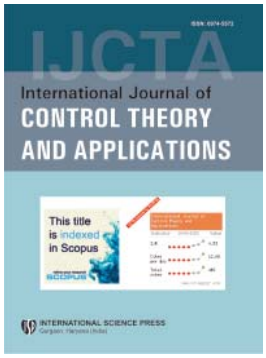

International Journal of Control Theory and Applications

ISSN : 0974–5572

Volume 9 • Number 42 • 2016

A Novel based Approach for Liaison Analysis In Data Summarization and Deep Web Interface Data Extraction

N. Raghavendra Sai[a] and K. Satya Rajesh[b]

[a] *Research Scholar Bharthiar University,Coimbatore*

E-mail: nallagatlaraghavendra@gmail.com

[b]*HOD, ComputerScience SRR and CRR Govt College*

E-mail: ksatyarajesh@gmail.com

Abstract: World Wide Web is developing rapidly; there are large number of Web databases available for users to access. This fast development of the World Wide Web has changed the way in which information is managed and accessed. So the Web can be divided into the Surface Web and the Deep Web. Surface Web refers to the Web pages that are static and linked to other pages, while Deep Web refers to the Web pages created dynamically as the result of specific search. In the same way the Tweets are being created as short text message. Tweets are shared for each users and knowledge analysts. Twitter that receives over four hundred million tweets per day has emerged as a useful supply of reports, blogs, and opinions and additional. In general, tweet summarization and third to observe and monitors the outline - based mostly and volume based variation to supply timeline mechanically from tweet stream. Implementing continuous tweet stream reducing a text document is but not an easy task, since an enormous range of tweets are paltry, unrelated and raucous in nature, because of the social nature of tweeting. However, due to the large volume of web resources and the dynamic nature of deep web, achieving wide coverage and high efficiency is a challenging issue. In this paper, we have a tendency to introduce a unique summarization framework known as summarization (continuous summarization by stream clustering) and also propose a two-stage framework, namely Smart Crawler, for efficient harvesting deep web interfaces. In the first stage, Smart Crawler performs site based searching for center pages with the help of search engines, To achieve ranks websites to prioritize highly relevant ones for a given topic. In the second stage, Smart Crawler achieves fast in-site searching by excavating most relevant links with an adaptive link-ranking crawlers.

Keywords: *Deep Web, Surface Web, Tweet summarization.*

1. INTRODUCTION

Implementing continuous tweet stream summarization is however not an easy task, since a large number of tweets are meaningless, irrelevant and noisy in nature, due to the social nature of tweeting. Further, tweets are strongly correlated with their posted time and new tweets tend to arrive at a very fast rate. Consequently, a good solution for continuous summarization has to address the following three issues: (1) Efficiency—tweet streams are always very large in scale, hence the summarization algorithm should be highly efficient; (2) Flexibility—it should provide tweet summaries of arbitrary time durations. (3) Topic evolution—it should automatically detect sub-topic changes and the moments that they happen.

Unfortunately, existing summarization methods cannot satisfy the above three requirements because: (1) They mainly focus on static and small-sized data sets, and hence are not efficient and scalable for large data sets and data streams. (2) To provide summaries of arbitrary durations, they will have to perform iterative/ recursive summarization for every possible time duration, which is unacceptable. (3) Their summary results are insensitive to time. Thus it is difficult for them to detect topic evolution.

The *deep* (or *hidden*) web refers to the contents lie behind searchable web interfaces that cannot be indexed by searching engines. Based on extrapolations from a study done at University of California, Berkeley, it is estimated that the deep web contains approximately 91,850 terabytes and the surface web is only about 167 terabytes in 2003. More recent studies estimated that 1.9 zettabytes were reached and 0.3 zettabytes were consumed worldwide in 2007. An IDC report estimates that the total of all digital data created, replicated, and consumed will reach 6 zettabytes in 2014. A significant portion of this huge amount of data is estimated to be stored as structured or relational data in web databases — deep web makes up about 96% of all the content on the Internet, which is 500-550 times larger than the surface web. These data contain a vast amount of valuable information and entities such as Infomine, Clusty, BooksInPrint may be interested in building an index of the deep web sources in a given domain (such as book). Because these entities cannot access the proprietary web indices of search engines (*e.g.*, Google and Baidu), there is a need for an efficient crawler that is able to accurately and quickly explore the deep web databases. It is challenging to locate the deep web databases, because they are not registered with any search engines, are usually sparsely distributed, and keep constantly changing.

