LDA Team Project in CS5604

Spring 2015

Extracting Topics from Tweets and Webpages for IDEAL

May 10, 2015

by

Sarunya Pumma and Xiaoyang Liu

LDA Team

Instructor

Prof. Edward Fox

Virginia Tech, Blacksburg, Virginia

Abstract

IDEAL or Integrated Digital Event Archiving and Library is a project of Virginia Tech to implement the state-of-the-art event-based information retrieval system. A practice project of CS 5604 Information Retrieval is a part of the IDEAL project. The main objective of this project is to build a robust search engine on top of Solr, a general purpose open-source search engine, and Hadoop, a big data processing platform. The search engine can provide a set of documents, which are tweets and webpages, that is relevant to a query word that a user provides. To enhance the performance of the search engine, the documents in the archive have been indexed by various approaches including LDA (Latent Dirichlet Allocation), NER (Name-Entity Recognition), Clustering, Classification, and Social Network Analysis. As CS 5604 is a problem-based learning class, an implementation and a development of each technique has been assigned to each team in the class. In this report, the implementation of the LDA component is presented. LDA is responsible for extracting collections of topics from the documents. A topic in this context is a set of words that can be used to represent a document. We introduce background knowledge, design, tutorial for related packages, results and evaluation. The results are got from both tweet and webpages collections. And evaluation part contains different method to evaluate the performance of LDA. Based on the result and background knowledge, we give a conclusion of our work. Also a user manual and a developer manual are presented for readers.

Contents

1. Introduction 6
2. Background Knowledge and Literature Review 8
   2.1. Important Chapters of the Textbook for Our Team 8
   2.2. Literature Review 9
3. Work Plan and Deliverables 14
   3.1. Overall Plan 14
   3.2. Deliverables 15
4. Requirements 16
   4.1. Information Retrieval System Overview 16
   4.2. LDA Component Functional Requirements 17
   4.3. Analysis of Inter-Dependencies between Teams 18
5. Detailed Design 20
6. Implementation 22
   6.1. Input Data Preprocessing 22
      6.1.1. Tweets Preprocessing 22
      6.1.2. Webpages Preprocessing 23
   6.2. LDA processing 24
      6.2.1. Invoking LDA from Java 24
      6.2.2. Selection of Number of Topics 26
         6.2.2.1. Empirical Study For A Small Tweet Collection 27
         6.2.2.2. Empirical Study For A Big Tweet Collection 28
         6.2.2.3. Empirical Study For A Small Webpage Collection 29
   6.3. Output Data Storing 30
7. LDA Evaluation 32
   7.1. Human Judgement 32
      7.1.1. Word Intrusion 32
      7.1.2. Topic Intrusion 32
      7.1.3. Evaluation Result 33
   7.2. Cross-Validation with the Clustering Team 34
8. Topics for All Collections 37
   8.1. Jan 25 38
   8.2. Charlie Hebdo 38

8.3. Ebola 39
8.4. Election 39
8.5. Plane Crash 40
8.6. Suicide Bomb Attack 40
8.7. Winter Storm 41
8.8. Malaysia Airlines 41
8.9. Bomb 42
8.10. Diabetes 42
8.11. Egypt 43
8.12. Shooting 43
8.13. Storm 44
8.14. Tunisia 44
8.15. Conclusion 45
9. User Manual 45
9.1. Basic HDFS Usage 45
9.2. Compile and Run the Hadoop Program to Convert AVRO/Text File to Sequence File/Text File 45
9.3. Compile and Run the Hadoop Program to Invoke LDA 50
9.4. Read AVRO File 57
10. Developers manual 59
10.1. Solr Installation 59
10.2. Importing Example Documents to Solr 60
10.3. Crawling Web Contents using Python Script 63
10.4. Hadoop Installation 64
10.4.1. SSH Access Set Up 64
10.4.2. Hadoop Installation 64
10.5. Mahout Installation 67
10.6. Nutch Installation and Tutorial 67
10.7. Crawl Webpages from Small Collection of Tweets 68
11. Conclusion 70
12. Acknowledgements 70
13. References 71

14. Appendix 73
14.1. Appendix A: Questionnaire 73

1. Introduction

This report is part of CS 5604 Information Retrieval, of which the purpose is to design and construct a robust search engine based on Solr. Our team focuses on the application of LDA to extract topics from tweets and webpages. In this report, background knowledge is presented in section 2. 2.1 is information about related chapters in the textbook and 2.2 is a literature review, which includes most important knowledge based on several papers related to LDA. In requirements section, We introduce the relationship between our team and other teams. 4.1 is the overview of the whole system and 4.2 is about LDA component. Inter-Dependencies are presented in 4.3 to give information about the cooperation. Detailed Design of LDA is discussed in section 5, which contains the our design for the LDA component. Based on the knowledge of the chapters talked above, we present implementation in section 6. 6.1 is about input data preprocessing, which includes the preprocessing of both tweet and webpage collections. 6.2 is about LDA processing. In this section, we present a Java program in 6.2.1 to evoke LDA more easily. Several empirical studies on different collections are given in 6.2.2 to find the best number of topics. 6.3 is introduced for output data storing. After getting results from LDA, we introduce two evaluation method to evaluate the performance of our LDA. 7.1 is talking about evaluation based on human judgement and 7.2 is cross-validation with the Clustering Team. In section 8, we present the results of different collections and a conclusion based on these results. From the conclusion, reader can easily find the work we haven't finished yet. A user manual and a developer manual are also included in this report. In the user manual section, we discuss necessary knowledge for using LDA component. This knowledge contains basic HDFS usage, convert file formats, invoke LDA methods and method to read a specific type of file, AVRO. Section 10 is Developer manual. In this section, we present required information for construct a LDA component based on our experience. It includes Solr installation and a simple tutorial for importing example documents to Solr, which is in 10.1 and 10.2. 10.3 talks about crawling webpages using Python script, and 10.6 provides another method Nutch to crawl webpages. We also introduce an example of crawling webpages from small collection of tweets. 10.5 is about package Mahout, we presents a procedure for installation and a tutorial for using Mahout LDA.

This report contains three parts. Firstly, a literature review and related references are provided for readers to understand the algorithm of LDA and if needed, to know more details about the LDA model. Secondly, we give the overview of our work. From this part, readers will have an overall knowledge of our work to understand the function of LDA in the whole project and know the relations and inter-dependencies between our team and other teams, such as the Solr team. Finally, we describe details of the work, which include installations of certain packages, methods for applying LDA in Mahout and indexing data and the detailed design of our team's project. Also, a user manual and a developer manual are provided in this report to help people, who want to work on the same project, to get started quickly.

Overall, this report includes almost all the information about the LDA application to construct a search engine based on Solr. We hope the content in this report is beneficial to readers.

2. Background Knowledge and Literature Review

This section provides background knowledge and literature review for LDA.

2.1. Important Chapters of the Textbook for Our Team

According to the tasks mentioned above, the related chapters are Chapter 6, Chapter 7, Chapter 8, and Chapter 18. Among them, Chapter 6 and Chapter 7 are the most important chapters.

Since TF-IDF is a fundamental weight used in general information retrieval systems, we are required to understand the basic knowledge about it. Chapter 6, which is “scoring, term weighting and the vector space model”, talks about this. Studying this chapter will give us an idea about the importance of document indexing and scoring, for example, the importance of the TF-IDF weight. In Chapter 6, there are three important topics that we are certain will be useful for our work. The first one is parametric and zone indexes which will allow us to index and rank the documents in response to the query. The second one is the idea of weighting the importance of a term in a document. The third one is about the vector space model, which is a fundamental of information retrieval operations ranging from scoring documents based on a query, document classification to document clustering. Similar to Chapter 6, Chapter 7 gives details of the advanced scoring and ranking methods. Understanding the techniques that are described in this chapter will help us to create an efficient scoring and ranking method based on LDA.

Moreover, we need to learn more about LDA. Although this textbook does not provide much detail about LDA, it provides useful references to study extensively. Since LDA is an initial probabilistic extension of the latent semantic indexing (LSI) technique, it is necessary for our team to review Chapter 18, especially the LSI part.

In order to evaluate our LDA system, we need to study the evaluation methods from Chapter 8. This chapter provides many methods for evaluating different kinds of information retrieval system.

The following is the list of papers and websites that we are going to study:

Papers

1. Latent Dirichlet allocation [1]
2. LDA-based document models for ad-hoc retrieval [2]
3. Solr's reference guide [3]

Websites

1. Lucene [4]

Books

1. Pattern Recognition and Machine Learning [6]
2. An Introduction to Information Retrieval [7]

We also need to read the BodyOfKnowledge in order to understand more about LDA, especially Section 5: Topic. Section 5 gives the general idea of LDA and useful references that we should read including:

1. Topic Discovery through Data Dependent and Random Projections [11]
2. Topic discovery based on text mining technique [12]
3. Topic Discovery, Tracking, and Characterization of Social Media Conversation [13]
4. Towards an NLP-Based Topic Characterization of Social Relations [14]

2.2. Literature Review

Latent Dirichlet allocation (LDA) is a model that allows sets of observations to be explained by unobserved groups that can represent the similarity of the data. For example, in the case that an observation is a collection of words in a document, LDA extracts a set of words that are likely to be able to describe the document and uses this set of words to represent the document [8].

The first and most important paper about LDA is "Latent Dirichlet Allocation", which is written by David M. Blei, Andrew Y. Ng and Michael I. Jordan. However, it is not straightforward and actually hard to be understood by beginners. It is necessary to read papers and books to know the background of LDA since LDA, in general, is a topic model. So the first thing for us is to know more about topic model and what LDA will do as a topic model.

A topic model is a mixture idea from computer science, mathematics and other fields, which uses Bayesian statistics and machine learning to discover the latent, or hidden, structure in given documents and build the links between documents [4]. Then based on this knowledge, topic models can also give predictions on future documents. Because of these, topic models are powerful tools, which are used to understand information that seems not related to each other and even chaotic. LDA is an outstanding one in the family of topic models. And how does LDA work? In general, LDA gives two probability distributions. One is the probability distribution over words in the same topic and another is the probability distribution over topics in the document. For example, in the topic of "sports", there will be some words, such as "football", "basketball", "soccer", occur together frequently, and then LDA gives the probability distribution over these words. Also, by examining the documents, one document can be said as 100% about "sports" or 50% about "sports" and 50% about "politics".

In more detail, the algorithm of LDA can be illustrated in words. LDA represents documents as mixtures of topics that spit out words with certain probabilities. It assumes that documents are produced in the following way:

- Decide on the number of words N the document will have (say, according to a Poisson distribution).
- Choose a topic mixture for the document (according to a Dirichlet distribution over a fixed set of K topics).
- Generate each word w_i in the document by:
  1. Picking a topic (according to the multinomial distribution that you sampled above).
  2. Using the topic to generate the word itself (according to the topic's multinomial distribution).

It is obvious that LDA has a three-layer structure: words -> topics -> documents. From documents to topics, it follows the Dirichlet distribution. From topics to words, it follows the multinomial distribution [9].

Before giving the model of Blei's paper, we also need to illustrate two things. One is Bayesian statistics. The most important thing about the Bayesian method is the way of thinking, which is a little different from common sense. It can be described as following:

prior distribution $\pi(\theta)$ + sample information X => Posterior distribution $\pi(\theta|x)$

It tells us the observed sample information will update the knowledge we have for the objects. To be specific, the information will correct the value of parameter θ. It is the process of machine learning for LDA.

The other is the pLSA (Probabilistic latent semantic analysis) model. It can be described in Figure 1.

We have a deterministic topic distribution: {education:0.5, economy: 0.3, transportation: 0.2}, and for each topic we have a deterministic distribution over words, for example, education {university: 0.5, professor: 0.3, course: 0.2}. To get one word for the document, we choose one topic for the topic distribution and then from this topic we choose one word. Repeating this process for N times, we can get an N words document.

