ISSN: 2229-6956(ONLINE)
DOI: 10.21917/ijsc.2015.0140

AUTOMATIC TAGGING OF PERSIAN WEB PAGES BASED ON N-GRAM LANGUAGE MODELS USING MAPREDUCE

Saeed Shahrivari[1, *], Saeed Rahmani[2, *] and Hooman Keshavarz[3, *]
E-mail: [1]saeed.shahrivari@gmail.com, [2]s.rahmani.ce@gmail.com, [3]h.keshavarz@gmail.com

Abstract

Page tagging is one of the most important facilities for increasing the accuracy of information retrieval in the web. Tags are simple pieces of data that usually consist of one or several words, and briefly describe a page. Tags provide useful information about a page and can be used for boosting the accuracy of searching, document clustering, and result grouping. The most accurate solution to page tagging is using human experts. However, when the number of pages is large, humans cannot be used, and some automatic solutions should be used instead. We propose a solution called PerTag which can automatically tag a set of Persian web pages. PerTag is based on n-gram models and uses the tf-idf method plus some effective Persian language rules to select proper tags for each web page. Since our target is huge sets of web pages, PerTag is built on top of the MapReduce distributed computing framework. We used a set of more than 500 million Persian web pages during our experiments, and extracted tags for each page using a cluster of 40 machines. The experimental results show that PerTag is both fast and accurate.

Keywords:

Automatic Web Page Tagging, Persian Web, MapReduce

1. INTRODUCTION

Finding relevant documents according to the input query is the most important goal of every information retrieval system. A typical retrieval system answers incoming queries using a *full-text* index. Using the full-text index, an information retrieval system can examine all of the documents that match all of the words of the query. Afterwards, the system gives a weight to every matched document and sorts (ranks) the documents using the given weights [1].

The same approach is used by the web information retrieval systems commonly known as web search engines. In addition to full-text index, search engines use other information like graph-based ranks, knowledge graphs, user feedbacks, and user profiles to provide higher quality results [2]. However, when the number of documents becomes large, a full-text index cannot be maintained on a single commodity machine. Current search engines should index billion of web pages, and to provide a fast and usable index, they use a common approach known as sharding [3].

Sharding, which is initially propounded in the world of databases, is the act of partitioning a big thing to smaller independent parts. For example, if a large table of a database is sharded, each individual shard holds a portion of table rows on a separate database instance. For sharding a full-text index, the set of documents is divided between the available machines, and upon the arrival of a query, the query is given to all of the shards and the final result is made by aggregating results from all of the shards [3].

*Freelancer, Iran

Although sharding resolves the challenge of dealing with large collections of documents, it is not useful enough when a limited number of machines are available. When a large collection is sharded between a small numbers of machines, a large portion will be assigned to each machine. As the result, each machine cannot respond incoming queries in reasonable times. An effective solution to this problem is reducing the indexed information of each document. A common approach is using text summarization methods [4].

Summarization methods are effective, but there are some problems when applied to non-English languages like Persian. An effective summarization method should incorporate complex Natural Language Processing (NLP) methods, and the available methods for Persian are not as effective as English [5]. On the other hand, summarization does not reduce the size of the text of a web page significantly, and if we just want to use several machines, summarization is not effective. Therefore, using summarization is not feasible when the target is a large collection of Persian web pages and there are just several machines for indexing the pages.

We propose PerTag which is a solution to make indexing of a huge Persian web page collection on a small set of machines possible. PerTag uses n-gram models, and a simple tf-idf based method plus simple language rules to extract high quality tags per each web page. Roughly speaking, PerTag receives a collection of Persian web pages as the input, processes the collection, and produces a set of tags per each page as the output. After PerTag is applied on the web page collection, tags can be indexed instead of the text of each page and therefore, the size of index can be significantly decreased. Producing a smaller index makes it possible to handle queries in feasible time using a limited set of machines.

Since handling and processing large volumes of data is not possible on a single machine, we used the MapReduce framework for implementing PerTag [6]. We tested PerTag on a collection of Persian web pages with more than 500 million pages, and the results show that PerTag can generate high quality tags for each webpage both fast and efficiently.

2. RELATED WORK

The related work can be categorized into three major groups: key phrase extraction and automatic tagging, page summarization, web page clustering.