2. NEED AND IMPORTANCE

A variety of services on the Web such as news filters, text crawling, and topic detecting etc. have posed requirements for text stream clustering. A few algorithms have been proposed to tackle the problem. Most of these techniques adopt partition-based approaches to enable online clustering of stream data. As a consequence, these techniques fail to provide effective analysis on clusters formed over different time durations. While document summarization has been studied for years, micro blog summarization is still in its infancy. Sharifiet al. proposed the Phrase Reinforcement algorithm to summarize tweet posts using a single tweet. Unfortunately, almost all existing document/micro blog summarization methods mainly deal with small and static data sets, and rarely pay attention to efficiency and evolution issues. There have also been studies on summarizing micro blogs for some specific types of events, *e.g.*, sports events. The demand for analyzing massive contents in social Medias fuels the developments in visualization techniques. Timeline is one of these techniques which can make analysis tasks easier and faster. Diakopoulos and Shamma made early efforts in this area, using timelines to explore the 2008 Presidential Debates by Twitter sentiment. Dork et al. presented a timeline-based backchannel for conversations around events.

The emergence of micro blogs has engendered researches on many other mining tasks, including topic modeling, storyline generation and event exploration. Most of these researches focus on static data sets instead of data streams. For twitter stream analysis, Yang et al. studied frequent pattern mining and compression. Van Durme aimed at discourse participants classification and used gender prediction as the example task, which is also a different problem from ours.

To leverage the large volume information buried in deep web, previous work has proposed a number of techniques and tools, including deep web understanding and integration, hidden web crawlers, and deep web samplers. For all these approaches, the ability to crawl deep web is a key challenge. Olston and Najork systematically present that crawling deep web has three steps: locating deep web content sources, selecting relevant sources and extracting underlying content. Following their statement, we discuss the two steps closely related to our work as below. A recent study shows that the harvest rate of deep web is low — only 647,000 distinct web forms were found by sampling 25 million pages from the Google index (about 2.5%). Generic crawlers are mainly developed for characterizing deep web and directory construction of deep web resources that do not limit search on a specific topic, but attempt to fetch all searchable forms.

3. OBJECTIVES

1 We propose a continuous tweet stream summarization framework, namely Sumblr, to generate summaries and timelines in the context of streams.

2. We design a novel data structure called TCV for stream processing, and propose the TCV-Rank algorithm for online and historical summarization.

3. We propose a topic evolution detection algorithm which produces timelines by monitoring three kinds of variations.

4. Extensive experiments on real Twitter data sets demonstrate the efficiency and effectiveness of our framework.

5. We propose a novel two-stage framework to address the problem of searching for hidden-web resources. Our site locating technique employs a *reverse searching* technique (*e.g.*, using Google's"link:" facility to get pages pointing to a given link) and incremental two-level site prioritizing technique for unearthing relevant sites, achieving more data sources.

   a) During the in-site exploring stage, we design a link tree for balanced link prioritizing, eliminating bias toward web pages in popular directories.

   b) We propose an adaptive learning algorithm that performs online feature selection and uses these features to automatically construct link rankers. In the site locating stage, high relevant sites are prioritized and the crawling is focused on a topic using the contents of the root page of sites, achieving more accurate results. During the in site exploring stage, relevant links are prioritized for fast in-site searching

4. METHODOLOGY

The framework consists of 3 main elements, specifically the Tweet Stream bunch module, the High-level summarization module and therefore the Timeline Generation module. The core of the timeline generation module may be a topic evolution detection rule, which consumes online/historical summaries to provide real time/range timelines.

