KWB: An Automated Quick News System for Chinese Readers

Yiqi Bai1, Wenjing Yang1, Hao Zhang1, Jingwen Wang1, Ming Jia1, Roland Tong2, Jie Wang1
1. University of Massachusetts Lowell
2. Wantology

Abstract

We present an automated quick news system called KWB. KWB crawls and collects around the clock news items from over 120 news websites in mainland China, eliminates duplicates, and retrieves a summary of up to 600 characters for each news article using a proprietary summary engine. It then uses a Labeled-LDA classifier to classify the remaining news items into 19 categories, computes popularity ranks called PopuRank of the newly collected news items in each category, and displays the summaries of news items in each category sorted according to PopuRank together with a picture, if there is any, on http://www.kuaiwenbao.com and mobile apps. We will describe in this paper the system architecture of KWB, the data crawler structure, the functionalities of the central database, and the definition of PopuRank. We will show, through experiments, the running time of obtaining PopuRank. We will also demonstrate the use of KWB.

1 Introduction

We are living in the era of information explosion. To help people obtain information quickly, we would want to construct an automated system that collects information and provides accurate summarization to the user in a timely fashion. This would be a system that integrates advanced technologies and current research results on text automation, including data collection, storage, classification, ranking, summarization, web displaying, and app development. KWB is such a system that collects news items from the Internet and provides to the reader summarization and PopuRank of each news item, making it easier for people to obtain critical information quickly.

In this paper we will describe the data collection, data storage, and popular ranking of news items for KWB. Descriptions of the other components will be reported in separate papers, including Labeled-LDA classifier and content extractions. KWB uses a proprietary summary engine to retrieve a summary of up to 600 characters for each news item.

This paper is organized as follows. In Section 2 we will describe related work. We will describe the architecture of KWB in Section 3, the KWB Crawler Framework for collecting news items in Section 4, and the KWB central database in Section 5. We will present the PopuRank formula in Section 6. In Section 7 we will describe KWB and we will conclude the paper in Section 8.

2 Related Work

2.1 Web crawling

Web-crawling technologies are important mechanisms for collecting data from the Internet (see, e.g., (Emamdadi et al., 2014; Lin and Bilmes, 2011; Li et al., 2011; Li et al., 2009; Li et al., 2009; Li and Teng, 2010; Zheng et al., 2008)). The general framework of a crawling is given below:

1. Provide the crawler a seed URL.
2. The crawler grabs and stores the target page’s content.
3. Enter the URLs contained in the target page in a waiting queue.
4. Process one URL at a time in the queue.
5. Repeat Steps 2 to 4.

A crawler is responsible for the following tasks:

1. URL fetching. There are three approaches to grabbing URLs at the target site (initially the
target site is the seed URL): (1) Grab all the URLs in the target site. This approach may waste computing resources of the crawler machines on materials that are not useful for the applications at hand. (2) Grab a portion of the URLs and ignore certain URLs. (3) Grab only what is needed for the current application.

2. Content extraction. Parse the webpage to get the content for the given application. There are two ways to parse a page. One way is to write specific rules for each website, then use a web parsing tool such as Jsoup to extract content. The other way is to write common rules for all websites, such as Google’s content extractor.

3. Visit frequency. If a crawler visits a target website very frequently in a short period of time, then the website may consider it hostile and block the crawler’s IP to stop it. Thus, it is important to not to visit the target website too often in a short period of time to avoid being blocked.

4. Crawler monitoring. We should monitor if the target website blocks a crawler’s request and if the website changes the structures of the webpages.

2.2 Ranking of importance and popularity

There are a number of methods to measure the importance and popularity of an object or a person in a network. For example, the Pagerank mechanism measures the influence and popularity of a webpage (Page et al., 1999) and the Erdős’ collaboration network (Erdős Number Project, 2010) may be used to measure the impact of collaborators (direct and indirect) of Erdős. These measures, however, do not explicitly consider the effect of time in their ranking. To measure the importance and popularity of news items, we need to consider time explicitly. This calls for a new measure and we will present PopuRank to fill this gap.

3 KWB Architecture

KWB consists of five components (see Fig. 1): (1) crawlers, (2) central DB, (3) summary engine, (4) core processing unit, and (5) web display.

Given below are brief descriptions of each of these components:

1. The crawler component is responsible for collecting news items around the clock from over 120 news websites in mainland China.