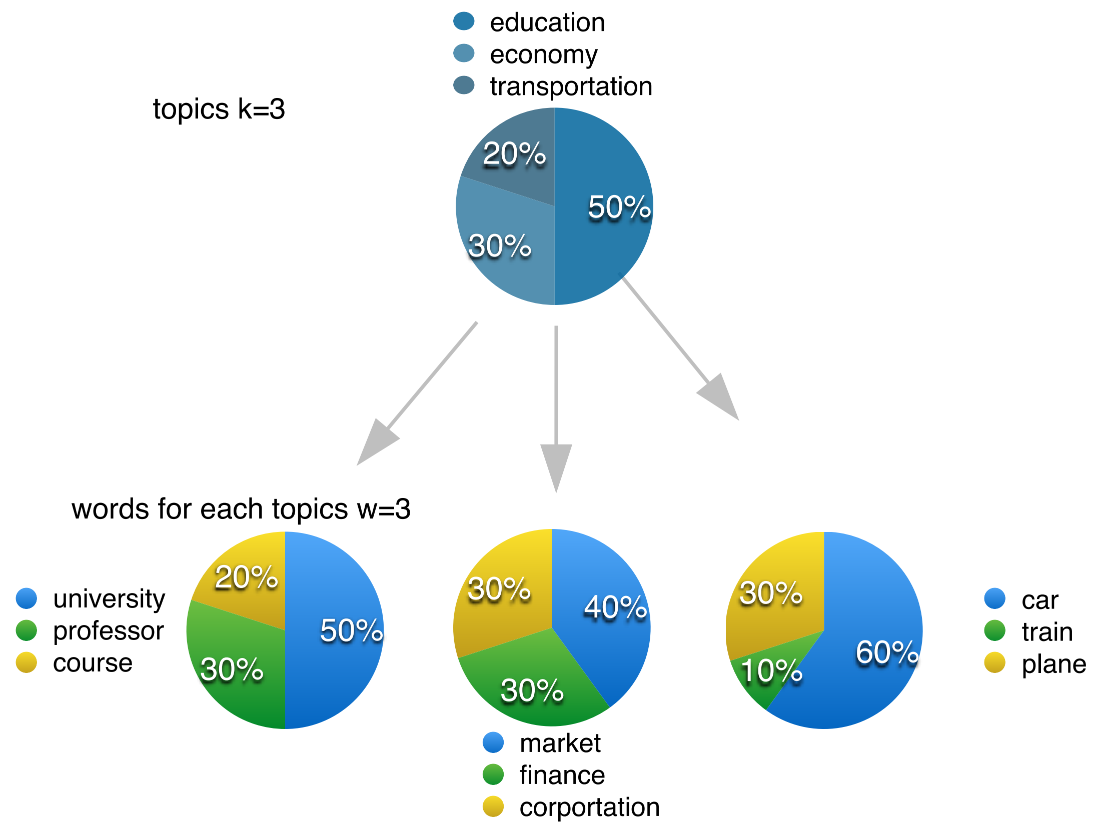


FIGURE 1 PLSA

Now it is the right time to talk about the exact LDA model in Blei's paper. Actually, the LDA model is the combination of pLSA model and Bayesian statistics. In the paper "Latent Dirichlet Allocation" [10], the authors give the LDA model using the mathematical and graphic model as shown in Figure 2.

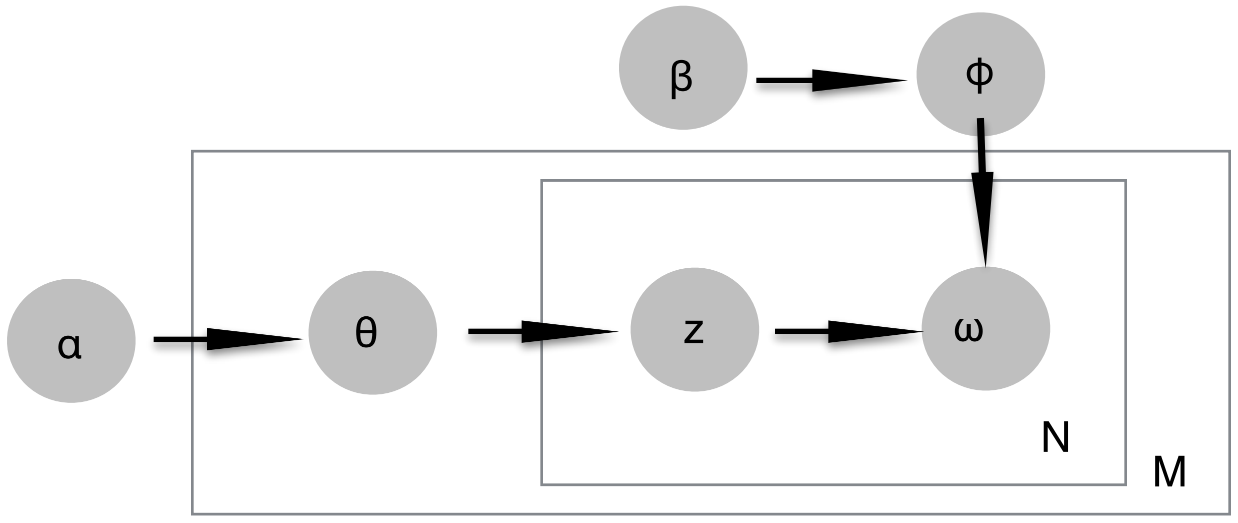


FIGURE 2 GRAPHICAL MODEL REPRESENTATION OF LDA. THE BOXES ARE "PLATES" REPRESENTING REPLICATES.[1]

The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document

As a mathematical model, LDA assumes the following generative process for each document w in a corpus D:

1. Choose N ~ Poisson(ξ).
2. Choose θ ~ Dir(α).
3. For each of the N words w_n:
   A. Choose a topic z_n ~ Multinomial(θ).
   B. Choose a word w_n from $p(w_n|z_n,\beta)$, a multinomial probability conditioned on the topic.

It is obvious that we get the LDA model by applying Bayesian statistics on the pLSA model. The only difference is that in the pLSA model, the probability distribution of topics and the probability distribution of words for each topic are determined. But in the LDA model, the distributions are not determined and follow the Dirichlet Distribution. The process is shown in Figure 3.

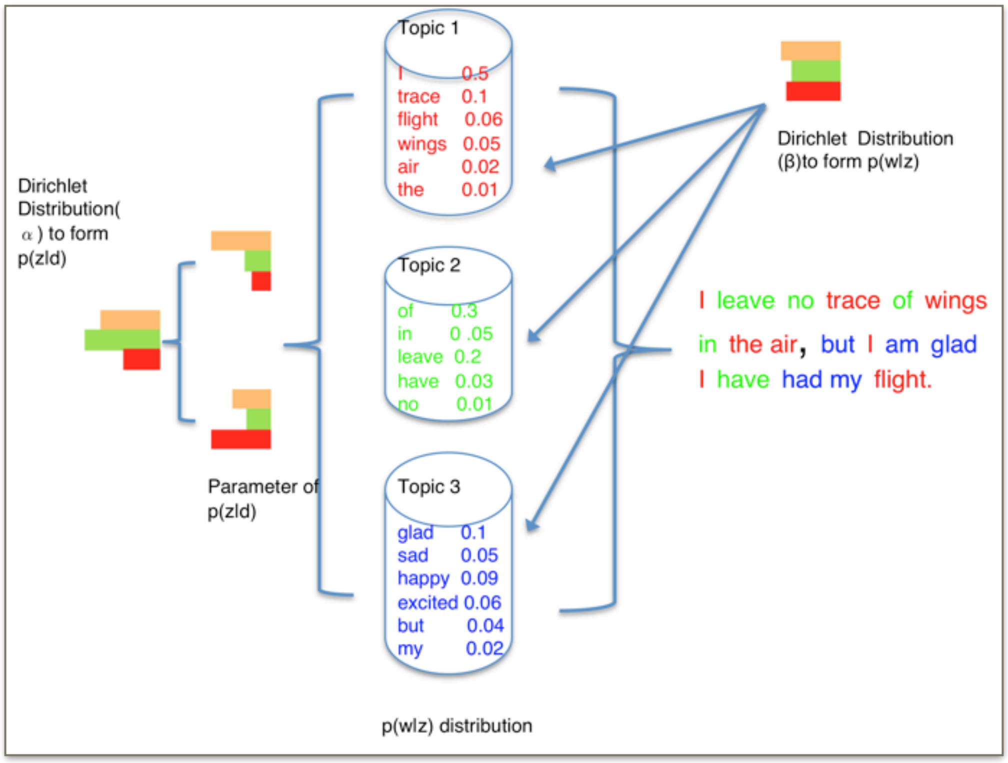


FIGURE 3 BAYESIAN STATISTICS ON PLSA MODEL

Now we have given the process for forming a document using LDA. However, we can think in another way because documents and words are obvious and topics are not. So how can we infer the topics from given documents and words? We will talk about this later.

Now, we think in another way. The article is here and how can we know the distribution of topics and the distribution of words in a particular topic. What we need to do is to estimate the two arguments in LDA model. There are different ways to calculate arguments of Φ and Θ. Among them, we want to talk a little about Gibbs Sampling, which is based on Bayesian idea. Gibbs Sampling algorithm is a little complex in mathematics. But we could illustrate the general idea of it using a picture.

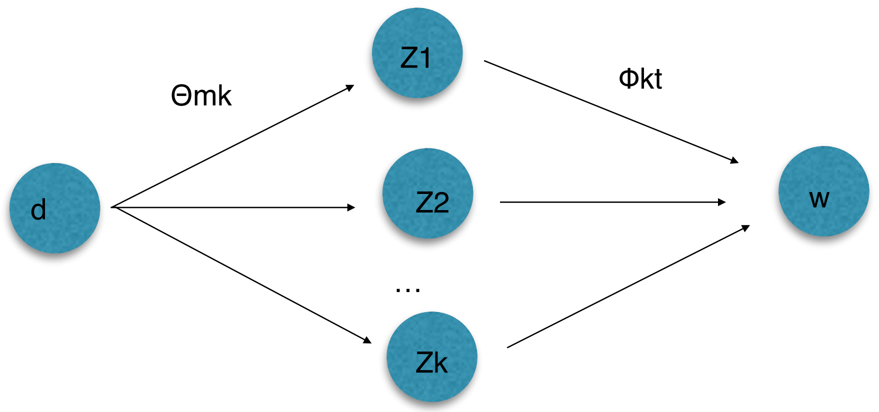


FIGURE 4 BAYESIAN STATISTICS ON PLSA MODEL

P(topicIdoc) * P(wordItopic) is the probability for the way doc—>topic—>word. So k topics are related to K routes so Gibbs sampling will get samples from these k routes. In this way, by calculating the Posterior distribution of distributions of topics and words, we successfully know the distributions of topics and words, which are related to unknown arguments.

In information retrieval, researches have introduced a couple of models solve the classic problem. Among them, LDA is a successful one and popular these days for its good quality in extending, good performance and so on. LDA can be extended to be achieved by using another distribution on the simplex instead of Dirichlet. And it can also be extended to a hierarchical LDA(hLDA), in which topics are collected together in a hierarchy. The LDA model is a Bayesian version of pLSA model, which make it has a good performance on small datasets. This advantage comes from the good quality of Bayesian method that it will not overfit data. And for big dataset, performances of pLSA and LDA are almost the same. So LDA is widely used nowadays.

3. Work Plan and Deliverables

3.1. Overall Plan

The following Gantt chart shows our work plan for the entire semester. The highlighted blocks in the chart indicate the amount of time (in weeks) that we require to complete the tasks.