4.1. Tweet Stream bunch

The tweet stream bunch module maintains the net applied mathematics knowledge. Given a subject primarily based tweet stream, it's able to expeditiously cluster the tweets and maintain compact cluster info a climbable bunch framework that by selection stores vital parts of the information. It consists of a web micro-clustering element associate degreed an offline macro - bunch element. A range have exhibit needs for text stream bunch Cluster Stream to come up with length - primarily based bunch results for text and categorical knowledge streams. In distinction, our tweet stream bunch rule is a web procedure while not additional offline bunch. We have a tendency to adapt the net bunch part by incorporating the new structure TCV, and limiting the amount of clusters to ensure potency and therefore the quality of TCVs.

4.1.1. Tweet Stream formatting

At the beginning of the stream, we have a tendency to collect a tiny low variety of tweets and use a *k*-prototype bunch rule to form the initial clusters. Next, the stream bunch method starts to incrementally update the TCVs whenever a replacement tweet arrives.

4.1.2. Incremental bunch

Suppose a tweet t arrives at time *ts*, and there are N active clusters at that point. The key downside is to choose whether or not to draw in into one amongst the current clusters or advance *t* as a replacement cluster. We have a tendency to initial notice the cluster whose center of mass is that the highest to *t*. specifically; we have a tendency to get the center of mass of every cluster, cipher its circular function similarity to *t*, and notice the cluster Cp with the biggest similarity.

4.1.3. Deleting noncurrent Clusters

For most events in tweet streams, timeliness is very important as a result of they typically don't last for a protracted time. To seek out such clusters, associate degree intuitive approach is to estimate the common point of the last P pace of tweets during a cluster. However, storing *p* pace of tweets for each cluster can increase memory prices, particularly once clusters grow massive. Thus, we have a tendency to use associate degree approximate technique to induce Avgp.

4.1.4. Merging Clusters

If the amount of clusters keeps increasing with few deletions, system memories are exhausted. To avoid this, we have a tendency to specify associate degree higher limit for the amount of clusters as N max. Once the limit is reached, a merging method starts. The method merges clusters during a greedy approach. First, we have a tendency to type all cluster pairs by their center of mass similarities during a downward order. Then, beginning with the foremost similar combine, we have a tendency to try and merge 2 clusters in it. Once each of them is united, if they belong to constant composite cluster, this combine is skipped; otherwise, the 2 composite clusters are united along. This method continues till there are solely megahertz share of the first clusters left.

4.2. High - Level summarization

The high-level summarization module provides 2 styles of summaries: on-line and historical summaries. A web outline describes what's presently mentioned among the general public. Thus, the input for generating on-line summaries is retrieved directly from this clusters maintained in memory. On the opposite hand, a historical outline helps folks perceive the most happenings throughout a selected amount, which suggests we'd like to eliminate the influence of tweet contents from the skin of that amount. As a result, retrieval of the specified info for generating historical summaries is a lot of sophisticated, and this shall be our focus within the following discussion. Suppose the length of a user -outlined time length is H, and therefore the ending timestamp of the length is ts.

4.3. Document/Micro blog summarization

Document summarization will be classified as extractive and theoretic. The previous selects sentences from the documents, whereas the latter could generate phrases and sentences that don't seem within the original documents. During this paper, we have a tendency to specialize in extractive summarization. Some works try and extract summaries while not such salient scores. In shapely documents as multi - attribute unsure knowledge downside and optimized a probabilistic coverage of the outline there have additionally been studies on summarizing micro blogs for a few specific styles of events, e.g., sports events. Additionally to on-line summarization, our technique additionally supports historical summarization by maintaining TCV snapshots.

4.4. Timeline Detection

The demand for analyzing huge contents in social media fuels the developments in image techniques. Timeline is one amongst these techniques which might build analysis tasks easier and quicker. Projected the organic process timeline summarization (ETS) to cipher evolution timelines the same as ours that consists of a series

of your time –sealed summaries. The dates of summaries are determined by a pre - outlined timestamp set. Many systems notice vital moments once speedy will increase or"spikes" in standing update volume happen. Developed associate degree rule supported TCP congestion detection, utilized a slope – primarily based technique to seek out spikes. After that, tweets from every moment are known, and word clouds or summaries are chosen. Totally different from this 2 –step approach, our technique detects topic evolution and produces summaries/timelines in a web fashion.