2. The central DB is responsible for processing the raw data collected from the crawlers, including removing duplicated news items and fetching summaries for each news article.

3. The summary engine is responsible for returning summaries for each new article with different lengths required by applications. This is preparatory technology.

4. The core processing unit consists of three parts: (1) Chinese text fragmentation. (2) News article classifications. (3) Ranking each document according to PopuRank.

5. The web display component is responsible for displaying on a website the news items in each category according to their PopuRanks each day, their summaries, pictures (if there is any), and links to the original news items.

Fig. 2 describes the data flow in KWB system in which each module will operate data and save new attributes.

4 KWB Crawler Framework

The KWB crawler in our system follows the framework of vertical crawling. It can be reused and customized according to the specific layout of a webpage. We observe that news websites tend to have the same structure: an index page and a number of content pages for news items. When grabbing the index page, we may want to set the crawling depth to 1 to stop the crawler from grabbing the URLs contained in the content page. Meanwhile, we also want to remove repeating URLs in the URL queue. The KWB crawler framework uses both specific rules and common rules, depending on the individual crawler for a given website.

The KWB crawler framework consists of the following modules (see Fig. 3):

1. Visual input module: This module allows the user to specify the pattern of the target webpage’s layout. The user may specify two kinds of patterns. The first kind is a regular expression representing what the content the user wants to extract. For example, the regular expression matches the opening and
closing pair of a specific HTML tag, within which is content the user wants to extract. The second kind is an XPath structure of the content that the user wants to extract. For example, Suppose that the user wants to select the content enclosed in all the tags. Then the user can specify an XPath query as.

2. **Webpage rule management.** It manages the webpage rules entered by users, including the following operations: deleting, checking, and updating.

3. **The core crawler cluster.** This cluster consists of the following components:
   (1) Thread pool. It is the set of threads in a multitask system.
   (2) URL pool. It is the database with all the pending URL information when a URL was grabbed. We use Bloom filter to detect duplicate URLs and remove them. The crawler will visit and remove a URL one at a time from the remaining URLs in this pool.
   (a) Pattern pool. It is the database of all the webpage rules entered by users.
   (b) DAO module. DAO (data access object) contains the interface for further operations, including data export and data interface.
   (c) Duplicate removal. It removes duplicate URLs in the URL pool and the patterns in the pattern pool.
4. **The crawler task module.** This module consists of the following submodules:
   
   (1) Priority processing. Some websites are updated more frequency than the others. This module determines which sites need more frequent visits.
   
   (2) Temp grab. Sometimes the user just wants to fetch a website once without paying a return visit. This component handles this type of crawling.
   
   (3) Regular grab. For most websites, the user sets up a schedule to grab them periodically. This component handles this type of crawling.

5. **The supervision module.** This module consists of the following submodules:
   
   (1) Resource control (proxy/account). It is a pool containing all the proxy information and account information. The proxy is used to avoid IP blocking problems, and the account is used to log on certain websites that require signing in, such as twitter and facebook.
   
   (2) Monitoring. It monitors if the crawler functions normally. For example, it monitors whether the target website has blocked the crawler.
   
   (3) Anti-blocking. When the monitoring submodule detects that a crawler is blocked, it decides whether to restart the crawler, change the pattern, or change proxy to avoid blocking.
   
   (4) Managing anti-blocking, exception, and restore rules. This submodule allows the user to manage and change patterns of a website rules. It also determines how often to test if a crawler is still functioning normally.

6. **The program entrance.** This component consists of a crawler controller/entrance submodule, which is responsible for starting the entire system.

We implemented the KWB crawler framework using Java. We use httpclient to connect to a website and get the DOM tree of the page. We use
CSS and Jsoup to parse and extract content. We implemented DAO using mysql and JDBC.

5 Central Database

Data collected from the KWB crawler are raw data. Although duplicate URLs are eliminated by the crawler, the same news article may be collected from different URLs because the it may be reposted on different websites. For each news article we need to retrieve its summary of different length (depending on applications) using a proprietary Chinese text summary engine. These two processes are time consuming. To reduce computations, we create a new database called central DB (see Fig. 4) to remove duplicates and retrieve summaries for raw data collected in every hour.