The target of key phrase extraction is to extract the most important phrases from a piece of text. The most notable work is KEA [7]. KEA identifies candidate keyphrases using lexical methods, calculates feature values for each candidate, and then uses a machine-learning algorithm to predict which candidates are good keyphrases. Some other works use domain specific

methods to generate more proper phrases. A good example is the work by Eibe et al [8]. An effective approach for increasing the precision of the method is using large document collections [9]. A good example that combines domain specific methods and large collections is proposed by Zhao et al in which they have applied topical model on Twitter's texts [10]. However, a key difference of key phrase extraction with automatic tagging is the number of extracted phrases.

In some specific fields, good tagging solutions have been proposed. For example, Liu et al have proposed an automatic tagging solution for web news [11]. Good et al have also proposed a method for extracting semantics from text information [12]. A tagging solution is proposed by Morrison based on folksonomies but his solution is based on manual tagging [13]. Handschuh et al have proposed S-CREAM which is a semi-automatic approach [14].

Page summarization is another related concept. TextRank and LexRank are the two most famous algorithms in the area of text summarization [15], [16]. TextRank uses the concept of the well-known algorithm PageRank, to make a graph of words in each document, and then summarizes each document using the word graph. LexRank uses a concept called centroid sentences to extract important sentences of each document. LexRankis part of a larger summarization system called MEAD that combines the LexRank score with other features like sentence position and length. An important distinction is that TextRank was used for single document summarization, while LexRank has been applied to multi-document summarization. A good survey on text summarization can be found in [17].

Document clustering has a deep relation with tagging, too. Document clustering uses cluster analysis methods to group similar documents in the same clusters. The available methods can be classified into two groups. The first group is the hierarchical algorithms, which includes methods like single linkage, and complete linkage methods. Hierarchical methods form a hierarchy of documents. The other group is partitional methods which mostly use the k-means algorithm and its variants. Most of the clustering algorithms extract tags per each document and use the tags of each document to compute similarities between each pair of documents [18], [19]. Further information on document clustering and tagging can be found in [20].

Some related works also focus on large scale text analysis. Grimmer et al have proposed a method for large scale content analysis [21]. Gang et al have proposed a clustering algorithm for large scale click stream analysis [22]. A good survey is done by Tsytsarau and Palpanas [23].

3. BACKGROUND

We split this section into two parts. First, we discuss the n-gram models, and then we describe the MapReduce model.

3.1 N-GRAM MODELS

Language models are one of the most important parts of natural language processing that has been used in many fields such as speech recognition, statistical machine translation and information retrieval systems.

A Language Model estimates the probability of every sequence of words. The quality of estimation depends on the available training data. Therefore, using large amount of training data is required to build powerful language models.

The most common language model is the n-gram model. N-gram language models use statistics of followed words in the training data, and compute the probability of a string $W = w_1, w_2, \ldots, w_n$ considering the sequence of previous words. Formally speaking, it can be expressed using a chain rule:

$$P(W) = P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_2|w_1)\ldots P(w_n|w_1, \ldots, w_{n-1})$$

Using the chain rule, the sentence probability decomposed into single-word probability. To estimate the probability of each term in the language model, we need to collect statistic of the n sequence word from large amount of text data. So, the large amount of sequence words may be never occurred in the training data. Thus, the estimated probability will be zero.

Most commonly, researchers use trigram language model. That is to say, the two previous words are considered to predict the third word. To achieve this purpose, statistics of three sequence words should be collected, aka trigram model. N-gram language model may also be estimated for two sequence of words, aka bigram model, single, words i.e. unigram model, or any other count of words.

The probability of trigram language models can be computed by: $P(W) = P(w_1)\ P(w_2|w_1)\ \ldots P(w_n|w_{n-2},\ w_{n-1})$. The simplest way to prediction probabilities $P(w_3|w_1,w_2)$ is straightforward. According to maximum likelihood estimation, we have:

$$P(w_3|w_1w_2) = \frac{count(w_1w_2w_3)}{count(w_1w_2)}$$

3.2 THE MAPREDUCE FRAMEWORK

MapReduce is a framework that allows declaring programs that can be easily run over hundreds and even thousands of machines [6]. MapReduce delivers three key features in a package: easy programming logic, fault tolerant execution, and near linear scalability [24]. Every MapReduce program compromises a Map step, and a Reduce step. The map step takes a set of key/value pairs and converts it into another set of key/value pairs. The reduce step takes the output of the map step and combines all the values that are associated to an identical key and then delivers the combined values to a reduce function. Hadoop is the most widely used open-source implementation of the MapReduce programming model. Hadoop is written in Java and it is capable of running across thousands of machines [25]. Further information about MapReduce and Hadoop can be found in [6], [24], [25].