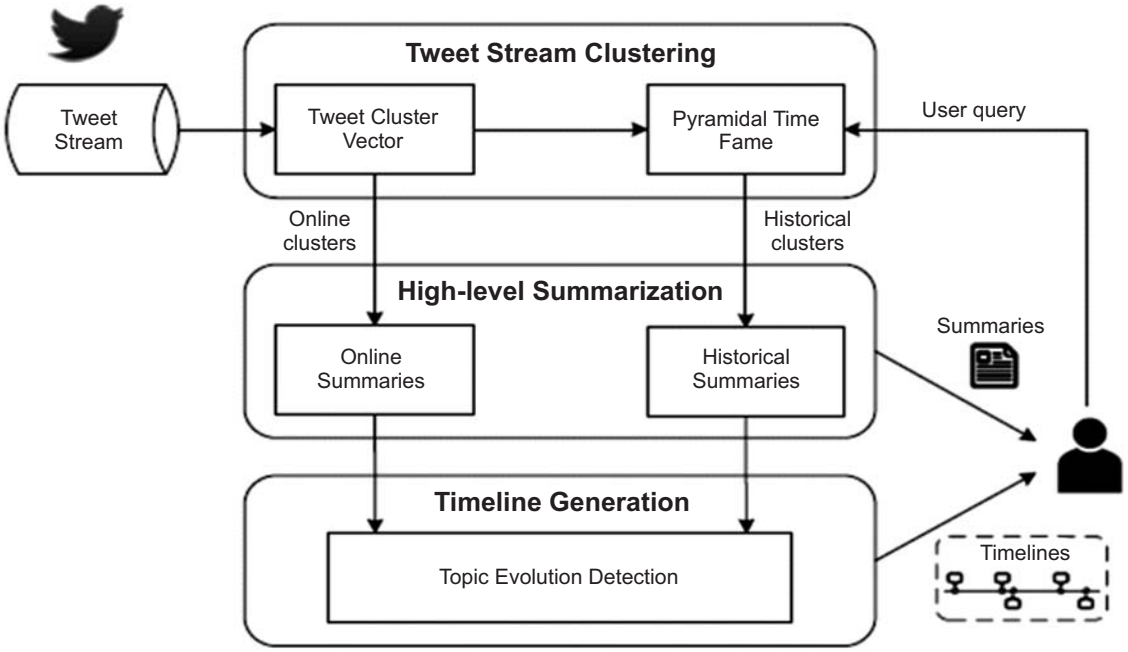

Figure 1: Summarization framework for processing topic evaluation based data extraction

4.4.1. Load Dataset

In this module, we tend to load the Twitter information sets. As a result of tweets are being created and shared at an unexampled rate. Tweets, in their raw type, whereas being informative also can be overwhelming. In this project, wetend to propose a unique continuous account framework known as summarization to alleviate the matter. Thus we tend to load the dataset for continuous account and timeline generation.

4.4.2. Tweet Stream cluster

In this module maintains the net applied math information. Given a subject - primarily based tweet stream, it's ready to with efficiency cluster the tweets and maintain compact cluster info an ascendable cluster framework that by selection stores vital parts of the info, and compresses or discards alternative parts. It consists of four phases like 1)Tweet Stream data formatting 2) Progressive cluster 3) Deleting out-of-date Clusters 4) Merging Clusters

Tweet Stream Initialization : At the beginning of the stream, we tend to collect a little variety of tweets and use a kprototypecluster rule to form the initial clusters. Next, the stream cluster method starts to incrementally update the TCVs whenever a brand new tweet arrives.

1. **Incremental Clustering :** Suppose a tweet t arrives at time ts, and there are N active clusters at that point. The key downside is to make your mind up whether or not to draw in into one amongst the ongoing clusters. That is the highest to t. Particular; we tend to get the centre of mass of every cluster, cipher its cos similarity to t, and realize the cluster Cp with the most important similarity.

2. **Deleting out-of-date Clusters:** for many events in tweet streams, timeliness is vital as a result of they typically don't last for a protracted time. To search out such clusters, AN intuitive method is to estimate the common point in time of the last *p*.