The work plan spans from February to May:

- Week 1 - Week 4 are in February
- Week 5 - Week 9 are in March
- Week 10 - Week 11 are in April

Week	1	2	3	4	5	6	7	8	9	10	11	12
Review literature and related documents	🟧	🟧										
Understand the overall information retrieval system	🟧	🟧										
Install Solr and import tweets and webpages into Solr		🟧										
Analyze the LDA component requirements		🟧	🟧									
Design the LDA component			🟧	🟧	🟧	🟧	🟧					
Study tools and library that can be used to implement the LDA component				🟧	🟧	🟧	🟧	🟧				
Use the tool/library to implement the LDA component (separate from Solr). Write Java code to run LDA on Mahout					🟧	🟧	🟧	🟧	🟧			
Test the LDA component (unit testing)								🟧	🟧	🟧		
Fix the problems found in testing								🟧	🟧	🟧		
Collaborate with the Solr and Hadoop teams to integrate the LDA component in Solr									🟧	🟧	🟧	
Test the LDA component after integrated in Solr										🟧	🟧	
Fix the problems found in testing										🟧	🟧	
Evaluate the quality of LDA										🟧	🟧	
Present the final system											🟧	
Submit the final report and all deliverables												🟧

3.2. Deliverables

The deliverables of our project are as follows:

Due Date	Deliverable
Weekly	Weekly report
March	Midterm presentation
April	Final presentation
May	Final report
May	The LDA component integrated with Solr

4. Requirements

This section provides the detailed requirements of the LDA component. The first subsection describes the details of the overall information retrieval system and shows where the LDA component will fit into the big picture. The second subsection provides the functional requirements of the LDA component. The third subsection analyzed the inter-dependencies between our team and other teams. The last subsection shows the deliverables and a work plan.

4.1. Information Retrieval System Overview

The highest goal of the Information Retrieval class is to build a robust search system in support of the Integrated Digital Event Archiving and Library (IDEAL) project. Every team in the class works in collaboration with each other to implement this innovative information retrieval system. Similar to Google, this system will allow users to search for documents that relate to events of interest using query strings. The documents in this context are tweets and webpages.

The search engine for the IDEAL project will be built on top of Solr, an open source search engine. Moreover, Solr will run on Hadoop, a big data processing platform, in order to accelerate searching performance. To allow the users to efficiently search for the required documents based on the events, various classification and clustering technique including LDA, NER, Noise Reduction, Classifying Types, Extraction & Feature Selection, Clustering, and Social Network Analysis, will be integrated in the system.

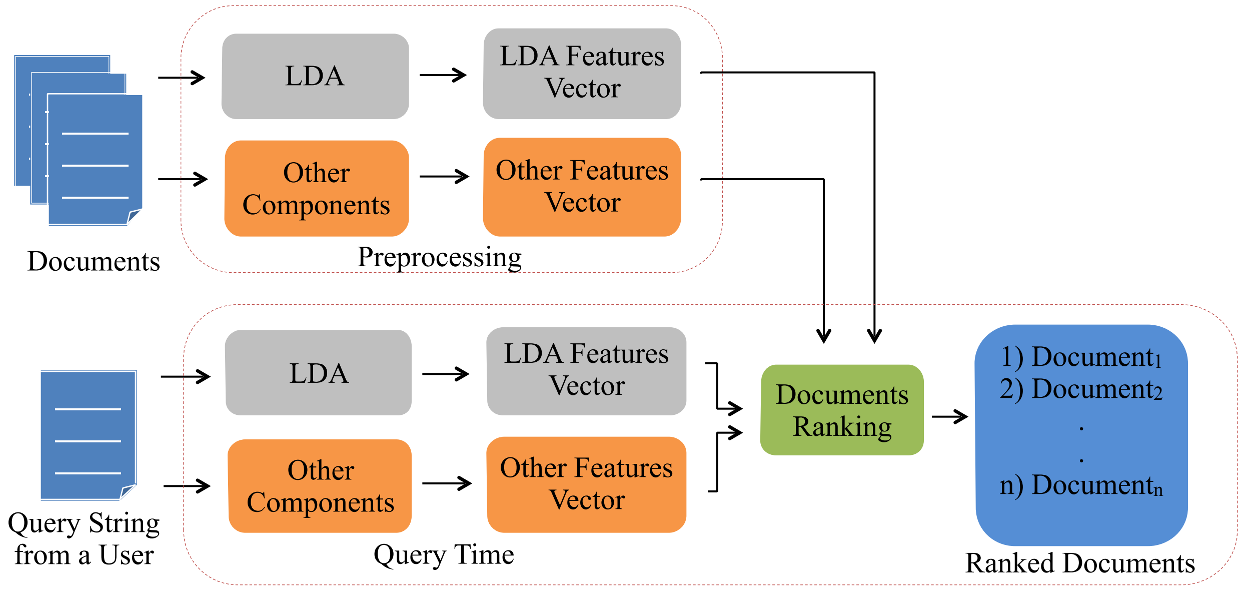


FIGURE 5 THE INTEGRATION OF LDA IN SOLR

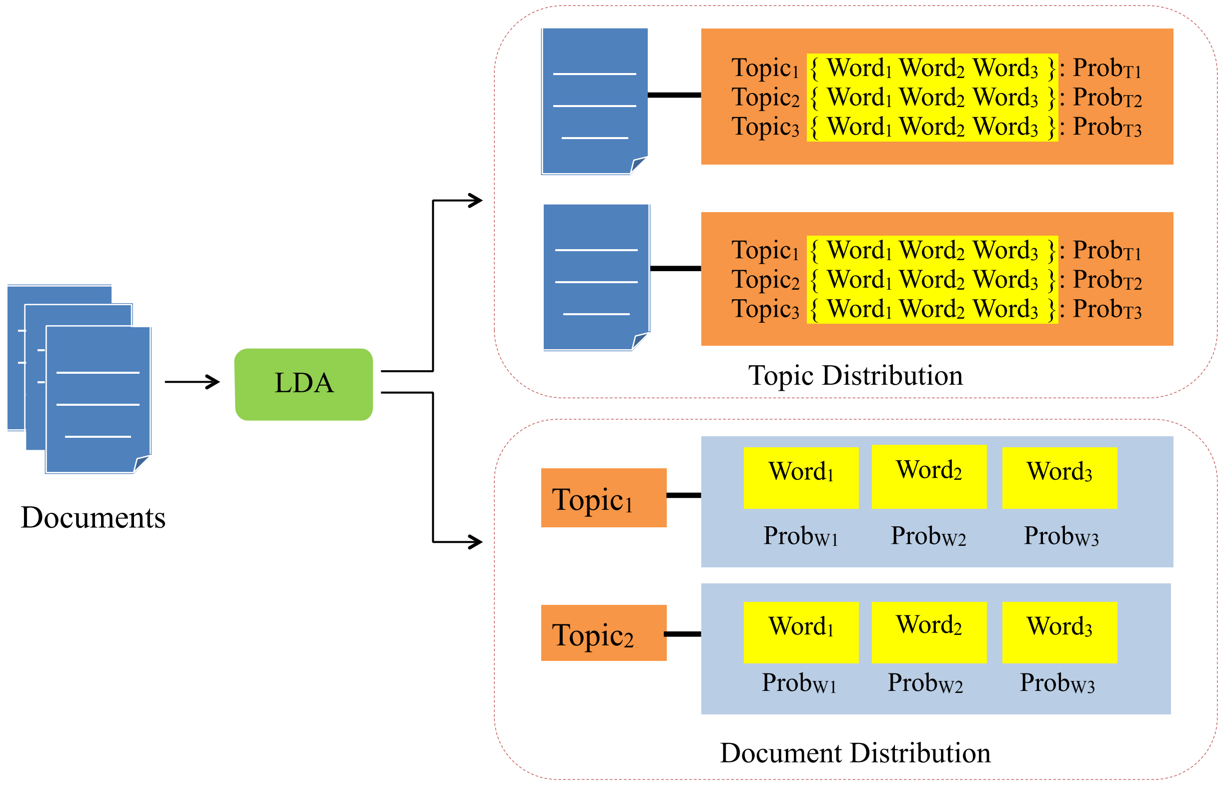
Our work focuses on the LDA component in this information retrieval system. Latent Dirichlet Allocation or LDA is an algorithm that provides the set of topics for the given input documents and determines the topic distribution for each document. In our work, the LDA component will be integrated in Solr as shown in Figure 5.

Figure 5 illustrates the workflow of the overall information retrieval system. The flow can be divided into 2 major parts: preprocessing and query time processing. The preprocessing phase performs before query time processing takes place. The preprocessing phase can be considered as offline processing, while the query time processing is online processing.

In the preprocessing phase, the system takes the documents (tweets and webpages) as the input and creates features vectors (i.e., LDA features vector) using LDA and other components (original Solr scoring, NER, Classifying Types, Extraction & Feature Selection, Clustering, and Social Network Analysis). These features vectors will be stored in the Solr database and will be used further in the query time processing.

The query time processing occurs when a user performs searching. The query string from the user will be used to create the feature vectors as in the preprocessing phase. Then the vectors from this step will be compared to the preprocessed vectors of each document in order to evaluate the importance of the document. The documents will be ranked based on their importance. Then, the documents with high importance will be shown to the user.

4.2. LDA Component Functional Requirements



Documents
LDA
Topic$_1$ { Word$_1$ Word$_2$ Word$_3$ }: Prob$_{T1}$
Topic$_2$ { Word$_1$ Word$_2$ Word$_3$ }: Prob$_{T2}$
Topic$_3$ { Word$_1$ Word$_2$ Word$_3$ }: Prob$_{T3}$
Topic$_1$ { Word$_1$ Word$_2$ Word$_3$ }: Prob$_{T1}$
Topic$_2$ { Word$_1$ Word$_2$ Word$_3$ }: Prob$_{T2}$
Topic$_3$ { Word$_1$ Word$_2$ Word$_3$ }: Prob$_{T3}$
Topic Distribution
Topic$_1$
Word$_1$ Word$_2$ Word$_3$
Prob$_{W1}$ Prob$_{W2}$ Prob$_{W3}$
Topic$_2$
Word$_1$ Word$_2$ Word$_3$
Prob$_{W1}$ Prob$_{W2}$ Prob$_{W3}$
Document Distribution

FIGURE 6 INPUT AND OUTPUT OF LDA

LDA processes tweets and webpages in order to find the topic distribution for each document and also the document distribution for each topic. Figure 6 shows the results from LDA after processing the documents.

The input of LDA is a set of documents (both tweets and webpages). Our LDA component is required to process the documents and produce the following outputs:

1. *Topic distribution:* LDA uses probability value ($Prob_{T_n}$ in Figure 6) to indicate the importance of topics in the document (each topic consists of words {$Word_1$ $Word_2$ $Word_3$} in Figure 6). The topic distribution is one of the features of the document.
2. *Word distribution:* each topic will contain different words. The probability value ($Prob_{W_n}$ in Figure 6) will be assigned to each word in order to show the importance of the word to the topic.

4.3. Analysis of Inter-Dependencies between Teams

In this section, we present the inter-dependencies between the teams in the CS5604 class. The dependencies are presented in Figure 7.

