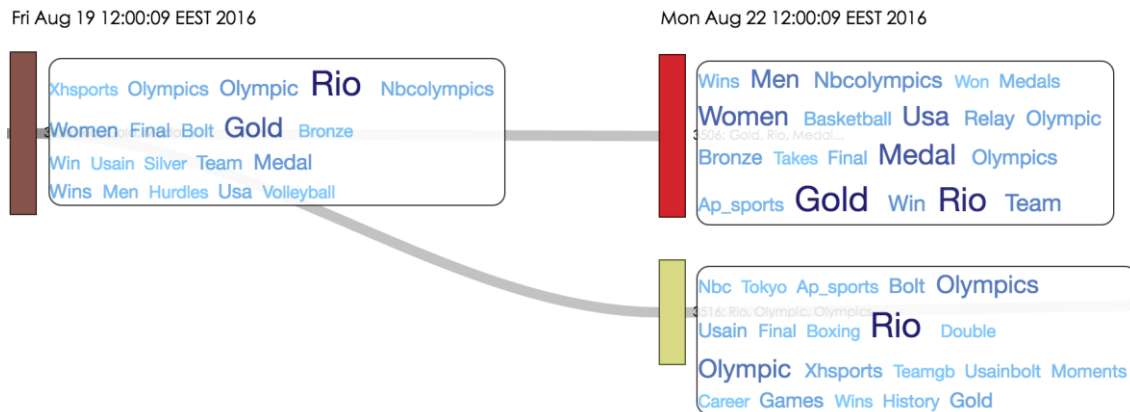


Figure 5.14. A split of topics

In Figure 5.14, a topic about the Rio Olympic Games was split into two topics. One focused on basketball teams and medals. The other one followed the final ceremony and the career of athlete Usain Bolt.

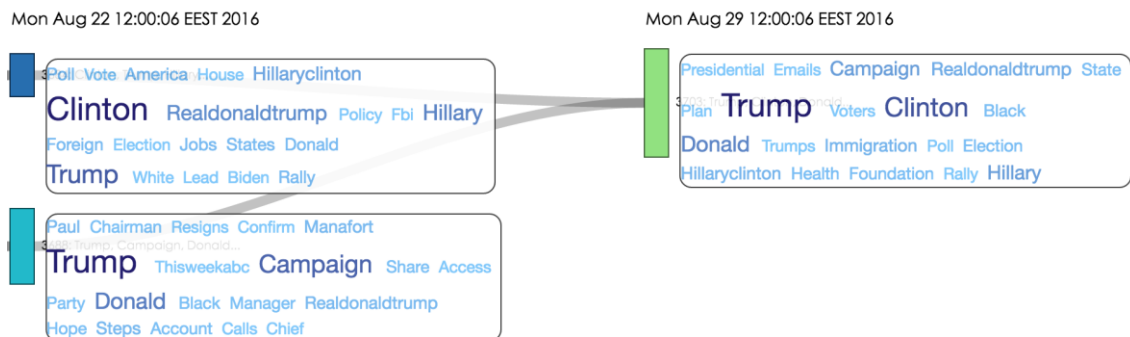


Figure 5.15. A merge of topics

In Figure 5.15, two topics about Clinton and Trump's policies in election merged into one topic about election campaign.

5.2.3. Performance

A vast number of social texts emerging in cyberspace require efficiency. A performance test will measure response speeds of the system under different levels of load. It is an essential standard of system applicability. We divided it into four parts, response speeds of text receiving, preprocessing, topic mining and evolution tracking. Test results are shown below.

NO.									
Num of tweets sent	372	1503	3411	3983	5842	8349	10634	11871	
Total time (second)	32	122	287	332	487	655	858	980	
Time of per tweet sent	0.0860	0.0812	0.0841	0.0834	0.0834	0.0785	0.0807	0.0826	
Average time	0.0825								

