International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 16 (2017) pp. 5472-5480

A Development of Automatic Topic Analysis System using Hybrid Feature Extraction based on Spark SQL

Kiejin Park[1] and Minkoo Kang[2]

[1]Professor, Department of Integrative Systems Engineering, Ajou University, Suwon, South Korea.
[2]Researcher, Engineering Research Institute of Ajou University, Suwon, South Korea.

[1]Orcid: 0000-0002-6872-2806 & Scopus Author ID: 55654521300

Abstract

As text data have characteristics such as informal and unstructured nature, in common, we have difficulties in analyzing the data based on the technology of relational data model only. Especially, for the dynamically generated large scale online text data, a real-time analysis of the reaction of a SNS user is hard to accomplish. To deal with the problem, we design and implement topic analysis systems based on LDA (Latent Dirichlet Allocation) model which has no model training process in learning phase while capturing the topic (e.g.: theme) of the informal text data. To explain and summarize the contents of informal text data more precisely, at the preprocessing step for input data set, a feature vector extracted from text string is integrated with categorical vector. We adopt this hybrid vector as the input for LDA model for driving topics of the input texts. For all documents, the topic analysis job is performed automatically.

The proposed system is implemented on top of Hadoop-Spark platform. During LDA calculation, as hard disk access is minimized and intermediate results of the LDA model are located at main memories, high speed in-memory processing performance is achieved. In the experiments, around 160 million number of informal text data (e.g.: SNS comments) are processed in topic analysis. Because added categorical feature results in higher weight values in the topic distribution, we can understand the theme of a document more clearly using the extracted topics which show higher topic weights.

Keywords: Topic Model, LDA (Latent Dirichlet Allocation), Spark SQL, Hadoop

INTRODUCTION

Since unstructured text data that is currently handled online has characteristics of a large capacity and nonstructural form, there is a limit to analytical work only with storing and processing method of the relational data model. In particular, it is difficult to find the meaning (semantics) of user's reactions from the results that are analyzed by using online text data such as comments on SNS (Social Network Service). Recently, in order to solve this problem, LDA (Latent Dirichlet Allocation) topic modeling method that can extract topic inherent in a specific document set without prior training process is getting the spotlight in the field of machine learning [1]. The LDA model is a method of finding a characteristic inherent in a document by using a ratio of topic instead of using a conventional method of analyzing a document as a type or frequency of a specific word. Especially, LDA is effective for analyzing large volumes of data quickly because it is a kind of unsupervised machine learning model which can structuralize unstructured text data automatically.

However, in the case of comments on SNS or message data on Internet which has merely technical terms in a specific field, the result of LDA model can has an ambiguous topic pattern because the result of the distribution of words has only for everyday terms. To this end, in order to abstract and explain the details of unstructured text data, we extracted strings from the input data set of the LDA model and applied the hybrid feature extraction method including category-specific information. In other words, at the initial preprocessing stage of the input data set, a feature vector extracted from the text strings and a feature vector partitioned into categories are combined for the input value of topic extraction, then LDA model can be executed automatically.

On the other hand, for rapid large capacity big data analysis, distributed data storing and parallel processing suitable for cluster computing environment and constructing various IT infrastructures are indispensable [2]. To this end, distributed in-memory based Spark [3] platform on Hadoop cluster was adopted in this paper. Spark can minimize disk access at the time of data calculation and process all intermediate output results on memory, so it has very fast data processing performance compare to the existing MapReduce [4] based processing method. In particular, the column-wise distributed data processing scheme of Spark Dataframe (i.e., Spark SQL) was applied for structured and unstructured big data query processing [5,6].

The rest of the paper is organized as follows. Chapter 2 describes the probability-based topic models, and Chapter 3 discusses the design of automatic topic analysis system by the hybrid feature extraction and explain data processing steps.

Chapter 4 describes implementation for verifying analysis system and shows experimental results. Chapter 5 concludes the paper.