4. THE PROPOSED SOLUTION PERTAG

PerTag consists of three phases:

1. Anchor text extraction
2. N-gram extraction
3. N-gram selection

During the anchor extraction phase, PerTag uses the link structure between pages to extract inbound anchor text for each page. Most of the links in a web page contain some texts that

describe the destination linked page. The attached text of a link is called anchor text. Since anchor texts are assigned to a page by other pages, they are usually more important than the actual text of a web page. In the first phase, PerTag extracts anchor texts for each page by inverting the link structure.

The second phase is called N-gram extraction, in which all of the n-grams of every page are extracted, and we build an inverted index in which n-grams are the keys, and the list of pages that contain the n-gram is the value.

In the last phase which is n-gram selection, PerTag selects tag per each page using a composite approach that is based ontf-idf and Persian language rules. The main task is dealing with very frequent and very infrequent n-grams. Very frequent n-grams occur in many pages and hence, the set of pages that get the tag should be pruned. On the other hand, very rare tags are mostly typos and encoding errors and hence can be discarded safely. Also, we apply simple Persian language rules to remove n-grams that do not hold meaningful information. Next, we discuss each phase one by one.

4.1 ANCHOR TEXT EXTRACTION

Before we explain the anchor text extraction procedure, it is necessary to define the structure of a page. We assume that a page has five fields: URL, title, links (the outgoing links), anchors (inbound anchor texts), and the body (actual text of the page). A page must have URL but other fields can be empty, and the anchors field is initially empty for all of the pages. A schematic is given in Fig.1.

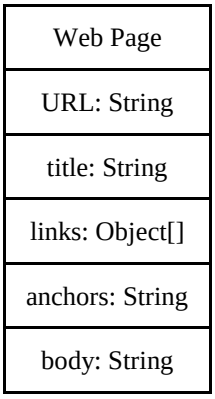

Fig.1. Structure of a web page

Since PerTag is built on top of MapReduce, it should be declared as a series of map and reduce functions. For filling anchors of each page, we should process the reverse link structure between documents. Therefore, in the map function we emit the reverse links plus anchor texts, and in the reduce function we aggregate all of the tags associated to each page. The formal declaration of anchor extraction step is given in Algorithm 1.

After execution of Algorithm 1, the anchors of each URL will be extracted, and according to them, the anchor field of each web page can be set.

4.2 N-GRAM EXTRACTION

After extraction of anchor texts, we can extract n-grams using text fields of the pages. Extracting n-grams is very simple. We tokenize the text fields and emit n-grams by sliding a window over the tokens sequence. In this paper, we just considered n-grams of size 1, 2, and 3, aka onegrams, bigrams, and trigrams.

Algorithm 1: Extracting anchor texts

Let *d* be a web page with structure given in Fig.1.
Function *map*(*d*)
foreach*link*in*links*do:
hlink←hyper_link(*link*)
text←anchor_text(*link*)
emit(*hlink,text*)
endfor

Function *reduce*(*link, anchors*)
all_anchors=''
foreach*anchor*in*anchors*do:
all_anchors=all_anchors+anchor
endfor
emit(*link,all_anchors*)

Algorithm 2: Extracting n-grams

Let *d* be a web page with structure given in Fig.1.
Function *map*(*d*)
let*dic*be a dictionary.
foreach*text_field*of*d*do:
let*NG* be the set of onegrams, bigrams, and trigrams of *text_field.*
foreach*gram* in*NG* do:
add (*gram,count(gram)*)to*dic*
endfor
endfor
foreach*entry* in*dic*do:
emit(*entry.key*,{d.URL, *entry.value*})
endfor

Function *reduce*(*gram, values*)
list={}
foreach*val*in*values*do:
add*val*to *list.*
endfor
sort *list* by count
emit(*gram*,*list*)

For declaring n-gram extraction in MapReduce, again, we should define map and reduce functions. The idea is to generate n-grams in the map function, and aggregate the URLs and the n-gram's frequency in the reduce function. The formal definition of n-gram extraction is given in Algorithm 2.

Algorithm 2 scans the text fields of each web page and aggregates the URLs and the number of occurrence of each n-

gram in the reduce step. After the execution of Algorithm 2, we have an inverted table in which keys are the n-grams and the value is the list of URLs plus the frequencies of the n-gram, and the list is sorted by the frequencies descendingly. The n-grams are the candidates for tags. However every n-gram is not suitable and we should select proper tags from candidate n-grams.