Figure 5.16. Average time of receiving a tweet

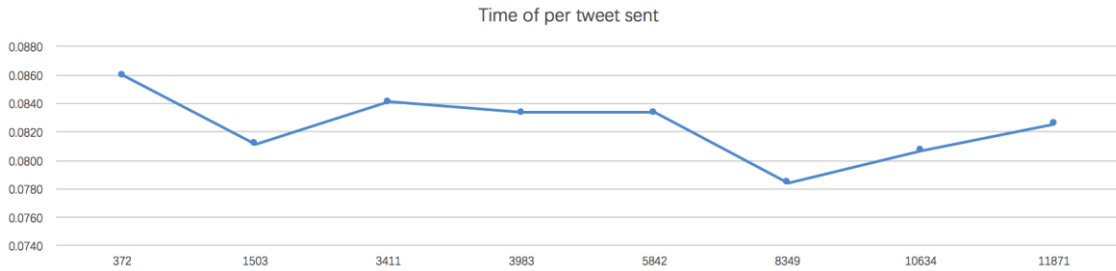


Figure 5.17. Diagram of average time of receiving a tweet

As Figure 5.17 shows, number of tweets being received by the interface is increasing from 372 to 11871. The average time for receiving per tweet was stable between 78ms and 86ms. The speed is acceptable because the main program functionality for receiving texts is a database operation and performance of test server was limited.

NO.	1	2	3	4	5	6	7	8	9	10
Total num of text	324	1075	1699	2165	3402	4028	6824	9529	19403	21810
Preprocess time	36	120	182	187	280	219	370	501	1009	1146
Mining time	3	3	4	6	4	5	7	7	25	41
Evolution tracking time	1	2	2	1	1	1	1	1	1	1
Entire process time	40	125	188	194	285	225	378	511	1035	1188
Preprocess time (per tweet)	0.11111	0.11163	0.10712	0.08637	0.08230	0.05437	0.05422	0.05258	0.05200	0.05254
Mining time (per tweet)	0.00926	0.00279	0.00235	0.00277	0.00118	0.00124	0.00103	0.00073	0.00129	0.00188
Evolution tracking time (per tweet)	0.00309	0.00186	0.00118	0.00046	0.00029	0.00025	0.00015	0.00010	0.00005	0.00005
Entire process time (per tweet)	0.12346	0.11628	0.11065	0.08961	0.08377	0.05586	0.05539	0.05363	0.05334	0.05447
Average Preprocess time	0.07643									
Average Mining time	0.00245									
Average Evolution tracking time	0.00075									
Average Entire process time	0.07965									