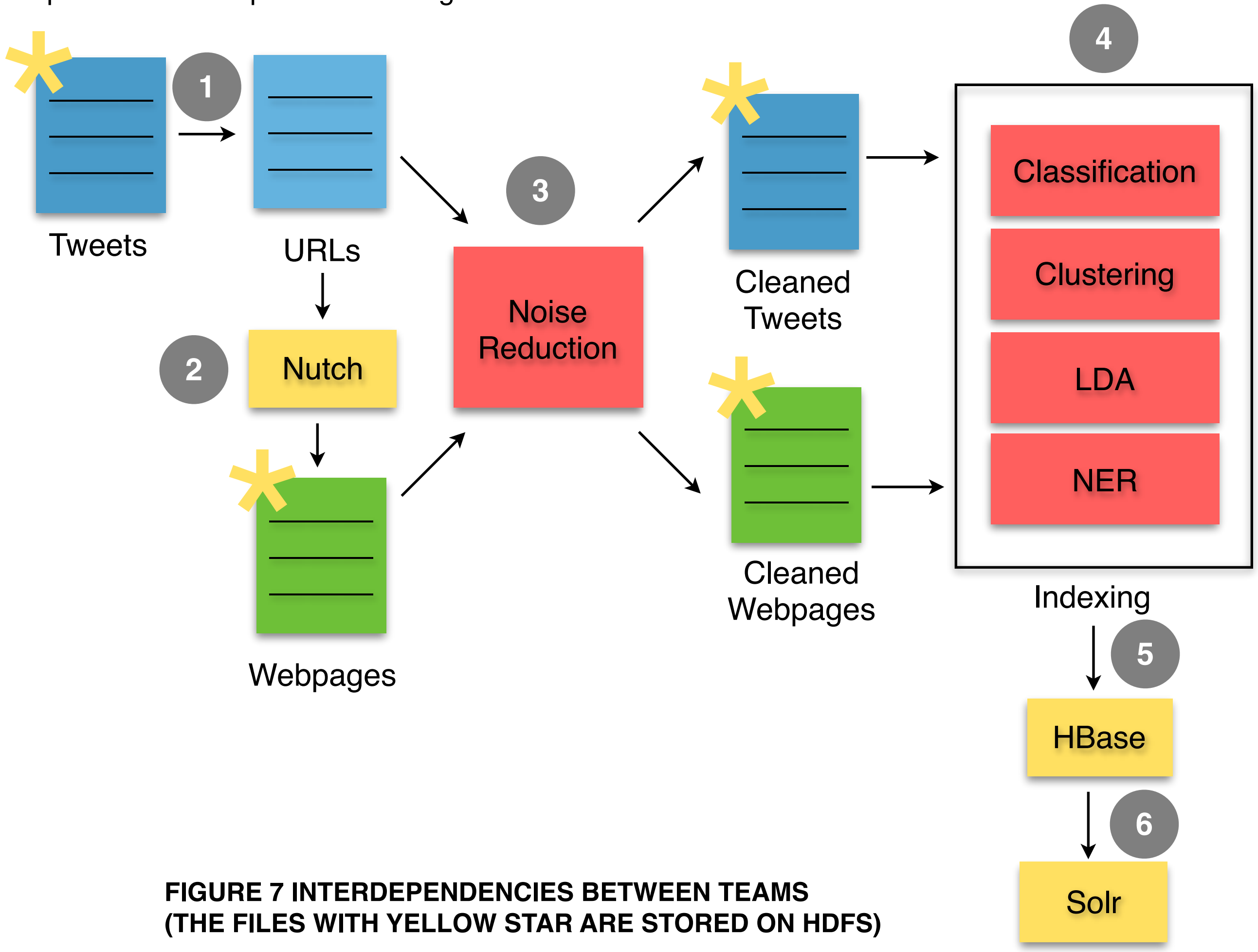


FIGURE 7 INTERDEPENDENCIES BETWEEN TEAMS
(THE FILES WITH YELLOW STAR ARE STORED ON HDFS)

From Figure 7, we will describe the dependencies based on the numbers that labeled in the diagram. Notice that tweets, webpages, cleaned tweets, and cleaned webpages are stored on HDFS (Note that the files are marked with the yellow star in the diagram).

1. The URLs of webpages are extracted from the tweets by using the Python script from the TA. The script gets the short URLs and convert them to the original URLs. The list of URLs is contained in the simple text file.

2. The obtained URLs from the tweets are fed to Apache Nutch, a powerful web crawler, in order to get the contents of the webpages.

3. The tweets and webpages are cleaned by using the script provided by the noise reducing team. As of now, we are going to follow the steps in the user manual of the noise reducing team's report in order to clean our small and large collections.

4. The cleaned tweets will be fetched from HDFS and indexed by the classification team, the clustering team, the NER team and our team.

    - The clustering team will group the documents into clusters in order to improve searching performance. The technique that they will use is the top-down clustering approach.

    - The classification team will apply some techniques to extract the important features from the tweets and webpages. Then, they will categorize the data into the predefined categories based on the extracted features from the previous step. These categories will be additional metadata which can efficiently improve a query process.

    - The NER team will be responsible for extracting the entities from the documents and classifying them into predefined categories. Once they successfully classify the documents, the classes will become important metadata in the query process.

    - Our team is responsible to extract the topics from the tweets and webpages. Thus, we can

5. For our team, after getting the extracted topics, we have to store the topics and the document-topic distribution in HBase. The Hadoop team is responsible for creating the schema and the script to upload the data to HBase. In the next few weeks, we will need to consult this team in order to import our output to HBase.

6. The Solr team works in collaboration with the Hadoop team to design the HBase schema. Therefore, we also need help from the Solr team in order to import/export the data to/from HBase.

5. Detailed Design

This section explains a relationship between major elements of the LDA component. LDA can be broken down into 3 main parts: data preparation, topics extraction, and data storing. The relationship between the components can be simply represented in Figure 8.

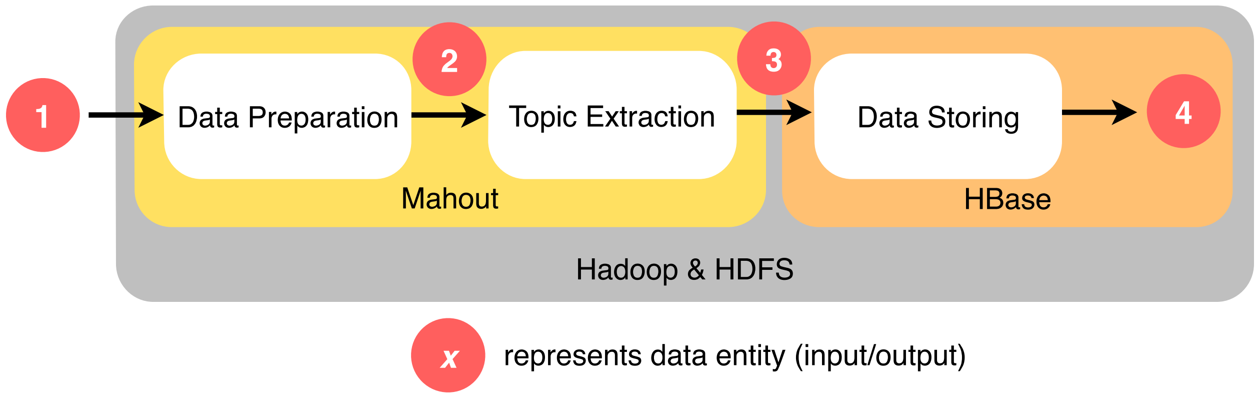


FIGURE 8 ARCHITECTURE OF LDA

The input and output of each part are shown in Table 1.

TABLE 1 INPUT AND OUTPUT OF EACH COMPONENT IN LDA

Data No.	Data Name	Data Type	Location
1	1. Cleaned tweets 2. Cleaned webpages	AVRO	HDFS
2	1. Tweets 2. Webpages	Sequence file	HDFS
3	1. Document-Topic distribution 2. Topic-Word distribution	AVRO	Local Storage
4	1. Document-Topic distribution 2. Topic-Word distribution	See HBase schema	HBase

According to Figure 8 and Table 1, the task of each component is as follow:

- **Data preparation** converts the cleaned tweets and webpages in the AVRO format (data no. 1) to the sequence file (data no. 2). Notice that these files are stored in HDFS.
- **Topic extraction** employs CVB (Collapsed Variational Bayesian) on Mahout to get a collection of topics from the documents. The inputs of this component is the sequence

file. The output from this module are topic and word distributions, which are stored in the AVRO format. The output files are stored in a local storage temporarily.

- **Data storing** reads the output files from the topic extraction module and stores them to HBase. As we talked to the Hadoop team, we will use JSON as our data-exchange format. Therefore, the JSON schema for LDA output as shown as follows:

```
{
     “type”: “array”,
     “items”: [{
          “words”: {“type”: “array of strings”}
          “prob”: {“type”: “number”}
     }]
}
```

JSON schema shown above is associated with only one document. Each item in the array contains the words in the topic and the probability value of the topic.

The JSON data will be stored in the AVRO file. Then, the AVRO file will be uploaded to HBase by using the script provided by the Hadoop team

6. Implementation

This section presents the implementation of the LDA component.

6.1. Input Data Preprocessing

There are two types of inputs in our work, tweets and webpages. The data preprocessing methods for tweets and webpages are not the same. Therefore, this subsection is divided into 2 parts, tweets preprocessing and webpages preprocessing.

6.1.1. Tweets Preprocessing

The input data is stored in the AVRO file format. Since LDA on Mahout only accept the sequence file as the input, we need to convert the AVRO file to the sequence file. We then implement a Hadoop program to read the tweets from AVRO and create the sequence file.

In order to convert the AVRO file to the sequence file, the schema of the AVRO file is required. We thus use the schema file (*tweets.avsc*) of the cleaned tweets provided by the noise reduction team. The schema is shown in List 1.

LIST 1 CLEANED TWEET SCHEMA

```
{
  "namespace": "cs5604.tweet.NoiseReduction",
  "type": “record",
  "name": "TweetNoiseReduction",
  "fields": [
      {"name": "doc_id", "type": "string"},
      {"doc": "original", "name": "tweet_id", "type": “string"},
      {"doc": "original", "name": "text_clean", "type": “string"},
      {"doc": "original", "name": "text_original", "type": "string"},
      {"doc": "original", "name": "created_at",  "type": “string"},
      {"doc": "original", "name": "user_screen_name", "type": “string"},
      {"doc": "original", "name": "user_id", "type": “string"},
      {"doc": "original", "name": "source", "type": ["string", “null"]},
      {“doc": "original", "name": "lang", "type": ["string", “null"]},
      {"doc": "original", "name": "favorite_count", "type": ["int",
  “null"]},
      {"doc": "original", "name": "retweet_count", "type": ["int",
  "null"]},
      {"doc": "original", "name": "contributors_id", "type": ["string",
  "null"]},
      {"doc": "original", "name": "coordinates", "type": ["string",
  "null"]},
      {"doc": "original", "name": "urls", "type": ["string", "null"]},
      {"doc": "original", "name": "hashtags", "type": ["string", "null"]},
      {"doc": "original", "name": "user_mentions_id", "type": ["string",
  "null"]},
      {"doc": "original", "name": "in_reply_to_user_id", "type":
  ["string", "null"]},
      {"doc": "original", "name": "in_reply_to_status_id", "type":
  ["string", "null"]}
  ]
}
```

From the schema, we use only the text_clean field to produce the sequence file. The program to convert the AVRO file to the sequence file is shown in List 4, in Section 8.6. The output sequence file will be used as the input of Mahout LDA.

6.1.2. Webpages Preprocessing

We use the cleaned webpages as the input of LDA. The data is also stored in the AVRO format. In order to process the webpages using Mahout LDA, we need to convert the AVRO file to the sequence file. Thus, we use the schema file (*webpages.avsc*) provided by the noise reduction team to read the AVRO file. The schema of the cleaned webpages is shown in List 2.

LIST 2 CLEANED WEBPAGE SCHEMA

```
  {
    "namespace": “cs5604.webpage.NoiseReduction",
    "type": "record",
    "name": "WebpageNoiseReduction",
    "fields": [
        {"name": "doc_id", "type": "string"},
        {"doc": "original", "name": "text_clean", "type": ["null",
    "string"], "default": null},
        {"doc": "original", "name": "text_original", "type": ["null",
    "string"], "default": null},
        {"doc": "original", "name": "created_at",  "type": ["null",
    "string"], "default": null},
        {"doc": "original", "name": "accessed_at",  "type": ["null",
    "string"], "default": null},
        {"doc": "original", "name": "author", "type": ["null", "string"],
    "default": null},
        {"doc": "original", "name": "subtitle", "type": ["null", "string"],
    "default": null},
        {"doc": "original", "name": "section", "type": ["null", "string"],
    "default": null},
        {"doc": "original", "name": "lang", "type": ["null", "string"],
    "default": null},
        {"doc": "original", "name": "coordinates", "type": ["null",
    "string"], "default": null},
        {"doc": "original", "name": "urls", "type": ["null", "string"],
    "default": null},
        {"doc": "original", "name": "content_type", "type": ["null",
    "string"], "default": null},
        {"doc": "original", "name": "text_clean2", "type": ["null",
    "string"], "default": null},
        {"doc": "original", "name": "collection", "type": ["null",
    "string"], "default": null},
        {"doc": "original", "name": "title", "type": ["null", "string"],
    "default": null},
        {"doc": "original", "name": "url", "type": ["null", "string"],
    "default": null},
        {"doc": "original", "name": "appears_in_tweet_ids", "type": ["null",
    "string"], "default": null},
        {"doc": "original", "name": "domain", "type": ["null", "string"],
    "default": null}
    ]
  }
```

From the schema, we also use only the text_clean field to produce the sequence file. The program to convert the AVRO file to the sequence file is shown in List 1, in Section 8.6. The output sequence file will be used as the input of Mahout LDA as well.