RELATED RESEARCH

Probability-based LDA topic model

A topic model based on probability theory is a statistical inference technique designed to find topics hidden in text documents. Typical topic models are LDA and PLSA [7]. LDA is a methodology of finding topics and the ratio of such topics in each document, assuming that one document is composed of multiple topics. In LDA model, it infers the posterior probability according to condition of word generation. That is, in the topic model, "if we know the distribution of topic $p(z)$ and the probability $p(w|z)$ that are generated word according to each topic, we can calculate the probability $p(d)$ of generating a document" under the premise that "the document is expressed in many topics and words are generated by topics." That is, it means that the posterior probability is inferred according to the condition for generating the word on the assumption that the words are not independent from each other. On the other hand, the PLSA model cannot represent the tendency of the topic distribution of the whole document set because each word in any document is associated with only one topic. Moreover, in the PLSA model, there are drawbacks such as over-fitting [8].

In the LDA model, words are generated from a specific topic and whether the document has a topic proportion θ is determined by Dirichlet distribution with parameter value of α. If n documents are given and all documents belong to any of the k topics, the probability of document generation can be calculated by equation (1).

$$p(D|\alpha,\beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum p(z_{dn}|\theta_d) p(w_{dn}|z_{dn},\beta) \right) d\theta_d \quad (1)$$

where $p(\theta_d|\alpha)$ is the K-dimensional Dirichlet distribution, α and β are hidden parameters, θ and z are hidden variables, and the word w is the only observation data. The posterior probability of the hidden variable θ can be calculated easily by placing the topic distribution θ for each document in the multinomial distribution and determining the prior distribution as the Dirichlet distribution.

The parameter estimation process of the LDA model is shown in Figure 1. The expected value of the hidden variable can be calculated by using the current parameter in the E-step. In the M-step, it finds the parameter that maximizes the log likelihood by setting the expected value found at the E-step as the observation values of the hidden variable [9].

Algorithm: LDA Parameter Inference Pseudo Code

for *d* = 1 to D **do**
  E-Step:
  repeat
    update document/topic distribution for *d*
    update topic/word assignments for *d*
  until convergence
  M-Step:
    update topic/word distribution
 end for

Figure 1: LDA topic model parameter inference process

Since LDA method is suitable for the distributed parallel processing environment, in this paper, topic analysis system is designed and implemented in the Spark cluster environment based on Hadoop YARN (Yet Another Resource Negotiator) [10]. On the other hand, on-line learner of the LDA model was developed on the Hadoop-based MapReduce framework [11]. However, the performance of the on-line learner is limited compared with the in-memory based Spark platform because they use disk-based computation.

HYBRID TOPIC ANALYSIS SYSTEM

The architecture of distributed parallel processing system

The proposed topic analysis system that a system for finding topics in various unstructured document sets uses the in-memory based distributed processing platform of Spark. For rapid processing, main memory based computation is more efficient than hard disk drive based computation. Especially when iterative data processing is mainly achieved as machine learning algorithms, it is efficient to continuously use the intermediate output saved in the memory.

The Spark platform operating on the Hadoop cluster is provided with node management functions (e.g., resource allocation, task scheduling, fault tolerance, etc) via YARN as shown in Figure 2. Then it executes tasks assigned in a distributed parallel manner from each slave node via the Spark main program (i.e., driver program) that operates in an interactive way on the master node. In addition, HDFS (Hadoop Distributed File System) [12] allows RM (ResourceManager) of YARN to place in the master node for resource management. Also, it allows NM (NodeManager) and AM (ApplicationManager) to place in the slave node for processing and storing distributed large capacity data.