Figure 5.18. Average time (per tweet) of each process of mining task

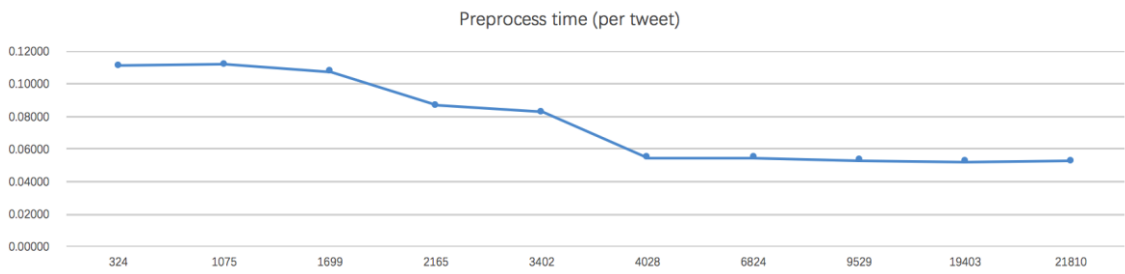


Figure 5.19. Diagram of average time of preprocessing

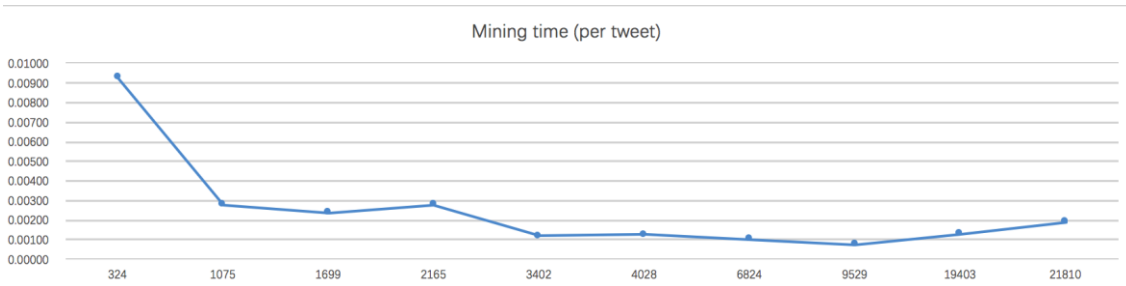


Figure 5.20. Average mining times

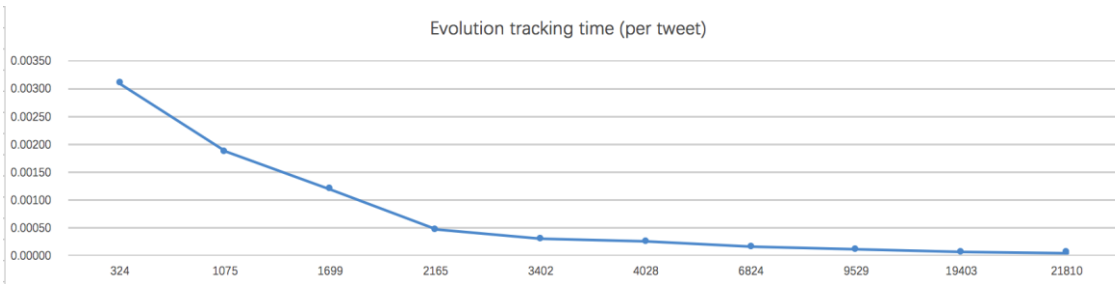


Figure 5.21. Average evolution tracking times

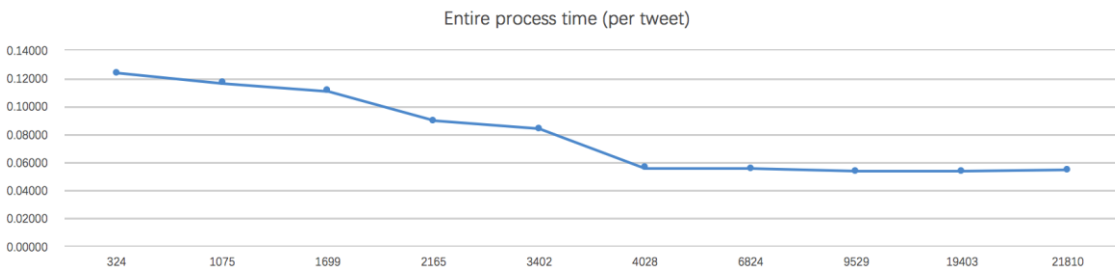


Figure 5.22. Average complete task times

Average processing times per tweet in the entire mining task and three sub-processes are shown in Figures 5.19 – 5.22. We can see that the preprocessing time accounts for the most part of the entire time and the evolution tracking is the least part. The average time per tweet for the entire process is around 60ms. So the system can approximately handle 10000 tweets in 10 minutes. Meanwhile, average processing times significantly decrease with the growth of number of texts. It fits with general performance regulations of common systems. It's fairly a good result, considering the limited computation power of the test platform.

6. Evaluation and discussion

6.1. Evaluation of system

A vast amount of social texts are available in the cyberspace. The amount of data is constantly increasing. We urgently need some tools to extract information from massive social data. Text mining is a relatively new and exciting way to solve the information overload problem by using techniques of data mining. Text mining involves the preprocessing of document collections, the storage of the intermediate representations, the techniques to analyse these intermediate representations, such as distribution analysis, clustering, trend analysis, and association rules, and visualization of the results [Feldman and Sanger, 2006].

Creating an entire and applicable system to mine text topics in social networks is beneficial to helping cyber users or media workers to extract useful information quickly and conveniently. The system has covered several main parts of text mining, including text collection, topic mining, evolution trend tracking and visualization. It helps users to detect occurrences and evolution trends of social topics. Thus, it will be beneficial to understanding effects of social medias on the public.

Algorithms are the core of solving topic mining and tracking and information extraction. Making use of existing algorithms to implement mining and tracking functions is an essential part of the system. By manual identification, test results of the system have been recognized by real viewers successfully. LDA topic mining has been proven to be effective for social texts.