6.2. LDA processing

This section presents the method to invoke Mahout LDA by using a Java program. The method to choose the number of topics for the data collection is also shown in this section.

6.2.1. Invoking LDA from Java

As mentioned above, we use a Java program to invoke LDA on Mahout. The program performs the following steps:

1. Convert the sequence file to a sparse vector based on TF-IDF
2. Decompose the vector to the singular value decomposition vectors (SVD vectors)
3. Run CVB algorithm on the SVD vectors
4. Export the document-topic distribution and the topic-words distribution vectors in the text file format to the local directory
5. Convert the document-topic and the topic-words distributions to the AVRO file

There are 2 products from LDA, the document-topic distribution and the topic-word distribution.

Structure of the document-topic distribution

$$\{\mathtt{topic}_1{:}\mathtt{P}_1(\mathtt{topic}_1),\mathtt{topic}_2{:}\mathtt{P}_1(\mathtt{topic}_2),\ldots,\mathtt{topic}_N{:}\mathtt{P}_1(\mathtt{topic}_N)\}$$

$$\vdots$$

$$\{\mathtt{topic}_1{:}\mathtt{P}_X(\mathtt{topic}_1),\mathtt{topic}_2{:}\mathtt{P}_X(\mathtt{topic}_2),\ldots,\mathtt{topic}_N{:}\mathtt{P}_X(\mathtt{topic}_N)\}$$

Note: topic$_n$ is topic *n*
P$_x$(topic$_n$) is a probability value of topic *n* for document *x*
N is a number of topics

Structure of the topic-word distribution

$$\{\mathtt{word}_{11}{:}\mathtt{P}(\mathtt{word}_{11}),\mathtt{word}_{12}{:}\mathtt{P}(\mathtt{word}_{12}),\ldots,\mathtt{word}_{1M}{:}\mathtt{P}(\mathtt{word}_{1M})\}$$

$$\vdots$$

$$\{\mathtt{word}_{N1}{:}\mathtt{P}(\mathtt{word}_{N1}),\mathtt{word}_{N2}{:}\mathtt{P}(\mathtt{word}_{N2}),\ldots,\mathtt{word}_{NM}{:}\mathtt{P}(\mathtt{word}_{NM})\}$$

Note: word$_{nm}$ is word *m* in topic *n*
P(word$_{mn}$) is a probability value of word *m in topic n*
M is a number of words in each topic
N is a number topics

The following contents are the examples of the document-topic distribution and the topic-word distribution.

The content of the *topic-dist* file

```
{0:0.4653814163484247,0.9:9.909485725912798E-5,00:0.43340406376113655,01:
0.0965042841954036,04:0.004611140837775986}

{0:0.14290405284000532,0.9:1.3330665414191818E-5,00:0.14598699218624975,0
1:0.69797128545587,04:0.013124338852460764}

{0:0.12847616913207754,0.9:0.5656171204828104,00:0.18990642426152174,01:0
.00269070620569358,04:0.11330957991789659}
                          :
```

The content of the *lad-topic* file

```
{election:0.09377751292611668,turnout:0.015083911979021125,kashmir:
0.014100209700970283,early:0.013935142916804111,first:
0.012675928548904244,high:0.011941311072608523,any:
0.010771568993729792,win:0.010105794563938113,other:
0.00958682872066384,must:0.00937156450246399}

{election:0.07178331424599466,you:0.02260734936435454,i:
0.015745045166685488,we:0.013520059835565763,about:
0.011924460627230609,all:0.009788375158349343,so:
0.008545898792866836,can:0.007851899797888513,have:
0.007744246542788778,what:0.007070752282093559}

{election:0.07709982543943172,next:0.016222794019969712,day:
0.013913697409794536,up:0.012617008186560684,amp:0.011917249328227677,i:
0.01147080259035411,have:0.00945404960992083,his:
0.009014498875395748,vote:0.007757042413072532,out:0.006882451375573196}
                                  ⋮
```

According to the example, the first topic consists of 10 words: *election*, *turnout*, *kashmir*, *early*, *first*, *high*, *any*, *win*, *other*, and *must*. The probability of the first topic for the first document is approximately 0.46 (≈ 0.4653814163484247).

6.2.2. Selection of Number of Topics

The important input parameter of LDA is a number of latent topics for each collection. It is challenging to select a suitable number of topics for each collection because there are over million data records for each collection. Since LDA is one of the clustering techniques, the number of topics is directly related to the number of clusters. If the number of topics is too small, the documents that are not likely to be in the same group may be forced to be together. If there are too many topics, strongly related document may be partitioned. In our work, we cluster the documents based on the top *N* topics. The documents that have the same top *N* topics will be grouped in the same cluster.

To determine the number of topics, we apply an exhaustive method to the small dataset and we use a heuristic method on the large dataset. We measure the quality of LDA by using Kullback-Leibler (KL) divergence. It is the value that measures the similarity between documents based on their probability distributions. In our work, KL divergence is calculated on the document-topic distribution matrix. For each cluster, we compute the average of the KL divergence values. Then, we use the mean of the average KL divergence values to express the clustering quality of LDA.

KL divergence can be computed as follows:

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i)\, \ln \frac{P(i)}{Q(i)}.$$

where P and Q are probabilities

If KL divergence is close to zero, documents are similar. Therefore, the more the similarity between documents, the less KL divergence.

6.2.2.1. Empirical Study For A Small Tweet Collection

We run LDA on the dataset and vary the number of topics from 3 to 20. We also vary a number of top topics, from 3 to 5. Our approach searches through every variations and select the setting that gives the best KL divergence.

The results of our empirical study are shown in Figure 9 and Table 2.

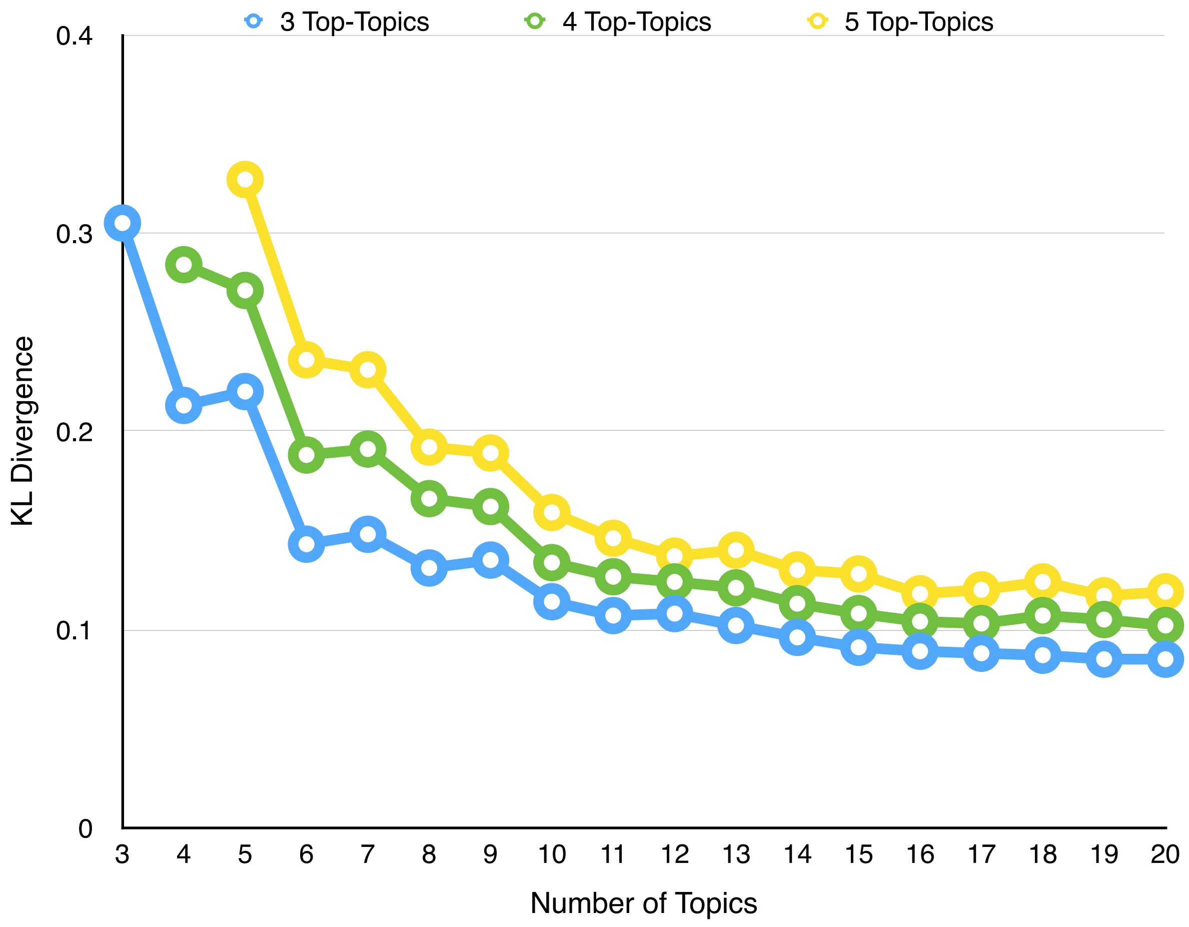


FIGURE 9 EMPIRICAL RESULT OF A SMALL TWEET COLLECTION

TABLE 2 EMPIRICAL RESULT OF A SMALL TWEET COLLECTION

Number of Topics	Number of Top Topics		
	3	4	5
3	0.305	-	-
4	0.213	0.284	-
5	0.22	0.271	0.327
6	0.143	0.188	0.236
7	0.148	0.191	0.231
8	0.131	0.166	0.192
9	0.135	0.162	0.189
10	0.114	0.1337	0.159
11	0.107	0.1266	0.146
12	0.108	0.124	0.137
13	0.102	0.121	0.14
14	0.096	0.113	0.13
15	0.091	0.108	0.128
16	0.089	0.104	0.118
17	0.088	0.103	0.12
18	0.087	0.107	0.124
19	0.085	0.105	0.117
20	0.085	0.102	0.119

According to the results, the KL divergence is decreased when the number of topics and the number of top topics increase. Therefore, the number of top topics that we choose for both small and big collections is 3 because it can give the best KL divergence value when compare to other numbers. Although 20 latent topics can give the best KL divergence value, we choose 15 as the number of topics for the small collection because KL divergence does not improve significantly when the number of topics is over 15.

6.2.2.2. Empirical Study For A Big Tweet Collection

We set the maximum number of topics for the big collections to 50, therefore, the search space is larger. For this reason, searching through every number of topics will take a long time to search for the most appropriate setting. We start searching from 3 topics and increase the number of topics by 3. The result from the empirical study is shown in Figure 10.