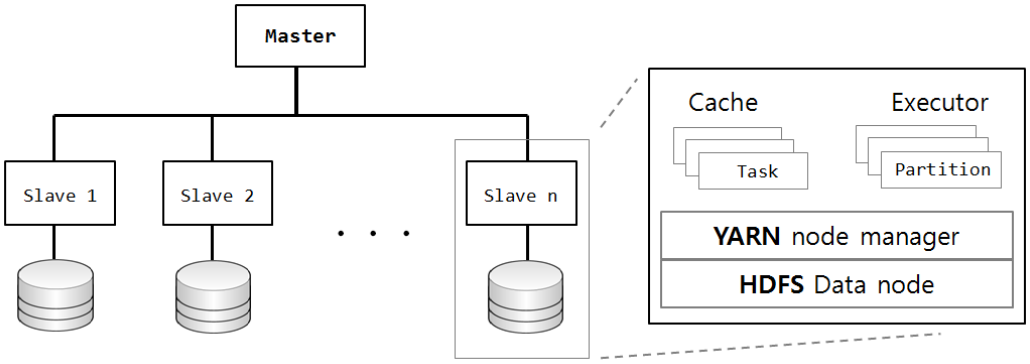


Figure 2: The architecture of Hadoop-YARN based Spark cluster

On the other hand, RDD (Resilient Distributed Datasets) [13] of Spark is a set of read-only data objects processed over multiple distributed memories. Spark adopts DAG (Directed Acyclic Graph) scheduling according to the distributed memory environment compared to MapReduce which is scheduled in Jobs.

Topic analyzing process

In order to analysis interactively, full-text document data is read into a DataFrame that can simultaneously satisfy procedural processing of functional language provided by Spark SQL and declaration processing of SQL. The comments data inputted (input data) go through the hybrid feature extraction process as shown in Figure 3. 1) As the data preprocessing for topic analysis, after dividing the word for each document then deleting the stop word from the word separated for each comment. Next, it is expressed as a value transformed into a vector for each comment. 2) The value derived after the processing for the category is converted into a vector. After executing VectorAssembler with combining the two result values, it is used as features for LDA input values.

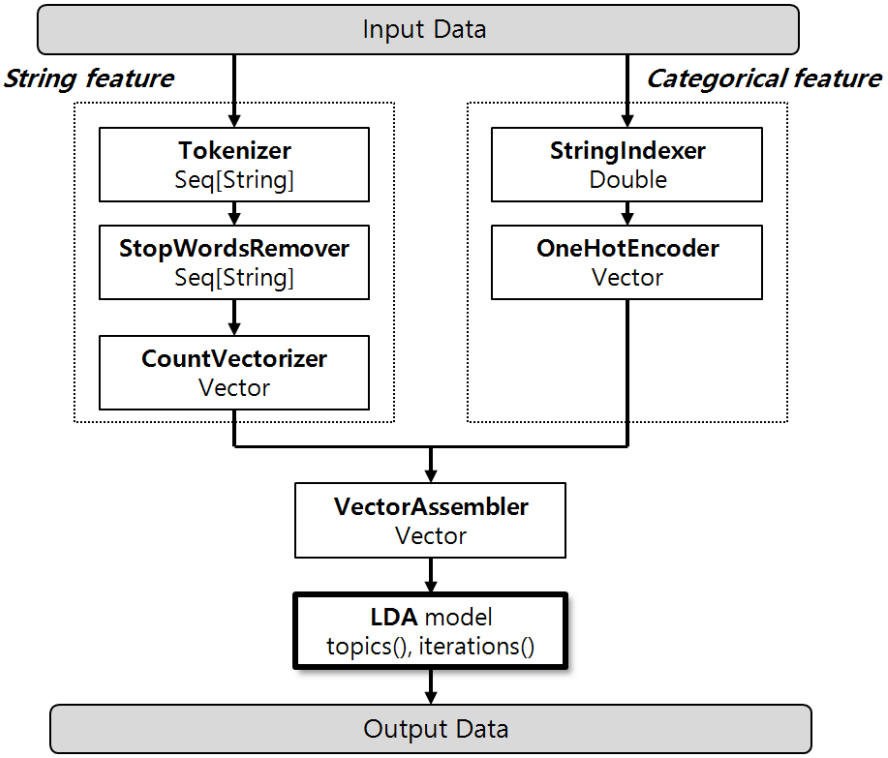


Figure 3: The hybrid feature extraction process

The main source code which is based on Scala language of the feature extraction process is shown in Figure 4. For the comment data after the preprocessing, 1) find the feature 1 which is the value of the first feature extraction by using CountVectorizer library. Also, change the value of string classified as subcategory value from input data to feature 2 by using 2) StringIndexer and OneHotEncoder. By combining feature 1 and feature 2, 3) extract features used as input values of LDA model by using VectorAssembler which changes to one vector value. The proportion of the word used for each topic and the ratio of this topic in each comment can be obtained by executing the LDA model for each comment expressed by a word vector.