Human-centric text mining emphasizes the centrality of user interactivity to the knowledge discovery process [Feldman and Sanger, 2006]. As a consequence, text mining systems need to provide users with a range of graphical approaches for interacting with data [Feldman and Sanger, 2006]. This demands designers of text mining systems to create more sophisticated visualization approaches to facilitate user interactivity.

By diagrams in Section 5.2.2, we can see that mined topics and detected evolution relations are very meaningful to humans. Evolution tracking explicitly reveals associations between prior topics and post topics. Meanwhile, length of the bar, font size and color of the word show heats of topics and weights of key words in a user friendly manner. These illustrate that the system is helpful to people to track trends of social texts.

Progress has been made steadily by researchers in the field of information retrieval. As to the field of topic mining, many useful algorithms have been created and put into

practice within approximate in a decade, from TF-IDF [Salton and McGill, 1986], PLSI [Hofmann, 1999] to the important LDA model [Blei, Ng and Jordan, 2003]. In addition, there are many improved versions of basic algorithms for different application targets. Therefore, there is a need to handle requirements of different algorithms.

The system does not implement several topic mining algorithms but it provides the support of development by adopting the Interface design pattern. Developers can make implementations of the three process components (Section 4.3.2) so as to construct concrete algorithms and enable users to choose respective options in the web page by configurations.

In the performance test (Section 5.2.3), mining and tracking processes are very fast in real tests. The largest consumption of resources comes from accessing and writing the database. Average process times significantly decrease in Figure 5.19-5.22 with the growth of number of texts. It fits with general performance regulations of common systems and the times became stable after a period of running. It obviously means that database has become the bottleneck and affected further improvement of execution efficiency.

Although the computer for deployment has no high performance, system test still performs favourable effects. All tasks can run successfully and smoothly around the time. For mining tasks with relatively longer intervals, there were no performance issues in tests. More frequent tasks have not been adopted to test the ability of system handling high load.

There are some systems to mine topics and demonstrate the evolution visually. For example, D-VITA, which is mentioned in Section 1, is a system to support users exploring and interacting with numbers of documents. It can extract topics hidden in the texts and highlight the evolution of selected topics. However, it only works on prepared and ready-made data. The main difference between this system and other similar systems is the ability to handle dynamic data. This feature is significant to receive real-time texts from social platforms so the system is able to react with changes of hot topics discussed timely. In comparison with common systems of public opinion monitoring, this system mines topics unbiasedly and discovers topic evolution from general aspects. Systems of public opinion monitoring aim at certain topics (set by users usually) so as to monitor or censor contents people focus on.

In general, the system operates well and smoothly in a real environment. Existing algorithms have been demonstrated to be effective for social texts. The system

successfully covers the whole process of social text mining, including receiving real-time texts, topic mining and continuous evolution tracking as expected. Meanwhile, an applicable user interface is provided for control and viewing results.

6.2. Discussion of system development

The system development includes several stages, including initial technology validation, architecture design, iterative development and detail improvement. The main process of development lasted approximately 4 months.

Firstly, LDA algorithm in Mallet library (Section 3.2.4) had been validated preliminarily on some sample data. Then, the system architecture design was drawn according to system goals and social text features, in combination with writer's developing experience as well.

For the development of the system, iterative model [Larman and Basili, 2003] was adopted. Frameworks of each system were constructed. After assuring they worked and communicated well, Console was developed first. When user interface had been built, the development continued to Mining Core. After testing core functions of topic mining and evolution tracking successfully, the Text Receiver interface was constructed continuously. Finally, details were improved and bugs were fixed.

The system architecture is Java-based because Java is a popular language being applied in many areas widely and there is a favourable software ecosystem built on it. It means that there are various perfect frameworks and tools developed using Java to help developers to build their applications, including most of data mining platforms', language processing tools' and mining algorithms' implementation. Considering the convenience from users' aspect, B/S application is easy to use. Users do not need to install client software on local computers and just operate the system directly by browsers. Therefore, Java EE, which is one of the most popular B/S architectures, will facilitate the system development. In addition, Java provides native support to multiple threads, which benefits parallel running of multiple mining tasks.

One of the key parts of the system is integration of algorithms and the system. Although Mallet library already has a ready-made LDA algorithm, the data format of it was not compatible with our system. Mallet only can handle static text files in local computers and output text files as well. So there was a demand to transfer the data format to what we need. The source code of Mallet was investigated and core classes of LDA were extracted. After packaging core classes, we used them in our system.