4.3 N-GRAM SELECTION

For tag selection, we use a composite approach which is based on the tf-idf model, and also some simple Persian language rules. tf–idf is the short form of term frequency–inverse document frequency, which is a numerical statistic that reflects how important a word is to a document in a collection. tf is the frequency of a term in a document. Here, the terms are the n-grams and the documents are the web pages. In the map function of Algorithm 2, we computed the frequency of each n-gram in each page. tf shows how important is an n-gram in a page.

Idf is a measure that shows how much information a term provides. Idf of a term is formally defined as the total number of documents divided by the number of documents that contain the term. The idf for each n-gram can be easily computed using the length of the list of pages that is emitted in the reduce function of Algorithm 2.

For selecting n-grams based on tf-idf, we consider two extreme cases: 1) when an n-gram is very rare, 2) when an n-gram is very frequent and has occurred in many documents. For the case of very rare n-grams, we discard them. PerTag discarded all of the n-grams that occur in less than 50 pages. For the case of very frequent n-grams, we cut the list of documents that contain the n-gram. PerTag cuts the list of pages for the n-grams that occur in more than 100,000 pages and just keeps the 100,000 pages with the highest n-gram frequency.

Most of the rare n-grams are whether misspelled, or are from other languages aside Persian. On the other hand, most of the frequent n-grams were stop words or other regular n-grams like the Persian bigram‘نرم افزار’ which means ‘software’ in English. Some of the most frequent n-grams that PerTag has cut the list of pages are given in Table.1.

Besides tf-idf based pruning, many meaningless n-grams can also be removed using simple grammatical rules. PerTag uses three simple rules: conjunctions & prepositions, numbers, andplurals. Next, we discuss them one by one:

Conjunctions & prepositions: we remove an n-gram when one or both of the outer words are conjunctions or prepositions. For example, the bigram ‘to go’ can be safely discarded because the onegram ‘go’ is enough to express it. However, for the case of trigrams, if the conjunction or preposition is in the middle of the trigram, the trigram should not be pruned. The list of the 10 most frequent conjunctions and prepositions that PerTag uses is given in Table.2.

Numbers: PerTag discards an n-gram which its size is greater than one, and all of its words are numbers. Although this rule is simple, it prunes many n-grams that do not have a piece of information.

Plurals: Persian has many plural postfixes. The most widely used plural postfixes are: ‘ها’, ‘ان’, ‘ات’, ‘ون’, ‘های’. PerTag converts all of the plural forms to the ‘ها’ postfix.

Table.1. List of n-grams that their list of pages is cut

Persian n-gram	Size	English equivalent
ایران	1	Iran
سایت	1	site
دانلود	1	download
جدید	1	new
انجمن	1	forum
صفحه اصلی	2	home page
ثبت نام	2	registration
درباره ما	2	about us
نرم افزار	2	software
وب سایت	2	web site
تماس با ما	3	contact us
پرسش و پاسخ	3	question and answer
جمهوری اسلامی ایران	3	Islamic republic of Iran
اس ام اس	3	SMS
دانلود نرم افزار	3	software download

Table.2. List of n-grams that their pages list is cut

Persian preposition	English equivalent
و	and
به	to
با	with
در	in
از	from
را	particle suffixed to a noun or pronoun
برای	for
که	that
تا	to
بر	on

Using the above 3 rules, PerTag prunes many n-grams. Some sample n-grams that were pruned using the language rules are given in Table.3. N-gram selection can be done in a separate MapReduce job, but it is more efficient to perform n-gram selection in the reduce step of Algorithm 2. The procedure is straight forward. Before emitting an n-gram in line 11, we should just add few if statements and check that the n-gram should be pruned or not.

Table.3. Some n-grams that PerTag pruned using language rules

Persian n-grams	***cause of removal***
های خود را	Prepositions
با ما درباره	Prepositions
ارتباط با	Prepositions
و در	Conjunction
3 2 1	Numbers
4 3	Numbers
سایت های	Plurals
درختان	Plurals

5. EXPERIMENTAL RESULTS

For evaluation of the accuracy and speed of PerTag, we performed several experiments on a collection of 500 million Persian web pages. The collection was compressed using the Snappy compression algorithm and its size was more than 3 Terabyte. We performed the experiments on a Hadoop cluster with 40 dual-core Xeon machines. Hadoop 1.2 and Ubuntu Linux 14.04 were installed on all machines.