![Figure 4: Central DB](image)

There are two different types of duplicates in the raw data: (1) exactly the same news items due to reposting; (2) different news items reporting the same news. We will keep the second type of news items, for they report the same event from different angles, which are useful. To identify the first type of duplicates we may compute cosine similarities for all the raw data collected by the KWB crawlers, but this approach is time consuming. Instead, we take a greedy approach to reducing the number of news items that we need to retrieve summaries by eliminating duplicates posted in a small time window. We will further remove duplicates later before computing news classifications.

The central DB retrieves article summaries and detects duplicates in a parallel fashion. In particular, it sorts all the unprocessed raw data in increasing order according to their IDs. These are incremental IDs given to the news items based on the time they are fetched by the KWB crawler framework. Starting from the first news article, repeat the following:

1. Send a request to the summary engine to retrieve summaries of required lengths.
2. Compute the cosine similarities of the article with the news items whose IDs fall in a small fixed time window after this article. If a duplicate is found, remove the one whose ID is in the time window (i.e., with a larger ID), for it is likely a reposting and the news article with a smaller ID may have already had the summaries generated from the summary engine running on a different server.
3. Move to the next news article in the sorted list

The index of the news items stored in the central DB contains, among other things, the following four fields: news title, news URL, image URL, first and last sentence of the news content. We further remove news items that match any of these fields for all pairs of news items. In other words, for each pair of news items, if there is a match on any of these four fields, then remove the article with a larger ID.

6 PopuRank

KWB implements a Labeled-LDA classifier to classify all the news items stored in the central DB. To do so, it needs to segment each news article into a sequence of words, where a word is a sequence of Chinese characters. We show that using Labeled-LDA achieves higher classification accuracy than SVM (Support Vector Machines) for Chinese news items, and we will report this work in a separate paper.

KWB then determines the popularity ranking, called PopuRank, of news items. We observe that the news items that are popular during crawling are indeed the true popular news. In particular, in a given time period, breaking news will be fast reported and reposted online everywhere. In this case, the term frequency (TF) of certain words describing this news will increase sharply. Meanwhile, the document frequency (DF) of certain words describing the breaking news will also increase. We monitor each word (except stop words) in each time frame every day. By monitoring the TF and DF fluctuations of words, KWB calculates PopuRank of the news items collected in each time unit \( u \). The news item with higher PopuRank is more popular. The time unit \( u \) may be changed according to the actual needs and user interests. For example, if we want to determine popular news items in each hour, then we may set \( u \) to be the
unit of hour. The PopuRank of each article remains valid for a fixed number $\ell$ of time frames. For example, we may let $\ell = 24$ or 48, when $u$ is hour. The value of $\ell$ may also be changed.

Let $t_v$ denote the current time frame. Let

$$D_v = \{D_1, D_2, \ldots, D_N\}$$

denote the corpus of all news items collected in this time frame with duplicates removed, where $D_i$ is a news article and $D_i$ contains $N_i$ words in the model of bag of words, denoted by

$$D_i = (w_1, w_2, ..., w_{N_i}),$$

where each word is a segment of two or more Chinese characters after segmentation.

We define the following terms:

1. **Term frequency (TF).** The term frequency of word $w_j$ in $D_i$ in time frame $t_v$, denoted by $tf(w_j, D_i, t_v)$, is the number of times it appears in $D_i$, denoted by $N_{ij}$, divided by $N_i$. That is,

$$tf(w_j, D_i, t_v) = \frac{N_{ij}}{N_i}.$$

Note that if $w_j \notin D_i$, then $tf(w_j, D_i, t_v) = 0$.

2. **Document frequency (DF).** The document frequency of word $w_j$ in the corpus $D_v$, denoted by $df(w_j, D_v)$, is defined as the total number of documents in $D_v$ that contain $w_j$, denoted by $N_j$, divided by the total number of words in $D_v$, denoted by $N$. That is,

$$df(w_j, D_v) = \frac{N_j}{N}.$$

3. **Average term frequency (ATF).** Let $atf(w_j, D_v)$ denote the average term frequency of word $w_j$ in corpus $D_v$. That is,

$$atf(w_j, D_v) = \frac{\sum_{i=1}^{N} tf(w_j, D_i, t_v)}{N}.$$

4. **Term rank (TR).** We define the term rank of word $w_j$ in document $D_i$ in time frame $t_v$, denoted by $tr(w_j, D_i, t_v)$, as follows:

$$tr(w_j, D_i, t_v) = \alpha \cdot tf(w_j, D_i, t_v) + \beta \cdot df(w_j, D_v),$$

where $\alpha \geq 0$, $\beta \geq 0$, and $\alpha + \beta = 1$. For example, we may let $\alpha = 0.6$ and $\beta = 0.4$ to indicate that we place more weight on term frequency over document frequency.