Sample Source in Scala

```
    :
    import org.apache.spark.sql.{Row, SparkSession}
    import org.apache.spark.ml.Pipeline
    import org.apache.spark.ml.feature.{RegexTokenizer, StopWordsRemover, CountVectorizer, … }
    :
    val dataDF = rawdata.select("id", "body")
    :
1)  val   cv = new CountVectorizer().setInputCol("terms").setOutputCol("feature1") / /feature1
    :
    val   si = new StringIndexer().setInputCol("category")
2)  val   ohe = new OneHotEncoder().setInputCol(si.getOutputCol).setOutputCol("feature2") // feature2
    :
    // feature1 + feature2
3)  val   va = new VectorAssembler().setInputCols(Array("feature1", "feature2")).setOutputCol("features")
    :
    dataDF.select("terms", "features").where($"category" === "gaming")
    :
```

Figure 4: Sample source codes for feature extraction process

In-memory distributed parallel processing of the Topic model

When LDA model is running, the entire document is distributedly stored in Hadoop cluster, and each slave node independently processes LDA model in parallel. In LDA model, Corpus (i.e., the bag of words) is created first through the process of separating and processing the contents for each word for the entire document $D\ (d_1, d_2, \ldots, d_n)$. That is, a $D * W$ matrix indicating only the frequency of the word is generated (see Figure 5). Next, in order to calculate the expected value of the hidden variable for each word, it is fixed as the frequency because the value of the parameter is not given at the beginning of the execution. And then, the parameter values obtained by executing the E-step and the M-step process are used repeatedly.