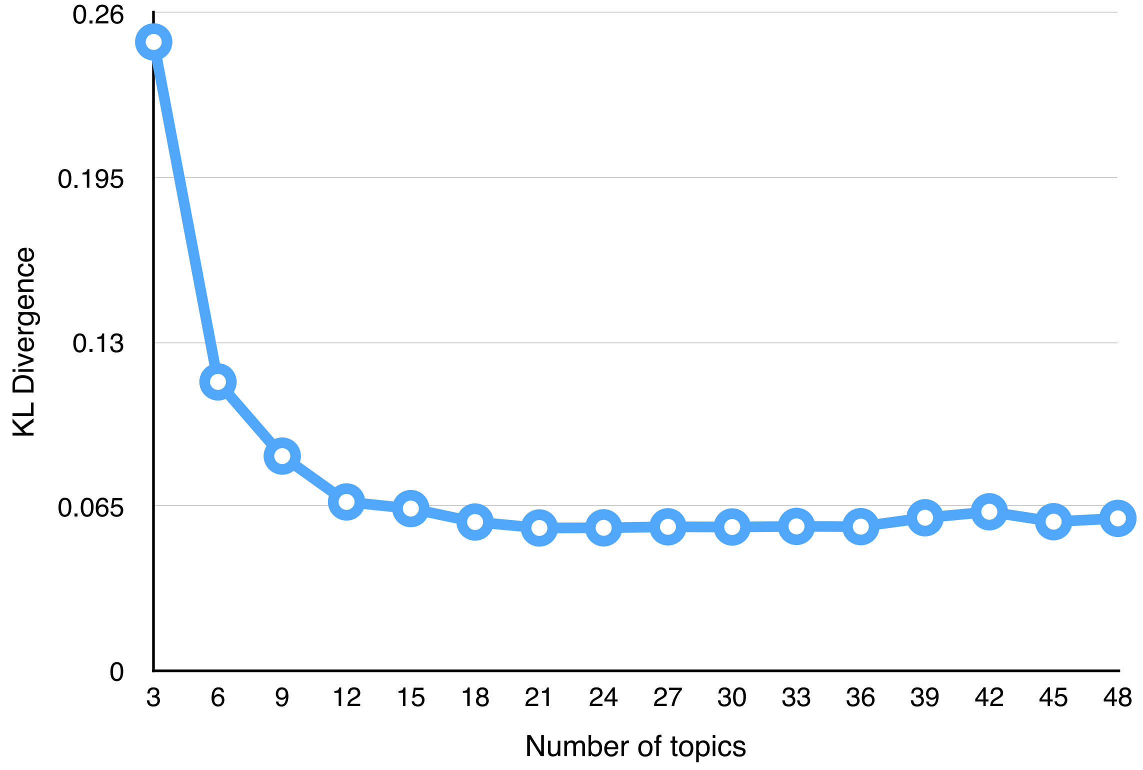


FIGURE 10 EMPIRICAL RESULT OF A BIG TWEETS COLLECTION

From the graph, the KL divergence value does not improve significantly when the number of topics is over 15. Therefore, the number of topics for the large data collection that we choose is 15.

6.2.2.3. Empirical Study For A Small Webpage Collection

The empirical study for the small webpage collection is as same as the one for a small tweet collection. We vary the number of topics from 3 to 20, and we vary the number of top-topics from 3 to 5. The results are shown in Figure 11 and Table 3.

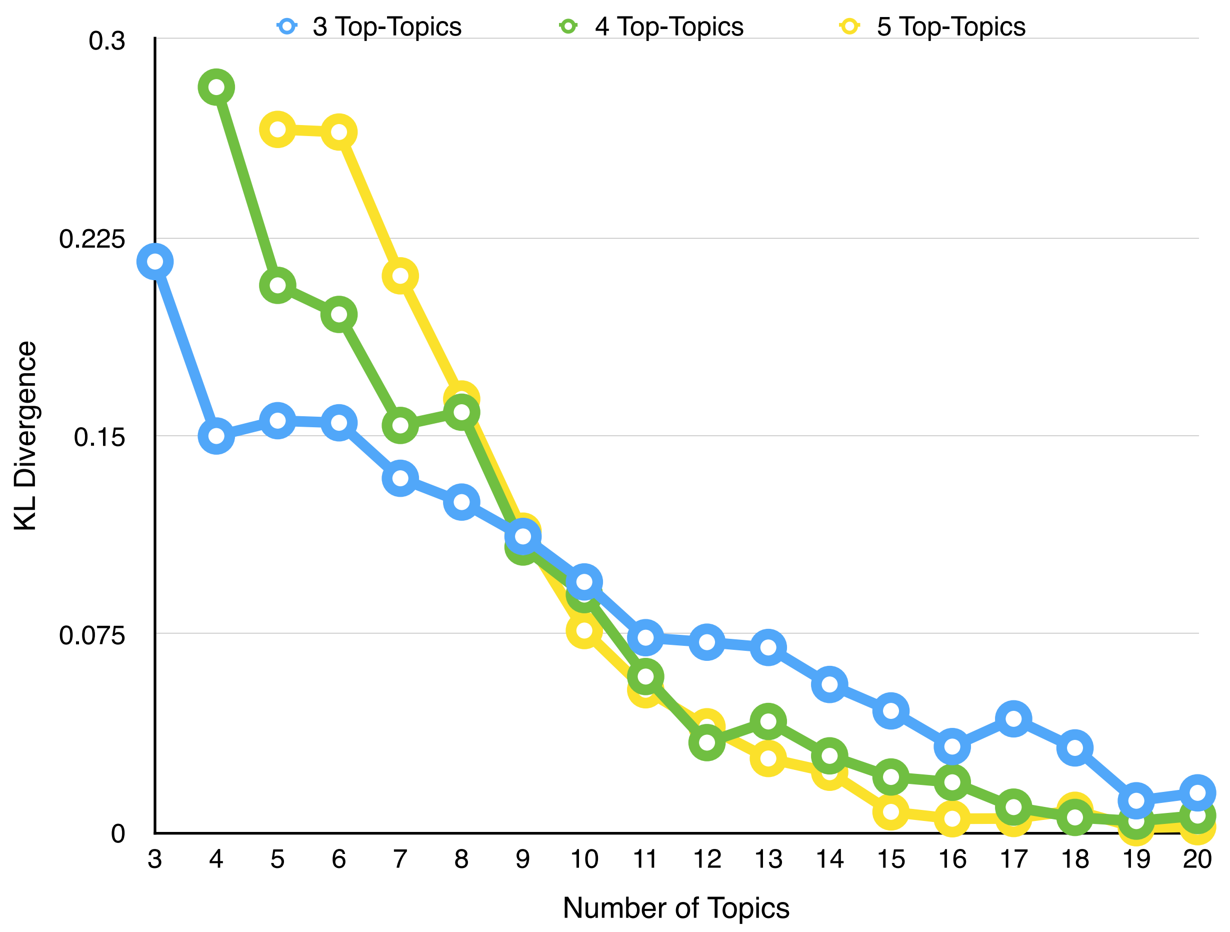


FIGURE 11 EMPIRICAL RESULT OF A SMALL WEBPAGE COLLECTION

From Figure 11, the number of topics that we choose for a small webpage collection is 15 and the number of top-topics is 5 because this setting can give the best KL divergence value.

6.3. Output Data Storing

The output of LDA is stored in the plain text format. In order to upload the output to HBase, we need to convert the text file to the AVRO file. The following is the schema of the AVRO file (*lda.avsc*).

LIST 3 LDA OUTPUT SCHEMA

```
{
  "namespace": "cs5604.tweet.LDA",
  "type": "record",
  "name": "TweetLDA",
  "fields": [
     {"name": "doc_id", "type": "string"},
     {"doc": "analysis", "name": "lda_topics", "type": ["string", "null"]},
     {"doc": "analysis", "name": "lda_vectors", "type": ["string", "null"]}
   ]
}
```

From the schema, the *doc_id* field is the ID of the document. This field is associated with the document ID provided by the noise reduction team.

The *lda_topics* field stores the topic for the document, which is the combination of words of the topic with highest probability. The words are separated by a pipe (|);

```
cat | pet | animal
```

The *lda_vectors* field stores the top-three-topic of the document in the JSON string format. The topics are separated by a pipe (|). The following is the example of the *lda_vectors* data.

```
{"words": ["cat", "pet", "animal"], "prob": 0.5} |
{"words": ["dog", "pet", "animal"], "prob": 0.25} |
{"words": ["dog", "cute", "animal"], "prob": 0.15}
```

Notice that the schema of the JSON string is shown in Section 5.

We write a Java program to convert the output data to the AVRO file. The source code is shown in List 5 in Section 8.7.

To upload the AVRO file to HBase, we use the script provided by the Hadoop team. The name of the script is *hbase-loader.jar* (for more information, please refer to the report of the Hadoop team).

7. LDA Evaluation

7.1. Human Judgement

To evaluate LDA topic model, human judgement [16] is an efficient method. Human judgment is applied by word intrusion task and topic intrusion task. The task of the user is to find the word or topic which is out of place of doesn't belong with the others, i.e., the intruder.

The questions in the questionnaire are divided into two parts. One is to select words that are not closely related with others in the words set of one topic. It is used to evaluate the performance of extracting words for each topic. Another is to decide whether the words in the set can describe the topic well based on subjects' judgement. It is used to evaluate the performance of extracting topics for the collection.

7.1.1. Word Intrusion

Word intrusion is designed to measure the coherence of topics.We give subject a group of randomly ordered words. The task of users is to find the word which doesn't belong with the others, i.e., the intruder. For example, in the set {dog, cat, horse, apple, pig, cow}, the word apple will be identified as intruder because all other words are related to animals.

Using the word intrusion method, we could calculate a value from the results of subjects to estimate LDA model. The formula is shown below[16]:

$$MP_k^m = \sum_s 1(i_{k,s}^m = w_k^m)/S$$

w (m,k) is the intruding word among the words of k th topic generated by m th model. And i(m,k,s) is the intruder selected by subject s on the testing set. S is the number of subjects.

7.1.2. Topic Intrusion

Topic intrusion is used to test whether a topic model's decomposition of document into groups, which agrees with human judgments of the documents. We use the formula shown below to calculate the value TLO[16], which is used to estimate the coherence of human judgment and model results.

$$TLO_d^m = (\sum_s \log \theta_{d,j_{d,*}^m}^m - log \log \theta_{d,j_{d,s}^m}^m)/S$$

The higher the value of TLO, the geater the correspondence between the human judgment and the model. θ is the possibility of the topic in the documents. Index * means it is the true intruder. And with the index s, j donate to the intruded topic selected by subjects.

7.1.3. Evaluation Result

To do the evaluation based on human judgement, we make a questionnaire that can be found in the appendix (Appendix A).

The questions in the questionnaire are divided into two parts. One is to select words that are not closely related with others in the words set of one topic. It is used to evaluate the performance of extracting words for each topic. Another is to decide whether the words in the set can describe the topic well based on subjects' judgement. It is used to evaluate the performance of extracting topics for the collection.

According to the responses collected, we calculated the value MP. It is around 65%. Actually, the performance of LDA is good. The relevance of words within a set between human judgement and results of LDA can be estimated as 65%. In addition, we find a more interesting conclusion from the data. If we choose first two most frequently selected words for each question, we can make a table as following:

TABLE 3 FIRST SELECTED WORDS FOR EACH QUESTION

Question	1	2	3	4	5	6	7	8
1	rt	rt	rt	rt	from	rt	rt	rt
2	says	jet	news	after	mh	pic	mh	pic

It is obvious that the word rt is selected in all questions except 5. Because the word set for question 5 doesn't contain the word rt. RT at the beginning of a Tweet is to indicate that they are re-posting someone else's content. So rt shows in tweets very frequently. According to this, we can make a conclusion that words, which are shown in documents frequently but not related to the content of the documents will have a lot negative effects on the result calculated from LDA.

For the second parts of the questionnaire, we make a chart according to the data collected. The answer Yes means that the set of words calculated from LDA can describe the topic or we extract the right topic. Answer no is just the opposite. The chart is shown as following:

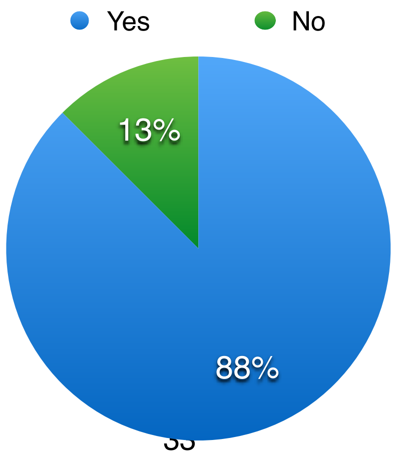


FIGURE 12 THE PROPORTION OF ANSWER YES TO NO

According to the chart, the percentage of getting right topics from LDA is 88%. It means that 88% topics extracted from LDA agree with human judgement.

By comparing the conclusions from these two kinds of questions, we know that the performance of LDA we used on extracting topics is better than the performance on calculating words for each topic. For future work, we will focus on improving the performance of extracting words for each documents using different ways.

7.2. Cross-Validation with the Clustering Team

We compare our clustering results to the clustering team. In this evaluation, the cosine similarity value is used to measure the similarity of the documents within the same cluster. As same as the KL divergence, we expect the cosine similarity to be as close to zero as possible. We decide to do the comparison on the Ebola collection (a small collection) and the Malaysia Airline collection (a big collection).