3. **Merging Clusters:** If the amount of clusters keeps increasing with few deletions, system memory is going to be exhausted. To avoid this, we tend to specify a higher limit for the amount of clusters as N max. Once the limit is reached, a merging method starts. The method merges clusters in an exceedingly greedy method. First, we tend to type all cluster pairs by their centre of mass similarities in an exceedingly dropping order. This method continues till there are solely megahertz share of the initial clusters left. The high-level account module provides 2 sorts of summaries: on-line and historical summaries. An internet outline describes what's presently mentioned among the general public. Thus, the input for creating on-line summaries is retrieved directly from the present clusters maintained in memory. On the opposite hand, a historical outline helps Folks perceive the most happenings throughout a particular amount, which implies we want to eliminate the influence of tweet contents from the skin of that amount. As a result, retrieval of the specified info for generating historical summaries is a lot of sophisticated, and this shall be our focus within the following discussion. Suppose the length of a user - outlined time period is H, and therefore the ending timestamp of the period is ts.

4. **Timeline Detection:** The demand for analyzing huge contents in social media the developments in visible techniques. Timeline is one amongst these techniques which may build analysis tasks easier and quicker. It given a timeline primarily based backchannel for conversations around events. It projected the organic process timeline account (ETS) to cipher evolution timelines just like ours that consists of a series of your time - sealed summaries. The dates of summaries are determined by a pre - outlined timestamp set. In distinction, our methodology discovers the ever-changing dates and generates timelines dynamically throughout the method of continuous account. Moreover, ETS doesn't specialize in potency and quantify ability problems, which are important in our streaming context.

5. ALGORITHMS

K-Prototype Clustering:

1. Select *k* initial prototypes from a data set X, one for each cluster.
2. Allocate each object in X to a cluster whose prototype is the nearest. Update the prototype of the cluster after each allocation.
3. After all objects have been allocated to a cluster, retest the similarity of objects against the current prototypes. If an object is found such that its nearest prototype belongs to another cluster rather than its current one, reallocate the object to that cluster and update the prototypes of both clusters.
4. Repeat (3) until no object has changed clusters after a full cycle test of X.

Algorithm 1. Incremental Tweet Stream Clustering

Input: A cluster set C set

1. While! stream.end () do
2. Tweet *t* = stream. Next ();
3. Choose Cp in C set whose centred is the Closest to *t*;
4. If MaxSim (*t*) < MBS then

5. Create a new cluster C new = ftg;
6. C set.add (Cnew);
7. Else
8. Update Cp with *t*;
9. If TS current % (*ai*) == 0 then
10. Store C set into PTF;

Algorithm 2. TCV-Rank Summarization

Input: A cluster set D(*c*)

Output: A summary set S

1. S = 0, T = fall the tweets in ft set of D(*c*)*g*;
2. Build a similarity graph on T;
3. Compute LexRank score LR;
4. T*c* = ftweets with the highest LR in each cluster *g*;
5. While *j*S*j* < L do
6. For each tweet ti inTcS do
7. Calculate *vi* according to Equation (2);
8. Select tmax with the highest *vi* ;
9. S.add (tmax);
10. While *j*S*j* < L do
11. For each tweett0i in T *c* – S do
12. Calculate *v*' *i* according to Equation (2);
13. Select tmax with the highest *v*' *i* ;
14. S.add (*t*'max);
15. Return S;

Algorithm 3. Topic Evolution Detection

Input: A tweet stream binned by time units

Output: A timeline node set TN

1. TN = 0;
2. While! stream.end () do
3. Bin Ci = stream. Next ();
4. If ha sLarge Variation ( ) then
5. TN.Add (*i*);
6. Return TN;

6. SUMMARIZATION FRAMEWORK FOR PROCESSING TOPIC EVALUATION BASED DATA EXTRACTION

An effective deep web harvesting framework, namely Smart Crawler, for achieving both wide coverage and high efficiency for a focused crawler. Based on the observation that deep websites usually contain a few searchable forms and most of them are within a depth of three our crawler is divided into two stages: site locating and in-site exploring. The site locating stage helps achieve wide coverage of sites for a focused crawler, and the in-site exploring stage can efficiently perform searches for web forms within a site. Existing strategies were dealing with creation of a single profile per user, but conflict occurs when user‘s interest varies for the same query Eg.When a user is interested in banking exams in query ―bank‖ may be slightly interested in accounts of money bank where not at all interested in blood bank. At such time conflict occurs so we are dealing with negative preferences to obtain the fine grain between the interested results and not interested. Consider following two aspects:

1. **Document-Based method:** These methods aim at capturing users‘ clicking and browsing behaviour. It deals with click through data from the user *i.e*. the documents user has clicked on. Click through data in search engines can be thought of as triplets (q, r, c)

   Where,

   q = Query
   r = Ranking
   c = Set of links clicked by user.

2. **Concept-based methods:** These methods aim at capturing users‘conceptual needs. Users browsed documents and search histories. User profiles are used to represent users‘interests and to infer their intentions for new queries.

Disadvantages of Existing System :

1. Deep-web interfaces.
2. Achieving wide coverage and high efficiency is a challenging issue.

To efficiently and effectively discover deep web data sources, SmartCrawler is designed with two stage architecture, site locating and in-site exploring, as shown in Figure 1. The first site locating stage finds the most relevant site for a given topic, and then the second in-site exploring stage uncovers searchable forms from the site. Specifically, the site locating stage starts with a seed set of sites in a site database. Seeds sites are candidate sites given for Smart Crawler to start crawling, which begins by following URLs from chosen seed sites to explore other pages and other domains. When the number of unvisited URLs in the database is less than a threshold during the crawling process, Smart Crawler performs ‖reverse searching‖ of known deep websites for center pages (highly ranked pages that have many links to other domains) and feeds these pages back to the site database. Site Frontier fetches homepage URLs from the site database, we going to rank the relevant information.

To efficiently and effectively discover deep web data sources, Smart Crawler is designed with two stage architecture, site locating and in-site exploring, as shown in Figure. The first site locating stage finds the most relevant site for a given topic, and then the second in-site exploring stage uncovers searchable forms from the site. Specifically, the site locating stage starts with a seed set of sites in a site database. Seeds sites are candidate sites given for Smart Crawler to start crawling, which begins by following URLs from chosen seed sites to explore other pages and other domains. When the number of unvisited URLs in the database is less than a threshold during the crawling process, Smart Crawler performs ‖reverse searching‖ of known deep websites for center pages (highly ranked pages that have many links to other domains) and feeds these pages back to the site database.