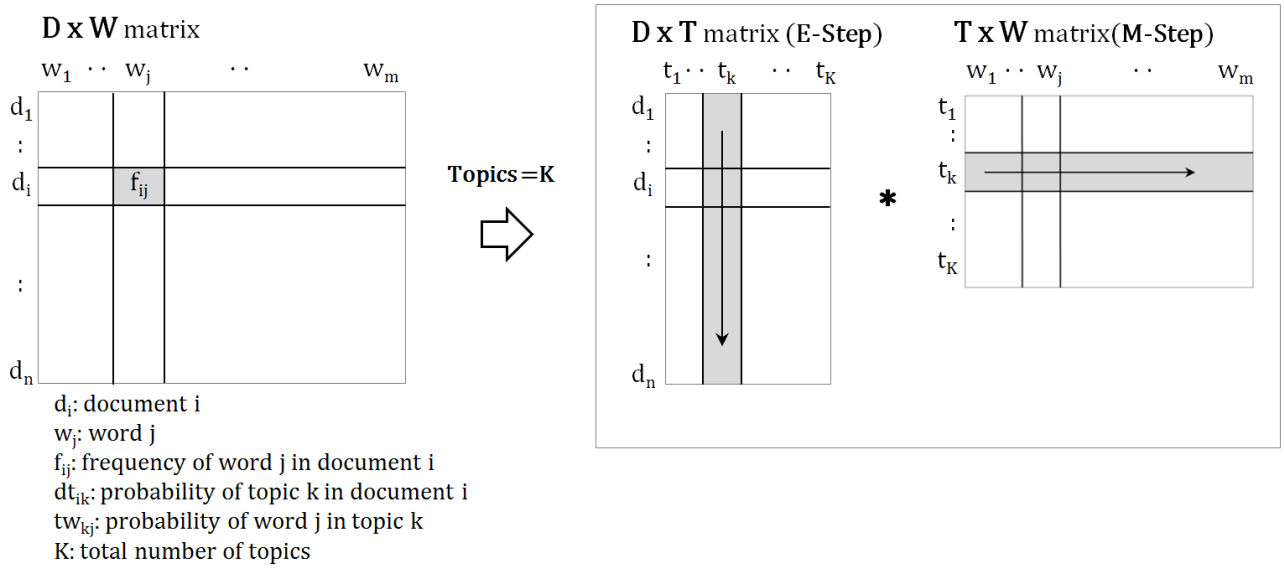


Figure 5: LDA calculation process used in Spark

Figure 5 shows the analysis of the entire document D into a total of K topics. The E-step is the process of calculating the expected value of the hidden variable using the current parameter. The M-step is the process of finding parameters that maximize the log likelihood by setting value found in the E-step to the observation value of the hidden variable. After completing M-step, a new parameter can be obtained from the previous parameters, this parameter goes to the E-step. The LDA model parameters are estimated while repeating E-step and M-step.

EXPERIMENTAL ENVIRONMENT AND TOPIC ANALYSIS RESULTS

The prototype cluster consists of a total of six slave nodes and one master node. It is constructed a distributed process platform by installing Hadoop 2.7 and Spark 2.0 on Ubuntu 14.04 LTS. 1) 384 GB memory (64 GB × 6 nodes) and 2) 64 TB HDD (8 TB × 6 nodes) are available for the cluster, 3) each slave node has i7 or i5 CPU processor (Skylake 3.2 GHz).

In order to experiment Spark LDA topic analysis, Spark LDA execution parameters are set as follows: driver program memory of the master node is 12 GB and executor memory of each slave node is 56 GB. There are 6 executors and each executor has 7 cores.

Feature extraction for unstructured text data

The social document used in the experiment is personal comment-related data of about 160 million records. It has 22 attributes and it is a json file format composed of key and value. Attributes of "id", "category", and "body" were selected for analysis. "id", "category", and "body" attribute represents the user ID, the group name to which each comment belongs, and the comment to be analyzed, respectively. In this experiment, preprocessing through Tokenizer and StopWordsRemover for comments corresponding to "body" attribute.

The results of data conversion for feature extraction of the DataFrame used in Spark SQL is shown in Figure 6. After preprocessing of input comment, refined words are obtained as shown in "terms" column of Figure 6. The refined words are transformed to vectors in order to use them as an input value of LDA. The feature 1 in Figure 4 shows the result after CountVectorizer processing for each comment, and the feature 1 means the frequency values of the words represented by the vector. On the other hand, the attribute "category" used as an input to the StringIndexer represents a group of original texts to which each comment belongs. It has string value represented a specific word. Later, it represented by a vector of feature 2. Finally, it is represented as the vector value of "features" column in Figure 6 by VectorAssembler which matches the values of two vectors.

id	category	body	terms	features
userxxx	hockey	https://twitter.c...	[https, twitter, ...	(97113,[4,47,655,...
userxxx	funny	How long does tha...	[long, usually, t...	(97113,[50,87,137...
userxxx	AskReddit	"oh, nice camera....	[oh, nice, camera...	(97113,[12,121,13...
userxxx	videos	Maybe have a talk...	[maybe, talk, tom...	(97113,[80,292,28...
userxxx	nfl	> Honestly, t...	[gt, honestly, fi...	(97113,[14,34,60...
userxxx	WTF	That marriage on ...	[marriage, druidi...	(97113,[9,28,39,4...
userxxx	television	> Then there's...	[gt, full, schedu...	(97113,[2,14,17,2...
...	...	...	...	...