We discuss the experiments from two aspects: the quality of tags, and the speed and scalability of execution. Measuring the quality of a web scale system is not straight forward because there is no public dataset for this problem. Therefore, we selected 100 random pages from the set and evaluated the quality of tags using three human experts. We measured two aspects of tags: completely correct tags, and completely wrong tags. We define the tags that exactly describe the page correctly as the correct tags, and the tags that describe the page with a complete false meaning as the wrong tags. There will be some other tags that are not totally correct, but on the other hand they are not totally wrong. This kind of tags is denoted as neutral. The results of 500 random pages are given in Fig.2.

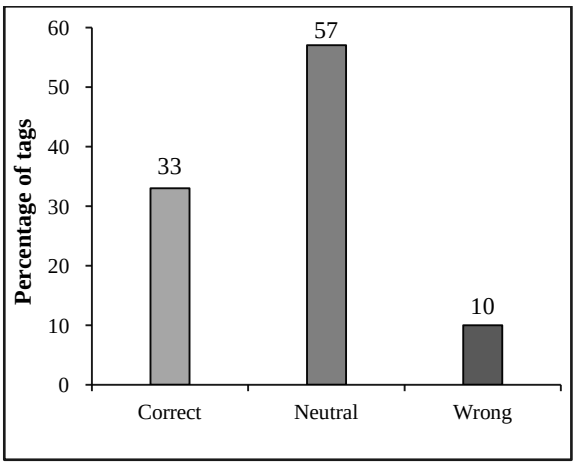

Fig.2. Type of extracted tags

As the numbers in Fig.2 show, most of the extracted tags are neutral. Although percentage of correct tags is less than neutral tags, it is about six times greater than the percentage of wrong tags. This shows that PerTag reaches to a promising quality. Note that most of the wrong tags were caused by advertisements shown in the borders of web pages.

In another experiment, we evaluated the scalability of PerTag. For this purpose, we executed PerTag using different numbers of web pages from 100 million pages to 500 million pages. The normalized execution times are plotted in Fig.3. Note that processing of 100 million pages take about 1.6 hour.

As the values of Fig.3 show, PerTag scales almost linearly when number of pages increases. This shows that PerTag is a scalable solution, too.

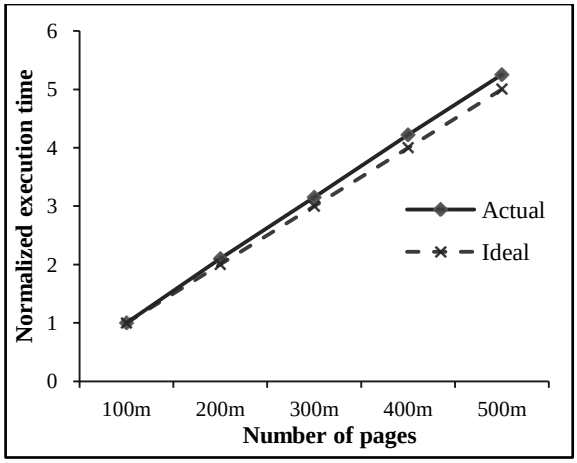

Fig.3. The execution time when the number of pages increases

6. CONCLUSION AND FURTHER WORK

Tagging of web pages is a useful solution for increasing the accuracy and speed of web-scale information retrieval systems. However, most of the available tagging solutions are optimized for English web pages. We proposed PerTag which is a scalable and automatic tagging solution for large-scale Persian web pages. PerTag uses a combination of tf-idf concepts plus simple Persian language rules for extracting meaningful tags for Persian web pages. Since PerTag should handle very large volumes of data, it is designed and implemented using MapReduce and runs over Hadoop. We tested PerTag using a collection of 500 million Persian web pages and the experimental results show that PerTag provide a solution that is both accurate and scalable.

A promising challenge for further research is to increase the accuracy of PerTag. Detecting the main content of a web page should be an effective solution for removing advertisements and hence, reducing the wrong tags. Another possible approach is using additional Persian NLP facilities like lexical databases, stemmers, and Persian parsers. Another direction is increasing the speed of execution using newer in-memory computing engines like Spark [26] instead of the lazy and batch Hadoop framework.

REFERENCES

[1] Ricardo Baeza-Yates and Berthier Ribeiro-Neto, "*Modern information retrieval*", New York: Addison-Wesley, 1999.

[2] Lawrence Page and Sergey Brin and Rajeev Motwani and Terry Winograd, "The PageRank citation ranking: Bringing order to the web", Technical Report, Stanford InfoLab, 1999.