To compare our results, we will use t-test, which is a statistical method to compare the means of two samples. We expect that the quality of LDA clustering is better than the clustering result of the clustering team, which means our average cosine similarity (μ_{LDA}) is lower than the average cosine similarity of the clustering team ($\mu_{Clustering}$). Therefore, our hypotheses are as follows:

Null hypothesis (H_0): $\mu_{LDA} - \mu_{Clustering} > 0$

Alternative hypothesis (H_1): $\mu_{LDA} - \mu_{Clustering} \leq 0$

TABLE 4 CROSS VALIDATION RESULTS

	Ebola Collection		Malaysia Airlines Collection	
Run	**LDA**	**Clustering**	**LDA**	**Clustering**
1	0.0472	0.4293	0.117	0.601
2	0.0475	0.4240	0.140	0.635
3	0.0486	0.4329	0.144	0.632
4	0.0483	0.4374	0.149	0.629
5	0.0501	0.4344	0.133	0.622
6	0.0475	0.3957	0.151	0.621
7	0.0491	0.4257	0.162	0.618
8	0.0466	0.4653	0.133	0.618
9	0.0472	0.4091	0.149	0.634
10	0.0483	0.4397	0.137	0.597

The significant level (α) in our experiment is set to 0.05 (95% confidence interval). We use Minitab, which is statistical analysis tool, to compute t-test. We can reject the null hypothesis and accept the alternative hypothesis if p-value is less than the significant level.

The result for the Ebola collection is shown as follows:

```
Two-sample T for LDA vs Clustering

             N     Mean    StDev  SE Mean
LDA         10  0.04804  0.00105  0.00033
Clustering  10   0.4294   0.0185   0.0059

Difference = mu (LDA) - mu (Clustering)
Estimate for difference:  -0.38131
95% upper bound for difference:  -0.37056
T-Test of difference = 0 (vs <): T-Value = -65.01  P-Value = 0.000  DF = 9
```

For the Ebola collection, we do not have enough evidence to accept the null hypothesis. Therefore, we accept the alternative hypothesis, which means LDA has better performance than the clustering method because the average cosine similarity of LDA is lower than the average cosine similarity of the clustering approach.

The result for the Malaysia Airlines collection is shown as follows:

```
Two-sample T for LDA vs Clustering

             N    Mean   StDev  SE Mean
```

```
LDA          10  0.1415  0.0125   0.0039
Clustering   10  0.6207  0.0131   0.0041


Difference = mu (LDA) - mu (Clustering)
Estimate for difference:  -0.47920
95% upper bound for difference:  -0.46926
T-Test of difference = 0 (vs <): T-Value = -83.83  P-Value = 0.000  DF = 17
```

For the Malaysia Airlines collection, we do not have a strong evidence to accept the null hypothesis, thus we reject the null hypothesis and accept the alternative hypothesis. Our approach also performs better the clustering method.

From the t-test, LDA can outperform the clustering approach. The goal of the LDA team is to maximize the similarity of the documents within the same group, while the objectives of the clustering team are both maximizing the similarity of the documents inside the same cluster and the dissimilarity of the documents in the different groups. Our measurement is somewhat bias towards LDA because it measures only the similarity of the members in the same cluster. Since we do not have enough time to do more experiments, we would like to suggest the extensive work to use a fair-comparison measurement. However, according to the experimental results, LDA has better performance when maximizing only the similarity of the documents within the same group. We believe that LDA will still have a comparable performance as the clustering method.

8. Topics for All Collections

We apply LDA to all data collections and the topics for each collection are presented in this section. Table 5 shows the summary of all collections.

TABLE 5 COLLECTION SUMMARY

Type	Name	Category	File Size	Number of Tweets
Small Collection	Jan 25	Revolution/Unrest	220.43 MB	301,605
	Charlie Hebdo	Shooting	73.51 MB	173,145
	Ebola	Disease	142.94 MB	380,267
	Election	Uprising/Election	339.24 MB	829,676
	Plan Crash	Accident	104.82 MB	265,999
	Suicide Bomb Attack	Bombing	16.61 MB	37,173
	Winter Storm	Storm	202.27 MB	485,903
Big Collection	Malaysia Airlines	Accident	320.14 MB	775,411
	Bomb	Bombing	6.54 GB	1,776,622
	Diabets	Disease	2.07 GB	1,594,337
	Egypt	Revolution/Unrest	3.31 GB	2,055,157
	Shooting	Shooting	8.40 GB	1,570,190
	Storm	Storm	7.60 GB	1,644,525
	Tunisia	Uprising/Election	1 GB	1,615,104

Notice that the collections used in this section are the cleaned collection from the noise reduction team.

8.1. Jan 25

Type: Small

Topics:

["died","martyr","age","gunshot","28","egypt","2","14","mahmoud"]
["egypt","age","6","martyr","30","died","2","gunshot","20","jan28"]
["martyr","egypt","20","today","died","protest","revolution","egyptian","18"]
["martyr","mohamed","died","egypt","amp","age","gunshot","4","today"]
["died","age","martyr","mohamed","gunshot","revolution","2011128","mahmoud","25"]
["martyr","died","25","egypts","police","28","mohamed","25jan","4"]
["30","3","died","amp","today","28","mohamed","2013","22"]
["martyr","amp","mohamed","age","28","3","egyptian","4","military"]
["died","martyr","gunshot","egypt","25","amp","el","square","2011128"]
["25jan","28","egyptian","egypt","amp","died","6april","3","24"]
["died","age","25","revolution","police","6","gunshot","square","people"]
["3","egypt","died","sites","police","unblock","fast","protesters","2"]
["amp","28","unblock","fast","week","2011128","martyr","next","19"]
["3","revolution","sites","fast","10","25jan","unblock","people","18"]
["martyr","age","today","police","3","29","one","egypt","killed"]

8.2. Charlie Hebdo

Type: Small

Topics:

["charlie","hebdo","fired","cartoon","charliehebdo","see","http","basketball","french"]
["charlie","hebdo","attack","new","muslims","paris","http","people","us","cover"]
["hebdo","charlie","cartoons","paris","via","protest","attacks","france","attack"]
["charlie","attack","french","new","hebdo","http","rally","muslim","respond"]
["hebdo","charlie","attack","cartoons","muslims","speech","protest","people","freedom"]
["charlie","hebdo","via","uk","police","french","cartoons","paris","muslim","massacre"]
["charlie","via","attacks","new","france","police","http","french","muslim"]
["hebdo","police","speech","paris","muslims","http","says","religion","british"]
["hebdo","charlie","police","free","new","http","attacks","attack","france"]
["charlie","cartoons","attack","attacks","hebdo","muslims","says","like","police"]
["hebdo","charlie","attacks","paris","free","speech","cartoons","via","muslim"]
["hebdo","charlie","police","muslims","attack","attacks","french","details","names"]
["charlie","hebdo","paris","police","france","attack","muslims","french","killed"]
["police","charlie","hebdo","french","paris","http","via","free","attacks"]
["charlie","hebdo","attack","speech","free","via","police","cover","http"]

8.3. Ebola

Type: Small

Topics:
["ebola","outbreak","us","patient","leone","sierra","hospital","africa","says"]
["us","ebola","spread","outbreak","get","medical","scientists","virus","fight"]
["ebola","virus","africa","says","amp","west","liberia","news","patient"]
["ebola","leone","us","doctor","sierra","virus","infected","get","case"]
["ebola","liberia","first","doctor","amp","case","new","cases","get"]
["ebola","via","amp","liberia","us","man","news","im","fight"]
["ebola","virus","outbreak","via","health","infected","us","people","world"]
["ebola","us","liberia","sierra","west","nigeria","wash","doctors","vaccine"]
["ebola","wash","better","195","hiv","sins","via","patient","soap"]
["ebola","west","via","liberia","fight","health","outbreak","nigeria","people"]
["ebola","africa","wash","health","us","news","*profanity*","like","cases"]
["ebola","us","virus","says","get","via","cases","west","fight"]
["africa","outbreak","ebola","first","stop","amp","people","spread","us"]
["sierra","leone","africa","liberia","patient","ebola","outbreak","hospital","via"]
["ebola","outbreak","africa","via","amp","health","spread","like","help"]

8.4. Election

Type: Small

Topics:
["election","reelection","general","labour","would","like","dont","people","news"]
["election","general","2015","win","next","presidential","campaign","via","vote"]
["election","bjp","next","presidential","campaign","votes","2016","one","labour"]
["election","next","vote","presidential","amp","http","dont","results","today"]
["election","amp","win","2015","results","party","jonathan","president","snap"]
["election","new","win","2015","party","labour","via","dont","cameron"]
["election","amp","general","vote","time","2014","elections","2015","won"]
["election","new","us","get","vote","party","2016","2015","want","presidential"]
["election","win","via","2015","http","day","want","campaign","next"]
["election","next","general","amp","http","like","2015","via","greek"]
["election","via","http","like","would","get","win","state","polls"]
["via","general","new","reelection","us","campaign","win","presidential","obama"]
["election","vote","2015","day","general","results","party","dont","one"]
["election","amp","day","vote","via","party","wins","new","delhi"]
[“election","general","voting","delhi","presidential","amp","one","2015","people"]

8.5. Plane Crash

Type: Small

Topics:
["crash","plane","killed","everyone","myles","jumps","people","family","bodies"]
["crash","plane","dead","died","near","airport","pilot","today","house","first"]
["crash","family","girl","plane","7yearold","year","dead","7","old"]
["plane","crash","died","taiwan","airasia","found","pilot","dead","engine"]
["plane","crash","airasia","survived","via","girl","died","air","amp"]
["crash","plane","airasia","pilot","flight","died","news","harrison","survivor"]
["plane","crash","via","family","news","pilot","die","flight","video","7yearold"]
["crash","camera","caught","killed","plane","man","ever","survived","managed"]
["plane","crash","dead","killed","die","near","news","bodies","airport"]
["plane","crash","pilot","killed","small","years","airasia","today","caught"]
["crash","plane","killed","survivor","years","old","girl","lost","today"]
["crash","plane","girl","camera","killed","survived","small","caught","family"]
["crash","die","killed","airasia","plane","found","one","flight","new"]
["plane","crash","killed","airasia","girl","via","people","near","pilot"]
["plane","die","ariana","fav","grande","crashes","join","ignore","caught"]

8.6. Suicide Bomb Attack

Type: Small

Topics:
["suicide","attack","car","people","least","bomber","kills","killed","pakistan","baghdad"]
["car","attack","least","killed","bomb","kabul","iraq","afghan","people","suicide"]
["bomb","attack","suicide","dead","killed","pakistan","left","blast","iraq"]
["bomb","attack","people","kills","car","least","afghan","afghanistan","police"]
["attack","suicide","bomb","killed","iraq","church","people","kills","pakistan","funeral"]
["bomb","suicide","attack","kills","killed","iraq","people","afghanistan","kabul"]
["bomb","suicide","killed","dead","kabul","syria","bombing","news","bomber"]
["bomb","attack","car","killed","least","kills","blast","baghdad","amp"]
["bomb","attack","suicide","killed","people","car","kabul","officials","church"]
["bomb","attack","blast","least","rogue","targets","libya","general","car"]
["suicide","bomb","cafe","car","attack","deadly","kills","iraq","police"]
["suicide","attack","killed","pakistan","least","dead","bomb","kabul","church"]
["attack","suicide","bomb","killed","pakistan","least","afghanistan","car","people"]
["suicide","attack","kills","blast","least","bus","afghanistan","afghan","killed"]
["suicide","attack","bomb","kills","car","bomber","afghan","kano","iraq"]