For each word $w_j$ appearing in $D_v$, compute $df(w_j, D_v)$ and $atf(w_j, D_v)$, and keep them for $\ell$ number of time frames.

We now define PopuRank of a document. Assume that word $w_j$ appears in the current time frame $t_v$. Let $T$ denote the following sequence of consecutive time frames, called a window:

$$T = (t_{v+1}, t_{v+2}, \ldots, t_v).$$

At each time frame in this window, we monitor the DF and ATF values for each word. Let $t_v$ be the current time frame. For each word $w_j$ in $D_v$, we have the following two cases:

Case 1: $w_j$ is a new word, that is, it did not appear in the previous time frames in the window $T$, then we compute the TF-IDF values of all the new words in this time frame and mark the top $d$ percent of the new words as popular words.

Case 2: $w_j$ is not a new word. Compute $atf(w_j, t_v)$ and $df(w_j, t_v)$. If the ATF and DF values of word $w_j$ at time $t_v$ suddenly increase $k_1$ and $k_2$ times over the previous average ATF and DF values, respectively, for word $w_j$, denoted by $avgATF(w_j, t_v)$ and $avgDF(w_j, t_v)$, then we will consider the word $w_j$ a popular word, where

$$avgATF(w_j, t_v) = \frac{ATF(w_j, t_v)}{\ell - 1},$$

$$avgDF(w_j, t_v) = \frac{DF(t_v)}{\ell - 1},$$

$$ATF(w_j, t_v) = \sum_{t \in T \setminus \{t_v\}} atf(w_j, t),$$

$$DF(w_j, t_v) = \sum_{t \in T \setminus \{t_v\}} df(w_j, t).$$

To specify the values of $k_1$ and $k_2$, let

$$ratATF(w_j, t_v) = \frac{atf(w_j, t_v)}{avgATF(w_j, t_v)},$$

$$ratDF(w_j, t_v) = \frac{df(w_j, t_v)}{avgDF(w_j, t_v)}.$$

If

$$ratATF(w_j, t_v) > \delta,$$

$$ratDF(w_j, t_v) > \sigma,$$

where $\delta$ and $\sigma$ are threshold values, then we say that word $w_j$ is popular in time frame $t_v$.

Let $H_v$ denote the set of all popular words in time frame $t_v$. We define the PopuRank of news article $D_i \in D_v$ to be the sum of term rank of the popular words in $D_i$ in time frame $t_v$. Namely,
\[
\text{PopuRank}(D_i, t_v) = \sum_{w \in H_v \cup D_i} tr(w, D_i, t_v).
\] (1)

Figure 5: The top 20 news items in all categories in a time frame

Figure 6: The top 20 news items in the sports categories in a time frame

The Fig. 5 depicts the top 20 news items in all categories within one time frame together with a timestamp when a news article becomes popular, while Fig. 6 depicts the top 20 news items in the category of sports in the time frame. The values of parameters for our PopuRank calculation are \( u = \text{hour}, \ell = 24, d = 20\%, \alpha = 0.6, \beta = 0.4, \delta = 1.5, \) and \( \sigma = 1.5. \) The time stamp 1430445600 is the Unix epoch time, which is equal to the total number of seconds since 00:00:00, January 1, 1970 Greenwich time, corresponding to 22:00:00, April 30, 2015 Eastern Time.

Parameters \( \alpha \) and \( \beta \) is related to TR and PopuRank. The value of \( \alpha \) and \( \beta \) are decided by which character, TF or DF, is regarded more important.

The running time of calculating PopuRank on news items in each time frame depends on the numbers of news items waiting to be processed. Table 1 shows the number of news items in each time frame on an average day and the time to compute PopuRank of all news items in each time frame on a server running QEMU Virtual CPU version 1.2.0 with 2.6 GHz and 16 GB RAM.