The other key part was the visualization of results. A friendly and concise diagram will benefit user viewing of mining results. The core of this part is a transition from data to diagrams. As HTML5 and JavaScript become more and more popular and powerful, gorgeous diagrams or graphics could be implemented on the web pages. So we chose the D3 (Section 4.4.1) as our diagram framework.

The idea of diagrams for topic tracking is inspired by the Sankey Diagram [Bostock, 2012]. However, links between bars are weight-fixed here. The length of bars is used to represent heats of topics. The box containing key words is from common tag clouds. In addition, the layout of the diagram has been optimized to make bars connected to links more concentrated so that users can view results distinguishingly. However, the way of adjusting layouts is not perfect due to time limitation. There are still some links crossing.

For the text interface, its inner logic is relatively simple but there are always demands of high concurrency load on it in real application. Thus, Node.js provides a fine approach to handle these problems with lower resource costs. Meanwhile, from the writer's actual perception, Node.js is easy to learn and development efficiency can be enhanced by its JavaScript features. Node.js has been widely used to build data-intensive applications so far because it is convenient to develop quick-responding and easy-extending web application.

Inner communication in the system is via Webservice interfaces. Three sub systems adopt general interfaces and the same message format. Considering the possibility of frequent interaction and monitoring, using unified interfaces is beneficial to lower difficulty and complexity of development.

6.3. Limitation and improvement

In the test, the coverage of LDA is above 80% and the validity ratio is also in the range we can accept. There is still room for improvement. Many duplicated topics in time intervals will make the results of evolution tracking messy to viewers.

As mentioned before, there are various versions of topic mining algorithms. Evolution tracking methods are also keeping advance with the times. Meanwhile, there are demands for the preprocessing of different languages. All of these can be completed by implementing three process component interfaces. We can expect better algorithm performance and more specific preprocessing to languages by development from other researchers.

Although performance figures are acceptable, there is need to improve the performance of the database. In the whole system, the most time-consuming part is the preprocessing because it contains most database operations. As we know, we are in the time of big data. In practice, the number of texts is to be analysed by millions, even billions. How to improve DB performance is a big issue in data-intensive systems. According to real situations, some common DB optimization technologies can be adopted, such as database sharding, using database procedures, database clusters, etc. Some distributed storage systems can also be considered, such as Hadoop.

In the test of the system, we only used default parameters or experience settings. Whether and how they have effects on results have not been validated sufficiently. Meanwhile, we only tested the system for the time up to one month. Therefore, there are more evaluations and longer time tests required in the real environment.

In addition, the system is a prototype so far and there may be a need to change with new requirements. There are also some bugs and some details to be elaborated if we want to put the system into wider practice.

7. Conclusions

This thesis presents an entire and coherent system which conducts topic mining and evolution tracking for dynamic social texts. The system functions include receiving texts from real-time data stream, mining topics, tracking topic evolution timely and visualization of results. Algorithms for topic mining and evolution tracking are chosen from existing research achievements. The thesis also demonstrates the processes of design and development of the system. Tests were conducted to validate functions of the system. Finally, evaluation and discussion were drawn according to test results.

In the last decade, social platforms developed with incredible speed and they play more and more important roles in the life of the public. Mining the meaning of text on social media draws attentions in both humanities and IT areas. Text topic mining is a useful approach to discover information from a massive amount of texts. In general, existing algorithms are effective in mining of topics, but there is more room for timely and dynamic mining of text topic and also lack of applicable work on entire solution.

Some literature has been presented about the social network, social texts and data mining process. So far, LDA (Latent Dirichlet Allocation) is a mainstream algorithm in the area of text topic mining, raised by Blei in 2003. It has been widely applied to topic mining process in various fields. Topic evolution method based on association filter by Qin [Qin and Le, 2015] is a new type of simple, effective topic tracking method and it is easy to implement. It is based on the LDA model. It shows obviously advantages over former ways. Hereon, we chose these two as the methods we integrated in the system.