Figure 6: The results of feature extraction applying Spark SQL DataFrame

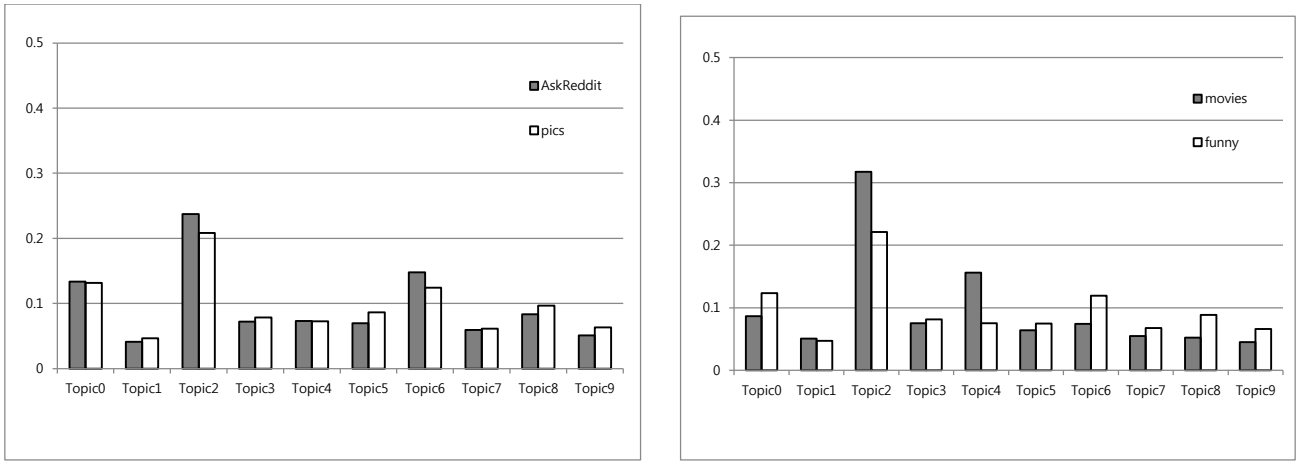
Results of topic analysis without 'category' attribute

Table 1 shows the results of topic analysis without "category" attribute. In Table 1, only 10 results were displayed in descending order of the weight of the word for each topic when performing LDA analysis after only the feature 1 process. The number of topics K was set to 10. It can be known that words derived for each topic are composed of general words not biased toward a specific filed because topic modeling were executed for all comments regardless of "category". Also, it is not easy to understand meaning for each topic in spite of comments are newly expressed as distribution of topic word through LDA execution.

Table 1: The results of topic analysis without 'category' attribute

Topic0	Topic1	Topic2	Topic3	Topic4	Topic5	Topic6	Topic7	Topic8	Topic9
deleted	com	really	de	one	http	will	game	can	com
people	watch	one	yes	think	org	know	people	good	http
get	www	well	now	game	wikipedia	people	still	thanks	www
think	youtube	can	can	get	en	can	get	one	reddit
will	http	people	get	can	wiki	get	one	get	amp
can	https	get	que	team	com	time	much	will	please
time	can	want	got	really	can	think	amp	even	comments
even	one	see	will	time	https	one	lol	go	message
good	get	love	la	will	one	make	name	now	https
one	also	good	fucking	good	us	want	games	also	post

Next, the weight average value of the topic distribution for each user group is used whether the topic shows the characteristics of the group. The topic distribution of the AskReddit, pics, movies, funny user groups is shown in Figure 7. In case of Figure 7 (a), it can be seen that the values of Topic 0, Topic 2, and Topic 6 are relatively high. However, it is difficult to grasp the features of an accurate user group (e.g., AskReddit, pics) by looking at the words for each topic in Table 1. In case of Figure 7 (b), it is difficult to find concrete words and topic that express the user group (e.g., movies) even though looking at the words in the highest Topic 2.



(a) Topic distribution for the group of AskReddit & pics (b) Topic distribution for the group of movies & funny

Figure 7: The weight of topic distribution before applying 'category' attribute

Results of hybrid topic analysis

Table 2 represents the number of comments of the top 20 user groups. The experimental results including "category" information is easy to grasp the characteristics of the group compared with the experiment results excluding "category" information because topic distribution for each group is concentrated on specific topic.