7 Web Displays of KWB

KWB is an automated quick news system that collects news items real-time from all major Chinese news websites, classifies the news items into 19 categories, and displays on http://www.kuaiwenbao.com news items in each category with summaries and pictures, sorted according to their PopuRank values. We have also implemented KWB in mobile apps (An-
The Android App may be downloaded by entering http://www.kuaiwenbao.com/kuaiwenbao.apk on a web browser of an Android phone. Fig. 10 depicts the web display of KWB, where the left-hand panel is a menu bar of news titles and picture thumbnails. The user simply points their mouse to a particular news title to see the original picture and the summary of the news items on the right-hand panel. The reader may also click the "read the original” button to the URL of the original news article and read it.

KWB classifies all news items into 19 categories. Users may click the menu icon on the upper-left corner to display the menu of categories and select a particular category of interests. Fig. 11 depicts the category menu.

8 Conclusion

We described KWB, an automated quick news system for the Chinese reader. In particular, we described the architecture of KWB, the KWB crawler framework, the central DB, the PopuRank, and the use of KWB. Required by blind reviews, we have removed the URL information of KWB in this version.

References

Dasgupta, Anirban, Kumar Ravi, and Sujith Ravi. 2013. Summarization Through Submodularity and Dispersion. *IBM Journal of research and development* 2.2, pages 159–165.

Emamdadi, Reihaneh, Mohsen Kahani, and Fattane Zarrinkalam. 2014. A focused linked data crawler based on HTML link analysis. *The 4th International eConference on Computer and Knowledge Engineering* (ICCKE), pp. 74–79. IEEE, 2014.

Erdős Number Project (Oakland University).

Lin, Hui and Jeff Bilmes. 2011. A class of submodular functions for document summarization. *Proc. ACL*, pages 510–520.

Li, Xueming, Minling Xing, and Jiapei Zhang. 2011. A Comprehensive Prediction Method of Visit Priori-

ity for Focused Crawler. *The 2nd International Symposium on Intelligence Information Processing and Trusted Computing* (IPTC), pp. 27–30. IEEE, 2011.

Li, Wei-jiang, Ru Hua-suo, Zhao Tie-jun, and Zang Wen-mao. 2009. A New Algorithm of Topical Crawler. *Second International Workshop on Computer Science and Engineering* (WCSE’09), vol. 1, pp. 443–446. IEEE, 2009.

Li, Wei-jiang, Ru Hua-suo, Hong Kun, and Luo Jia. 2009. A New Algorithm of Blog-Oriented Crawler. *International Forum on Computer Science-Technology and Applications* (IFCSTA’09), vol. 1, pp. 428–431. IEEE, 2009.

Li, Peng, and Teng Wen-Da. 2010. A focused web crawler face stock information of financial field.

Facts about Erdős numbers and the collaboration graph. 2010. Retrieved from http://wwwp.oakland.edu/enp/trivia/.

Erdős Number Project (Oakland University).

Table 1: Running time (seconds) for computing PopuRank for news items on an average day

| time frame | no. news items | running time |
|------------|----------------|--------------|
| 00:00      | 1238           | 13.671       |
| 01:00      | 11             | 0.119        |
| 02:00      | 16             | 0.116        |
| 03:00      | 5              | 0.088        |
| 04:00      | 4              | 0.076        |
| 05:00      | 2              | 0.070        |
| 06:00      | 3              | 0.082        |
| 07:00      | 15             | 0.249        |
| 08:00      | 3              | 0.196        |
| 09:00      | 7              | 0.203        |
| 10:00      | 841            | 6.343        |
| 11:00      | 602            | 4.735        |
| 12:00      | 6              | 0.283        |
| 13:00      | 1007           | 8.848        |
| 14:00      | 2089           | 38.700       |
| 15:00      | 1444           | 13.767       |
| 16:00      | 2100           | 25.918       |
| 17:00      | 2485           | 40.937       |
| 18:00      | 685            | 4.437        |
| 19:00      | 5              | 0.400        |
| 20:00      | 3              | 0.321        |
| 21:00      | 2              | 0.320        |
| 22:00      | 4              | 0.325        |
| 23:00      | 34             | 0.361        |
Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: Bringing order to the web.

Zheng, Xiaolin, Tao Zhou, Zukun Yu, and Deren Chen. 2008. URL Rule based focused crawler. *IEEE International Conference on e-Business Engineering (ICEBE’08)*. pp. 147–154. IEEE, 2008.
Figure 10: Web display of KWB

Figure 11: Web display of KWB with the menu of categories