The system is based on Java EE and B/S architectures. The whole system contains three subsystems, Console, Mining Core and Text Receiver interface. Each of them runs in an independent domain and perform their own functions. The mining task is the core of the system organization. It runs timely and continuously to execute each process of topic mining. Results of topic mining and tracking are persisted in database. Users can access these data and visualize them. Any errors or exceptions happened in mining or tracking processes will be recorded in database. The development achievement has been shown in this thesis, including user interfaces, related frameworks, class diagrams and running results. In addition, the system also supports the secondary development of each process and adopts some technologies, such as Node.js, Mallet, Restful Webservice, etc.

Tests were conducted after the system development. Topic coverage, system functions and performance tests are included. Test results indicated that LDA is effective to topic mining of social texts but there is still room for improvement. Results of evolution

tracking algorithm are meaningful to viewers. The performance figures are acceptable. There is a need to improve performance of the database, especially in the preprocessing.

In general, the system operated well and smoothly in a real environment. It successfully covered the whole process of social text mining, including receiving real-time texts, topic mining and continuous evolution tracking as expected. In addition, an applicable user interface is provided for control and for viewing the results. However, there is still room for system improvement and sophistication, since the system is a prototype. There is a need to change with real requirements if the system is put into practice and a lot of real tests should be performed in order to guarantee it is functioning well.

References

- [Asuncion, Welling, Smyth and Teh, 2009] Asuncion, M. Welling, P. Smyth, and Y. Teh, On smoothing and inference for topic models. In: Proc. of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, 2009.
- [Blei, Ng and Jordan, 2003] David M. Blei, Andrew Y. Ng and Michael I. Jordan, Latent Dirichlet Allocation. Journal of Machine Learning Research, 3, 2003, 993-1022.
- [Bostock, 2012] Mike Bostock, Sankey Diagrams. Available at <https://bost.ocks.org/mike/sankey/>
- [Chen, Xu and Chang, 2011] Songjian Chen, Yabo Xu and Huiyou Chang, A Simple and Effective Unsupervised Word Segmentation Approach, Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011
- [Clifton, Cooley and Rennie, 2004] Chris Clifton, Robert Cooley, Jason Rennie, TopCat: Data Mining for Topic Identification in a Text Corpus. IEEE Transactions on Knowledge and Data Engineering, 16(8), 2004, 949-964.
- [Cui, 2013] Anqi Cui, Study on Public Sentiment Analysis of Events in Microblogs, 2013
- [Cui, Li, Shi, Song, Gao, Tong and Qu, 2011] Weiwei Cui, Shixia Liu, Li Tan, Conglei Shi, Yangqiu Song, Zekai J. Gao, Xin Tong and Huamin Qu, TextFlow: Towards Better Understanding of Evolving Topics in Text. IEEE Transactions On Visualization And Computer Graphics, 17, 2011, 2412-2421.
- [D3.js, 2016] Mike Bostock. Data-Driven Documents. Available at <https://d3js.org>
- [Elshamy, 2013] Wesam Samy Elshamy, Continuous-time Infinite Dynamic Topic Models, 2013.
- [Feldman and Sanger, 2006] Ronen Feldman and James Sanger, Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press, New York, NY, USA, 2006.

- [Freemarker, 2016] Apache software foundation. What is Freemarker. Available at <http://freemarker.org>
- [Gimpel, 2006] Kevin Gimpel, Modeling Topics, 2006
- [Griffiths and Steyvers, 2004] Thomas L. Griffiths and Mark Steyvers, Finding Scientific Topics. Proceedings of the National Academy of Sciences, 101(S1), 2004, 5228-5235.
- [Gunnemann, 2013] Nikou Gunnemann, D-VITA: A Visual Interactive Text Analysis System Using Dynamic Topic Mining, 2013.
- [Han and Kamber, 2006] J. Han and M. Kamber, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, 2006.
- [He, Chen, Du and Jiang, 2015] Jianyun He, Xingshu Chen, Min Du and Hao Jiang, Topic evolution analysis based on improved online LDA model. Journal of Central South University (Science and Technology), 46(2), 2015, 547-553.
- [Hoffman, Blei and Cook, 2009] Matthew D. Hoffman, David M. Blei, and Perry R. Cook. Finding latent sources in recorded music with a shift-invariant hdp. In: Proc. of International Conference on Digital Audio Effects (DAFx), 2009.
- [Hofmann, 1999] Thomas Hofmann, Probabilistic latent semantic indexing. In: Proc. of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, New York, 1999, 50–57.
- [Hu and Chen, 2014] Jiming Hu, Guo Chen, Mining and Evolution of Content Topics Based on Dynamic LDA. Library and Information Service, 58(2), 2014, 138-142.
- [Hu, 2014] Xuan Hu, Research on the Analysis of Microblog Information Based on Text Clustering, 2014.
- [Huagong, 2012] Huagong, Brief introduction to LDA topic model. Available As: http://blog.csdn.net/huagong_adu/article/details/7937616
- [Indarto, 2013] Eko Indarto, Data mining, 2013. Available at <http://recommender-systems.readthedocs.io/en/latest/datamining.html#lit6>