Table 2: Top 20 counts of input comments along with 'category'

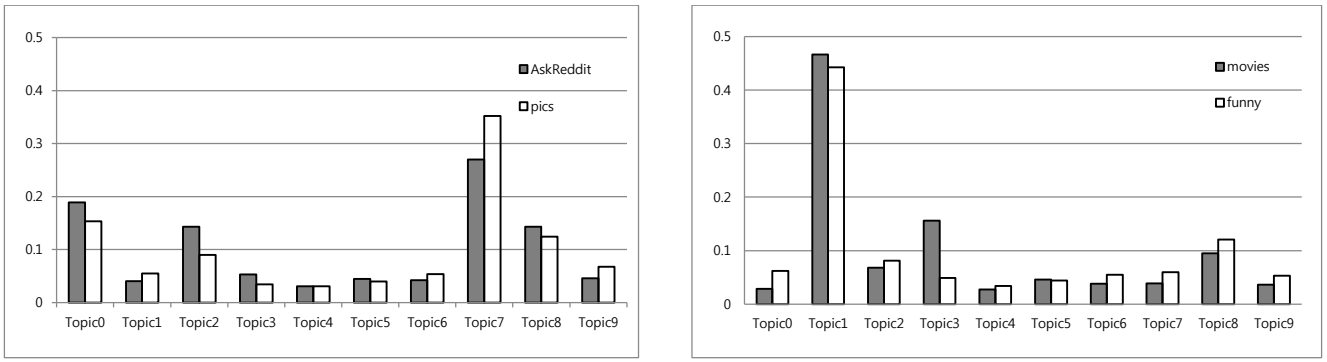
category	#comments	category	#comments	category	#comments	category	#comments
AskReddit	12,504,266	videos	1,700,974	Worldnews	1,557,302	WTF	1,229,188
leagueoflegends	3,266,978	nfl	1,628,609	Pcmaterrace	1,516,553	DotA2	1,222,178
funny	2,318,163	todayilearned	1,592,245	adviceanimals	1,406,310	DestinyTheGame	1,204,778
pics	1,998,732	news	1,589,967	soccer	1,350,930	gaming	1,190,526
nba	1,977,167	hockey	1,577,484	SquaredCircle	1,267,224	movies	1,087,207

Table 3 shows LDA processing performed on features including "category" information to which comments belong. It is possible to know that the values of "category" of the user groups has been reflected in the word distribution (underlined words in Table 3) for each topic. At this time, it is possible to know that the words of AskReddit, funny, pics user groups which are the high count of Table 2 are located in the top rank (i.e., high weight) even in the topic word distribution. As a result, it was confirmed that the hybrid topic analysis arranged the words so that the contents of the topic can be expressed more concretely. Also, by looking at the words "AskReddit" and "leagueoflegends" in Topic 0 and "funny" and "movies" in Topic 1, and "AskReddit" and "pics" in Topic 7, it was confirmed that the comments belonging to that group were related to each other only by the word distribution of 10 topics.

Table 3: The results of hybrid feature extraction topic analysis

Topic0	Topic1	Topic2	Topic3	Topic4	Topic5	Topic6	Topic7	Topic8	Topic9
deleted	*funny*	people	game	thanks	can	com	*AskReddit*	one	can
AskReddit	org	think	watch	de	people	http	love	get	will
get	wikipedia	*AskReddit*	can	oh	get	www	*pics*	can	one
will	en	one	one	que	good	amp	awesome	really	get
leagueoflegends	wiki	can	time	now	game	reddit	now	know	also
time	deleted	know	get	*pcmasterrace*	way	https	get	think	good
think	thank	will	*nba*	one	will	comments	well	good	people
year	really	even	play	man	much	please	one	time	need
one	*movies*	way	games	la	really	message	lol	people	use
people	know	time	good	really	make	post	know	going	new

On the other hand, in almost user groups, topics that can represent groups are clearly distinguished. Figure 8 (a) compares each comment belonging to the "AskReddit" group and "pics" group of Topic 7. Users in the two groups have a similar tendency in the distribution weight of the topics. The "pics" group shows that the characteristics of Topic 7 can grasp the strength even more. Figure 8 (b) shows a comparison between "funny" group and "movie" group displayed by Topic 1. The strength of Topic 1 can be confirmed in two groups.