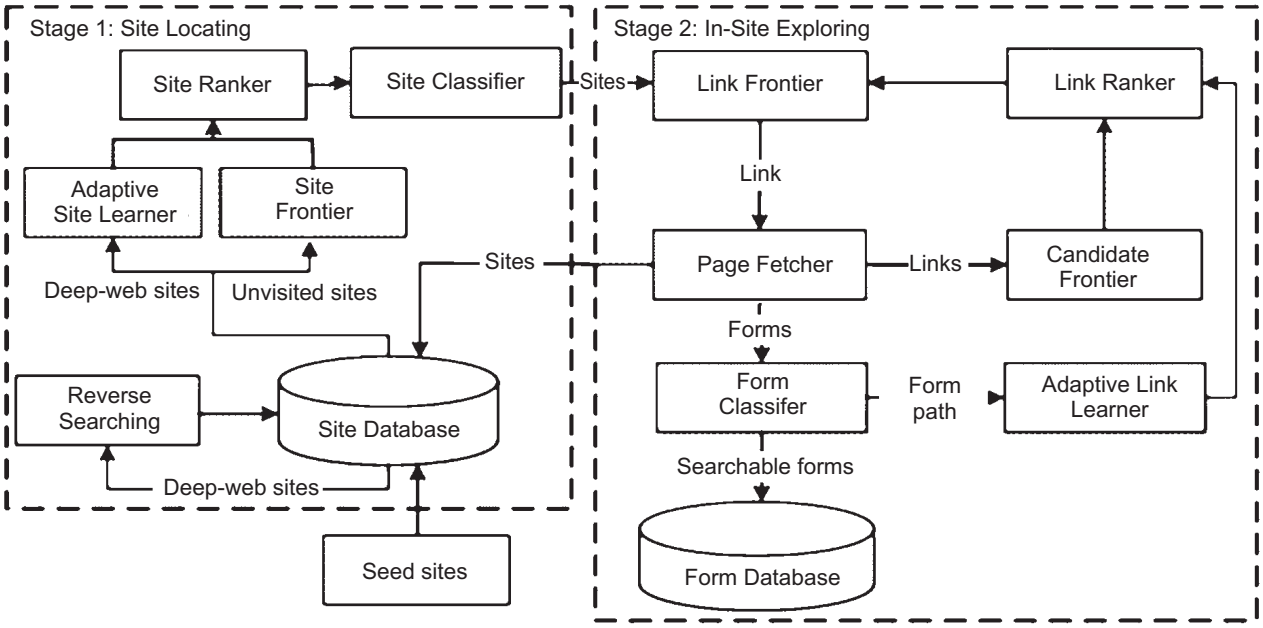

Figure 2: Web interface data extraction frame work based on summarization

6.1. Algorithms and Techniques used

Algorithm 1: Reverse searching for more sites

Input: Seed sites and harvested deep websites

Output: Relevant sites

1. While # of candidate sites less than a threshold do
2. // pick a deep website
3. Site = get Deep WebSite(site Database, seed Sites)
4. Result Page = reverse Search(site)
5. Links = extrac tLinks(result Page)
6. Foreachlink in links do
7. Page = download Page(link)
8. Relevant = classify(page)
9. If relevant then
10. Relevant Sites = extract Unvisited Site(page)
11. Output relevant Sites
12. End
13. End
14. End

Algorithm 2: Incremental Site Prioritizing

Input : SiteFrontier

Output: Searchable forms and out-of-site links

1. HQueue = Site Frontier.Create Queue(High Priority)
2. LQueue = Site Frontier.Create Queue(Low Priority)
3. While site Frontier is not empty do
4. If HQueue is empty then
5. HQueue.add All(LQueue)
6. LQueue.clear()
7. End
8. Site = HQueue.poll()
9. Relevant = classify Site(site)
10. If relevant then
11. Perform In Site Exploring(site)
12. Output forms and Out Of Site Links
13. Site Ranker.rank(Out Of Site Links)
14. If forms is not empty then
15. HQueue.add (Out Of Site Links)
16. End
17. Else
18. LQueue.add(Out Of Site Links)
19. End
20. End
21. End

7. MODULE INFORMATION

7.1. Two-stage crawler

It is challenging to locate the deep web databases, because they are not registered with any search engines, are usually sparsely distributed, and keep constantly changing. To address this problem, previous work has proposed two types of crawlers, generic crawlers and focused crawlers. Generic crawlers fetch all searchable forms and cannot focus on a specific topic. Focused crawlers such as Form-Focused Crawler (FFC) and Adaptive Crawler for Hidden-web Entries (ACHE) can automatically search online databases on a specific topic. FFC is designed with link, page, and form classifiers for focused crawling of web forms, and is extended by ACHE with additional components for form filtering and adaptive link learner. The link classifiers in these crawlers play a pivotal role in achieving higher crawling efficiency than the best-first crawler However, these link classifiers are used to predict the distance to the page containing searchable forms, which is difficult to estimate, especially for the delayed benefit links (links eventually lead to pages with forms). As a result, the crawler can be inefficiently led to pages without targeted forms.

7.2. Site Ranker

When combined with above stop-early policy. We solve this problem by prioritizing highly relevant links with link ranking. However, link ranking may introduce bias for highly relevant links in certain directories. Our solution is to build a link tree for a balanced link prioritizing. Figure 2 illustrates an example of a link tree constructed from the homepage of http://www.abebooks.com. Internal nodes of the tree represent directory paths. In this example, servlet directory is for dynamic request; books directory is for displaying different catalogs of books; Amdocs directory is for showing help information. Generally each directory usually represents one type of files on web servers and it is advantageous to visit links in different directories. For links that only differ in the query string part, we consider themas the same URL. Because links are often distributed unevenly in server directories, prioritizing links by the relevance can potentially bias toward some directories. For instance, the links under books might be assigned a high priority, because ―book‖ is an important feature word in the URL. Together with the fact that most links appear in the books directory, it is quite possible that links in other directories will not be chosen due to low relevance score. As a result, the crawler may miss searchable forms in those directories.