- [Java EE, 2016] Oracle Inc. Java EE at a Glance. Available at <http://www.oracle.com/technetwork/java/javaee/overview/index.html>.
- [Kakkonen, Myller and Sutinen, 2006] Tuomo Kakkonen, Niko Myller, and Erkki Sutinen, Applying latent dirichlet allocation to automatic essay grading, In Proceedings of the 5th international conference on Advances in Natural Language Processing (FinTAL'06), Tapio Salakoski, Filip Ginter, Sampo Pyysalo, and Tapio Pahikkala (Eds.), Springer-Verlag, Berlin, Heidelberg, 110-120, 2006.
- [Larman and Basili, 2003] Craig Larman and Victor R. Basili, Iterative and Incremental Development: A Brief History. *Computer* 36, 6, June, 2003, 47-56.
- [Li, Dai, Lai and Dai, 2011] Xiaoyu Li, Guanzhong Dai, Shuang Lai and Hang Dai, Hot Topic Detection in Chinese Web Forum Using Statistics Approach. In: Proc. of Signal Processing, Communications and Computing (ICSPCC), 2011 IEEE International Conference on Date of Conference, Xi'an, 14-16 Sept. 2011, 1-4.
- [Lienhart, Romberg and Horster, 2009] Rainer Lienhart, Stefan Romberg and Eva Horster, Multilayer PLSA for multimodal image retrieval. In: Proc. of the ACM International Conference on Image and Video Retrieval, New York, 2009, 91-98.
- [Liu, 2010] Xingliang Liu, Dissemination Mechanism of Weibo and Thinking of future development. *News and Writing*, 3, 2010, 43-46.
- [Luo and Li, 2014] Le Luo and Li Li, Defining and Evaluating Classification Algorithm for High-Dimensional Data Based on Latent Topics, 2014.
- [Mai, 2012] Yihua Mai, Chinese Microblog Oriented Social Network Analysis and Application, 2012.
- [Manning, Raghavan and Schütze, 2008] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.
- [McCallum and Kachites, 2002] McCallum, Andrew Kachites, MALLET: A Machine Learning for Language Toolkit, 2002. Available at <http://mallet.cs.umass.edu>

- [Minka and Lafferty, 2002] Thomas Minka and John Lafferty, Expectation-propagation for the generative aspect model, In Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence (UAI'02), Adnan Darwiche and Nir Friedman (Eds.). Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 352-359, 2002.
- [Monay and Gatica-Perez, 2004] Florent Monay and Daniel Gatica-Perez. PLSA-based image auto-annotation: constraining the latent space. In: Proc. of the 12th annual ACM international conference on Multimedia, New York, NY, USA, 2004, 348–351.
- [Nallapati and Cohen, 2008] Nallapati R, Cohen W. Link-pLSA-LDA : A new unsupervised model for topics and influence of blogs, May 23, 2008. Available as: http://videolectures.net/icwsm08_nallapati_plsa
- [Nguyen, Billingsley, Du and Johnson, 2015] Dat Quoc Nguyen, Richard Billingsley, Lan Du and Mark Johnson, Transactions of the Association for Computational Linguistics, 3, 2015.
- [Qin and Le, 2015] Xiaohui Qin and Xiaoqiu Le, Topic Evolution Research on a Certain Field Based on LDA Topic Association Filter. New Technology of Library and Information Service, 31(3), 2015, 18-25.
- [Qin, Dai and Li, 2006] Sen Qin, Guanzhong Dai, Yanling Li, Design and Implementation of Web Hot-Topic Talk Mining Based On Scale-Free Network. In: Proc. of International Conference on Machine Learning and Cybernetics, Dalian, China, 13-16 Aug, 2006, 1184-1189.
- [Quartz, 2016] Terracotta, Inc. What is the Quartz Job Scheduling Library. Available at <http://www.quartz-scheduler.org>
- [Ren, 2015] Tiangong Ren, Research on Retweet Predicting Based on Social Network, 2015.
- [Rodriguez, 2008] WSO2 Inc. Webservice based on REST. Available at <https://www.ibm.com/developerworks/cn/webservices/ws-restful/>
- [Salton and McGill, 1986] Gerard Salton and Michael J. McGill, Introduction to Modern Information Retrieval, 1986.