(a) Topic distribution for the group of AskReddit & pics (b) Topic distribution for the group of movies & funny

Figure 8: The weight of topic distribution after applying 'category' attribute

Through this experiment, when adding features separated by "category" in the process of feature extraction for LDA analysis of unstructured text data, it could improve the understanding of the entire document because the result value of the word distribution and the topic distribution for each document was refined for each topic.

CONCLUSIONS

In this paper, in order to grasp topics of unstructured text data easily and quickly, designing and implementing a topic analysis system based on LDA model which can analyze topics without prior learning. In the preprocessing stage of the input data set, feature vector extracted from text string and feature vector partitioned into categories are combined (hybrid). Then, the combined feature vector was used as the input value of the LDA model for topic extraction. As a result, the proposed system was able to automatically execute topic analysis on the entire document.

Since the proposed system was implemented based on the Spark platform on the Hadoop cluster, it can minimize disk access for processing data and process all intermediate output results on memory. As a result, the proposed system was able to obtain high performance of processing. In the experiments for performance evaluation, it took about 1 hour and 30 minutes to topic analysis of 160 million unstructured text data

(e.g., comments on SNS). Using the weights of the word distribution of each topic containing "category" information, it was able to more clearly classify the subject of the document to the extracted topic. In subsequent research, it is necessary to design an operation platform for processing unstructured data corresponding to the stream input data and develop various topic analysis algorithms related to the platform.

ACKNOWLEDGEMENT

This paper is supported by Ajou University Research Fund.

REFERENCES

[1] D. Blei, "Probabilistic Topic Models," Communication of the ACM, Vol. 55, No. 4, pp. 77-84, 2012.

[2] K. Park, C. Baek and L. Peng, "A Development of Streaming Big Data Analysis System Using In-memory Cluster Computing Framework: Spark," In Advanced Multimedia and Ubiquitous Engineering, pp. 157-163. Springer Singapore, 2016.

[3] M. Zaharia, M. Chowdhury, M. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster Computing with Working Sets," In HotCloud, 2010.

[4] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," In Proceedings of the 6th Symposium on Operating System Design and Implementation, pp. 137-150, 2004.

[5] M. Armbrust, R. Xin, C. Lian, Y. Huai, D. Liu, J. Bradley, X. Meng, T. Kaftan, M. Franklin, A. Ghodsi, and M. Zaharia, "Spark SQL: Relational Data Processing in Spark," In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1383-1394, 2015.

[6] K. Park and L. Peng, "A Design of High-speed Big Data Query Processing System for Social Data Analysis: Using SparkSQL," International Journal of Applied Engineering Research, Vol. 11, No. 14, pp. 8221-8225, 2016.

[7] T. Hofmann, "Probabilistic Latent Semantic Indexing," In Proceedings of the 22nd Annual International SIGIR Conference on Research and Development in Information Retrieval, pp. 50-57, 1999.

[8] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet Allocation," Journal of Machine Learning Research, Vol. 3, pp. 993-1022, 2003.

[9] M. Hoffman, D. Blei, and F. Bach, "Online Learning for Latent Dirichlet Allocation," Journal of Advances in Neural Information Processing Systems 23, pp. 856-864, 2010.

[10] V. Vavilapalli, A. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, and B. Saha, "Apache Hadoop YARN: Yet Another Resource Negotiator," In Proceedings of the 4th annual Symposium on Cloud Computing ACM, pp. 5:1-5:16, 2013.

[11] J. Park and H. Oh, "Distributed Online Machine Learning for Topic Models," Communications of the Korean Institute of Information Scientists and Engineers, Vol. 32, No. 7, pp. 40-45, 2014. (in Korean)

[12] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," In Proceedings of the 26th IEEE Transactions on Computing Symposium on Mass Storage Systems and Technologies, pp. 1-10, 2010.

[13] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica, "Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing," In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, 2012.