[3] Sergey Melink, Sriram Raghavan, Beverly Yang and Hector Garcia-Molina, "Building a distributed full-text index for the web", *ACM Transactions on Information Systems*, Vol. 19, No. 3, pp. 217–241, 2001.

[4] Triveedee Sandhya and Sahista Machchhar, "Unstructured Text Summarization Approach in the Age of Big Data: A Review", *Data Mining and Knowledge Engineering*, Vol. 7, No. 1, pp. 18–21, 2015.

[5] Martin Hassel and Nima Mazdak, "FarsiSum: a Persian text summarizer", *Proceedings of the Workshop on*

Computational Approaches to Arabic Script-based Languages, pp. 82–84, 2004.

[6] Jeffrey Dean and Sanjay Ghemawat, "MapReduce : Simplified Data Processing on Large Clusters", *Communications of the ACM*, Vol. 51, No. 1, pp. 1–13, 2008.

[7] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin† and Craig G. Nevill-Manning, "KEA: Practical automatic keyphrase extraction", *Proceedings of the fourth ACM conference on Digital libraries*, pp. 254–255, 1999.

[8] Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin and Craig G. Nevill-Manning, "Domain-specific keyphrase extraction", *Proceedings of Sixteenth International Joint Conference on Artificial Intelligence*, pp. 668–673, 1999.

[9] Peter D. Turney, "Coherent keyphrase extraction via web mining", *Proceedings of the 18th international joint conference on Artificial intelligence*, pp. 434–439, 2003.

[10] Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim and Xiaoming Li, "Topical keyphrase extraction from twitter", *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Vol. 1, pp. 379–388, 2011.

[11] LIU Wei and YAN Hua-liang, "Unified and Automatic Web News Object Extraction Approach", *Computer Engineering*, Vol. 38, No. 11, pp. 167–169, 2012.

[12] Mor Naaman, Tye Rattenbury and Nathaniel Good, "Automatic extraction of semantics from text information", US Patent No. US7899804 B2, 2011.

[13] P. Jason Morrison, "Tagging and searching: Search retrieval effectiveness of folksonomies on the World Wide Web", *Information Processing & Management*, Vol. 44, No. 4, pp. 1562–1579, 2008.

[14] Siegfried Handschuh, Steffen Staab and Fabio Ciravegna, "S-CREAM—semi-automatic creation of metadata", *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, Vol. 2473, pp 358–372 2002.

[15] Günes Erkan and Dragomir R. Radev, "LexRank: Graph-based lexical centrality as salience in text summarization", *Journal of Artificial Intelligence Research*, Vol. 22, No. 1, pp. 457–479, 2004.

[16] Rada Mihalcea and Paul Tarau, "TextRank: Bringing order into texts", *Proceedings of EMNLP*, pp. 404–411, 2004.

[17] Charu C. Aggarwal and Chengxiang Zhai, "*Mining text data*", Springer, 2012.

[18] Michael Steinbach, George Karypis and Vipin Kumar, "A comparison of document clustering techniques", *KDD workshop on text mining*, Vol. 400, No. 1, pp. 525–526. 2000.

[19] Alexander Strehl, Joydeep Ghosh and Raymond Mooney, "Impact of similarity measures on web-page clustering", *Workshop on Artificial Intelligence for Web Search*, pp. 58–64, 2000.

[20] Claudio Carpineto, Stanislaw Osiński, Giovanni Romano and Dawid Weiss, "A survey of web clustering engines", *ACM Computing Surveys*, Vol. 41, No. 3, 2009.

[21] Justin Grimmer and Brandon M. Stewart, "Text as data: The promise and pitfalls of automatic content analysis methods for political texts", *Political Analysis*, Vol. 21, No. 3, 2013.

[22] Gang Kou and Chunwei Lou, "Multiple factor hierarchical clustering algorithm for large scale web page and search engine clickstream data", *Annals of Operations Research*, Vol. 197, No. 1, pp. 123–134, 2012.

[23] Mikalai Tsytsarau and Themis Palpanas, "Survey on mining subjective data on the web", *Data Mining and Knowledge Discovery*, Vol. 24, No. 3, pp. 478–514, 2012.

[24] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: a flexible data processing tool", *Communications of the ACM*, Vol. 53, No. 1, pp. 72–77, 2010.

[25] Tom White, "*Hadoop: The definitive guide*", O'Reilly Media/Yahoo Press, 2012.

[26] Saeed Shahrivari, "Beyond Batch Processing: Towards Real-Time and Streaming Big Data", *Computers*, Vol. 3, pp. 117–129, 2014.