- [Savage, 2011] N. Savage, Twitter as medium and message. *Communications of the ACM*, 54(3), 2011, 8-20.
- [Shan and Li, 2010] Bin Shan and Fang Li, A Survey of Topic Evolution Based on LDA. *Journal of Chinese Information Processing*, 24(6), 2010, 43-49.
- [Sivic, Russell, Efros, Zis-serman and Freeman, 2005] J. Sivic, B. C. Russell, A. A. Efros, A. Zis-serman and W. T. Freeman, Discovering object categories in image collections. In: *Proc. of the International Conference on Computer Vision*, 2005.
- [Sorensen, 2009] L. Sorensen, User managed trust in social networking comparing facebook, myspace and linkdin. In: *Proc. of 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic System Technology, (Wireless VITAE 09)*, Denmark, 427-431.
- [Steyvers and Griffiths, 2006] M. Steyvers and T. Griffiths, Probabilistic topic models. In: T. Landauer, D. McNamara, S. Dennis, and W. Kintsch (eds.), *Latent Semantic Analysis: A Road to Meaning*. Laurence Erlbaum, 2006.
- [Thiel and Kötter, 2012] Killian Thiel and Tobias Kötter, Creating Usable Customer Intelligence from Social Media Data: Network Analytics meets Text Mining. *Bisociative Knowledge Discovery*, 4, 2012, 263-284.
- [Twitter, 2016] Twitter Inc. REST APIs. Available at <https://dev.twitter.com/rest/public>
- [Wang and Land, 2011] Han Wang and Bo Lang. Online N gram-enhanced Topic Model for Academic Retrieval. In: *Proc. of Digital Information Management (ICDIM), Sixth International Conference on Date of Conference*, 2011, 137-142.
- [Wang, Peng and Wang, 2014] Shaopeng Wang, Yan Peng and Jie Wang, Research of the text clustering based on LDA using in network public opinion analysis. *Journal Of Shandong University(Natural Science)*, 49(09), 2014, 129-134.
- [Wei, 2006] X. Wei and W. B. Croft, LDA-based document models for ad hoc retrieval. In: *Proc. of SIGIR '06 Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 178-185.

- [Xie, 2013] Ming Xie, Social Media at the Age of Big Data. China Media Report Overseas, 9(2), 2013, 11.
- [Xu and Wang, 2011] Ge Xu and Houfeng Wang. The Development of Topic in Natural Language Processing. Chinese Journal of Computers, 34(8), 2011, 1423-1436.
- [Zhao and Zhang, 2012] Xubin Zhao and Changkuan Zhang, Topic Community Mining in Blogosphere Based on LDA. Computer & Digital Engineering, 11, 2012.
- [Zheng and Han, 2013] Lei Zheng and Kai Han, Multi Topic Distribution Model for Topic Discovery in Twitter. In: Proc. of Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on Date of Conference: 16-18 Sept. 2013, 420-425.
- [Zhou, 2013] Erchong, Zhou, Blog Hot Topic Detection And Its Analysis On Public Opinion, 2013.

URLs of source codes:

Common library of the system:

https://github.com/zhangcong2711/usttmp_common.git

Console:

https://github.com/zhangcong2711/usttmp_console.git

Text Mining Core:

https://github.com/zhangcong2711/usttmp_dmcore.git

Text Receiver interface:

https://github.com/zhangcong2711/usttmp_textreceiver.git

SQL script:

https://github.com/zhangcong2711/usttmp_common/blob/master/src/Dump20160722.sql