7.3. Adaptive learning

Adaptive learning algorithm that performs online feature selection and uses these features to automatically construct link rankers. In the site locating stage, high relevant sites are prioritized and the crawling is focused on atopic using the contents of the root page of sites, achieving more accurate results. During the in site exploring stage, relevant links are prioritized for fast in-site searching. We have performed an extensive performance evaluation of Smart Crawler over real web data in 1representativedomains and compared with ACHE and site-based crawler. Our evaluation shows that our crawling framework is very effective, achieving substantially higher harvest rates than the state-of-the-art ACHE crawler. The results also show the effectiveness of the reverse searching and adaptive learning.

8. SIZE OF SAMPLES

Dynamically extract data from different web sites with registered domains. We construct five data sets to evaluate summarization. One is obtained by conducting keyword filtering on a large Twitter data set.

9. HYPOTHESIS

As no previous work has conducted similar study on continuous summarization, we have to build our own ground truth (reference summaries). However, manual creation of these summaries is apparently impractical due to the large size of the data sets. Summarization employs a tweet stream summarization formula to compress tweets into Tweet Clustering vector and maintains them in an internet fashion. Our planned k-prototype clump formula made tighter clusters than k- suggests that clump, particularly if the clusters square measure spherical. The subject evolution is detected mechanically, permitting summarization to supply dynamic timelines for tweet streams

An effective harvesting framework for deep-web interfaces, namely *Smart- Crawler*. We have shown that our approach achieves both wide coverage for deep web interfaces and maintains highly efficient crawling. We have studied how to build an effective web crawler. The study carried out based on crawl ordering which reveals that the incremental crawler performs better and is more powerful because it allows re-visitation of pages at different rates.

REFERENCES

[1] T. Zhang, R. Ramakrishna, and M. Livny, "BIRCH: Associate in Nursing economical data clump technique forterribly massive information bases," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 1996, pp. 103-114.

[2] P. S. Bradley, U. M. Fayyad, and C. Reina, "Scaling clump algorithms to massive databases," in Proc. Known.Discovery data processing, 1998, pp. 9-15.

[3] R. Yan, X. Wan, J. Otter batcher, L. Kong, X. Li, and Y. Zhang, "Evolutionary timeline summarization: Abalanced optimization framework via reiterative substitution,"in Proc. 34th Int. ACM SIGIR Conf. Res. Develop.Inf. Retrieval, 2011, pp. 745-754.

[4] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller, "Twit-info: Aggregating andvisualizing micro-blogs for event exploration," in Proc. SIGCHI Conf. Human Factors Comput. Syst., 2011, pp.227236.Conf. Data Mining, 2007, pp. 491-496.

[5] W.-T. Yih, J. Goodman, L. Vander wend, and H. Suzuki, "Multidocument summarization by increasinginformative content words," in Proc. 20th Int. Joint Conf. Artif. Intel. 2007, pp. 1776-178

[6] Peter Lyman and Hal R. Varian. How much information? 2003. Technical report, UC Berkeley, 2003.

[7] Roger E. Bohn and James E. Short. How much information? 2009 report on american consumers. Technical report, University of California,San Diego, 2009.

[8] Martin Hilbert. How much information is there in the‖information society‖? Significance, 2012.

[9] Idc worldwide predictions 2014: Battles for dominance – and survival on the 3rd platform.http://www.idc.com/research/Predictions14/index.jsp, 2014.

[10] Michael K. Bergman. White paper: The deep web: Surfacing hidden value. Journal of electronic publishing, 2001.

[11] Amol Dhepe, Bharat Burghate"Continuous Summarization Framework forEvolutionary Tweet Streams IJARIIE-ISSN(O)-2395-4396 "

[12] Namish A. Diwate1; Kanchan Varpe"Extracting the Web Data Through Deep Web Interfaces".