8.7. Winter Storm

Type: Small

Topics:
["storm","warning","winter","pm","weather","another","http","new","nws"]
["winter","storm","snow","weather","issued","amp","inches","cold","east","counties"]
["storm","issued","snow","watch","winter","http","counties","warning","est"]
["storm","winter","snow","warning","watch","weather","northeast","another","amp"]
["storm","amp","warning","new","snow","us","ice","effect","morning","southern"]
["winter","storm","warning","est","amp","weather","november","snow","another","pm"]
["storm","est","winter","issued","watch","february","northeast","warning","january"]
["winter","storm","effect","county","warning","weather","new","snow","watch"]
["winter","snow","storm","warning","issued","watch","area","http","nws"]
["winter","storm","warning","issued","snow","february","nws","new","cst"]
["storm","winter","watch","february","weather","amp","northeast","est","major"]
["winter","storm","warning","snow","est","watch","nws","new","issued"]
["winter","storm","issued","watch","snow","nws","november","warning","east"]
["winter","storm","weather","watch","issued","us","amp","morning","mst"]
["storm","winter","watch","new","northeast","est","due","pm","another"]

8.8. Malaysia Airlines

Type: Big

Topics:
["plane","mh370","malaysia","pic","1","missing","found","shocking","update"]
["airlines","search","flight","missing","mh370","crash","australian","says","plane"]
["malaysia","airlines","missing","mh370","search","jet","flight","new","plane"]
["flight","search","malaysia","missing","jet","mh370","plane","shocking","found"]
["search","plane","found","airlines","1","pic","malaysia","new","underwater"]
["airlines","malaysia","flight","missing","news","plane","ukraine","1","mh370"]
["airlines","malaysia","flight","mh370","370","jet","mh17","missing","new"]
["malaysia","airlines","pic","1","found","plane","mh370","flight","missing"]
["malaysia","airlines","flight","missing","mh370","plane","new","crash","found","370"]
["flight","370","malaysia","news","airlines","shocking","ukraine","search","via"]
["airlines","flight","plane","found","search","pic","370","mh17","mh370"]
["airlines","plane","says","malaysia","found","flight","search","1","new"]
["malaysia","airlines","plane","found","1","mh370","pic","flight","bucket"]
["flight","malaysia","airlines","mh370","370","plane","mh17","search","report"]
["malaysia","plane","search","crash","mh370","370","airlines","jet","underwater"]

8.9. Bomb

Type: Big

Topics:
["bomb","boston","im","time","love","man","bombing","bombings","suspect"]
["bomb","im","amp","u","time","one","*profanity*","like","us"]
["bomb","*profanity*","da","like","im","got","know","make","day","today"]
["bomb","like","boston","one","da","love","dont","car","man","police"]
["bomb","like","*profanity*","lol","amp","dont","u","dot","com"]
["bomb","boston","lol","time","im","right","marathon","da","*profanity*"]
["bomb","amp","go","u","im","sounds","right","make","threat"]
["amp","love","sex","bombing","got","get","police","threat","*profanity*","lol"]
["bomb","*profanity*","boston","amp","new","got","want","like","sounds"]
["bomb","da","like","im","right","dick","amp","people","get"]
["bomb","like","*profanity*","*profanity*","right","da","lol","af","amp"]
["bomb","boston","bombing","suspect","dont","look","lol","marathon","photo"]
["bomb","boston","*profanity*","like","da","go","say","dick","*profanity*"]
["bomb","boston","bombing","marathon","go","amp","da","lol","suspect"]
["bomb","*profanity*","sounds","boston","time","u","us","man","made"]

8.10. Diabetes

Type: Big

Topics:
["2","type","help","disease","dont","know","health","new","de"]
["diabetes","risk","im","know","chocolate","dont","heart","bill","sweet"]
["diabetes","get","new","risk","people","1","amp","juan","eats"]
["like","know","2","people","risk","get","bars","one","de"]
["sugar","bars","type","one","juan","2","study","may","disease"]
["diabetes","john","type","2","new","day","get","today","w"]
["diabetes","amp","disease","type","blood","help","sugar","risk","im","chocolate"]
["diabetes","get","via","new","disease","sugar","one","diet","study"]
["type","amp","help","2","diabetes","1","people","via","chocolate"]
["diabetes","amp","chocolate","know","im","jim","de","la","type"]
["diabetes","2","risk","type","1","de","people","could","weight"]
["diabetes","type","2","risk","like","day","one","dont","amp"]
["diabetes","type","juan","bars","2","1","eats","chocolate","get"]
["amp","2","cancer","diabetes","heart","weight","type","via","help","like"]
["diabetes","bars","type","jim","45","health","people","cancer","eats"]

8.11. Egypt

Type: Big

Topics:
["new","today","http","iran","25jan","political","free","5","two"]
["2011","via","news","state","people","2","5","president","one"]
["25jan","us","2","amp","political","news","new","protest","egyptian"]
["egypt","us","military","amp","6","president","killed","people","protest"]
["brotherhood","muslim","says","free","human","cairo","http","military","killed"]
["3","amp","egypt","2011","egyptian","via","http","4","killed"]
["police","http","killed","support","one","watch","people","day","muslim"]
["egypt","25jan","via","us","coup","6","muslim","new","protesters"]
["president","people","egypt","http","morsi","army","al","news","military"]
["says","new","2","via","people","egypt","time","amp","iran","4"]
["egypt","3","2011","amp","coup","military","live","army","egyptian"]
["amp","egyptian","via","new","2","brotherhood","us","protest","police"]
["2011","today","amp","via","says","25jan","military","day","egyptian"]
["amp","via","http","police","people","4","us","cairo","6"]
["amp","egyptian","military","muslim","brotherhood","security","protesters","people","army"]

8.12. Shooting

Type: Big

Topics:
["shooting","police","school","people","high","video","shot","3","two"]
["shooting","like","star","people","police","saw","3","im","today"]
["shooting","school","day","police","game","time","dont","going","suspect"]
["school","amp","police","one","gun","like","star","niggas","3"]
["shooting","school","time","video","today","day","man","suspect","first","people"]
["shooting","gun","star","police","amp","go","today","new","like"]
["shooting","video","im","like","one","day","amp","2","get"]
["shooting","amp","star","new","day","u","like","today","one"]
["im","police","today","shooting","dont","dead","video","time","last"]
["shooting","amp","night","stars","today","video","people","3","day"]
["shooting","like","stars","video","man","amp","first","police","im"]
["shooting","today","video","like","new","music","people","good","star"]
["shooting","star","stars","one","killed","2","man","school","dont"]
["shooting","im","video","new","stars","star","amp","time","man"]
["shooting","im","man","go","night","one","dont","new","see","like"]

8.13. Storm

Type: Big

Topics:
["storm","thunder","im","snow","coming","got","get","winter","us"]
["storm","like","us","amp","nothing","rain","new","im","together","life"]
["storm","tropical","go","perfect","im","right","ill","via","night","http"]
["storm","snow","like","calm","us","new","get","winter","right"]
["storm","every","rain","like","runs","calm","away","snow","big"]
["storm","rain","get","winter","amp","day","night","calm","snow"]
["storm","us","winter","amp","one","via","night","weather","tropical"]
["storm","im","new","day","weather","get","last","cant","would"]
["like","snow","day","cant","weather","tornado","one","night","tweet"]
["storm","snow","rain","dance","amp","life","pass","learning","waiting"]
["storm","know","dont","lol","severe","perfect","every","day","hope"]
["storm","like","rain","winter","dont","warning","get","go","cant"]
["storm","weather","snow","amp","know","dont","im","rain","today"]
["storm","like","winter","coming","calm","today","ice","god","snow"]
[“storm","im","every","love","life","waiting","going","snow","see"]

8.14. Tunisia

Type: Big

Topics:
["tsunami","one","wave","distance","japan","100","day","miles","lol"]
["tsunami","like","amp","warning","m","via","hit","lol","new"]
["tsunami","get","greater","undersea","like","earthquake","dont","devastating","lol"]
["tsunami","japan","got","de","today","like","separated","would","make"]
["m","warning","amp","like","im","california","2014","earthquake","gmt"]
["tsunami","like","earthquake","japan","via","clouds","coming","old","one"]
["tsunami","amp","di","warning","aceh","quake","video","japan","lol","tahun"]
["tsunami","greater","power","whack","warning","creation","devastating","led","get"]
["tsunami","new","get","dont","5","like","quake","tides","di"]
["tsunami","eyes","tides","waves","like","theyre","anymore","missing","dont"]
["tsunami","get","like","dont","tides","missing","anymore","waves","eyes","theyre"]
["tsunami","im","like","get","japan","tortoise","family","make","dont","friend"]
["tsunami","got","amp","best","japan","baby","friend","year","hippo"]
["tsunami","baby","year","new","tortoise","old","hippo","friend","like"]
[“tsunami”,"like","warning","di","japan","im","m","get","alaska"]

8.15. Conclusion

Based on the results presented above, we can see that LDA works well for most time. And topics extracted from documents are coherent with human judgement. However, there are some problem we need to deal with. One is that there are sometimes some unrelated words in a word set of a topic. These words are frequency words that are not related to a topic. We need to find ways to get rid of the influence of this kind of words. Another is to find a reliable method to find the best number of topics in a documents and number of words to present a topic.

9. User Manual

This section demonstrates how to use HDFS (Hadoop Distributed File System) and Mahout's LDA.

9.1. Basic HDFS Usage

HDFS is a part of Hadoop. It is a distributed file system used for distributing files across machines in a Hadoop cluster. HDFS (for Hadoop version 1.2.1) can be invoked using the command `hadoop fs -<options> <arguments>`

- Create a new directory

```
    hadoop fs -mkdir <HDFS directory name>
```

- List files in a directory

```
    hadoop fs -ls <HDFS directory name>
```

- Remove a file in a directory

```
    hadoop fs -rm <HDFS file name>
```

- Remove a directory

```
    hadoop fs -rmr <HDFS directory name>
```

- Upload files to a directory

```
    hadoop fs -put <file in local storage> <path of HDFS directory>/
```

9.2. Compile and Run the Hadoop Program to Convert AVRO/Text File to Sequence File/Text File

The program that is presented in this section can read an AVRO file or a text file and convert the file to a sequence or a text file. The source code of the program is shown in List 4.

LIST 4 THE PROGRAM FOR CONVERTING CSV FILE TO SEQUENCE FILE

```
//Java libs
import java.io.File;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.Scanner;
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;

//Hadoop libs
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.BytesWritable;
import org.apache.hadoop.io.IOUtils;
import org.apache.hadoop.io.SequenceFile;
import org.apache.hadoop.io.SequenceFile.Reader.Option;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.util.ReflectionUtils;


//AVRO libs
import org.apache.avro.Schema;
import org.apache.avro.io.DatumReader;
import org.apache.avro.generic.GenericRecord;
import org.apache.avro.generic.GenericDatumReader;
import org.apache.avro.file.*;
import org.apache.avro.mapred.FsInput;

public class SequenceFileOperator {
  private Configuration conf = new Configuration();

  public static void main(String[] args) throws IOException {
  //Validate input parameters
  if (args == null || args.length < 4) {
     System.out.println("Usage: hadoop jar seq.jar   SequenceFileOperator
        <arguments>");
     System.out.println("\t<arg1> is an absolute path of the AVRO
        file or the text file\n\t"
     + "<arg2> is an absolute path of the AVRO's schema file\n\t"
     + "<arg3> is a HDFS path of the sequence file or a local
        directory path of the txt file\n\t"
     + "<arg4> is an option of input file [0 is an AVRO file,
        1 is a txt file]\n\t"
```

```
      + "<arg5> is an option of output file [0 is a sequence file,
         1 is a txt file]");

      return;
   }

   SequenceFileOperator docToSeqFileWriter = new SequenceFileOperator();
   String avroPath = args[0];
   String avroSchema = args[1];
   String filePath = args[2];
   int optionIn = Integer.parseInt(args[3]);
   int optionOut = Integer.parseInt(args[4]);

   docToSeqFileWriter.loadDocument(avroPath, avroSchema,
      filePath, optionIn, optionOut);
   }

   private void loadDocuments(String inputPath, String avroSchema,
                              String outPath, int optionIn,
                              int optionOut) throws IOException {
   Schema schema = null;
   File file = null;
   DatumReader<GenericRecord> datumReader = null;
   DataFileReader<GenericRecord> dataFileReader = null;
   GenericRecord data = null;
   Scanner scanner = null;

   //the input is an AVRO file
   if(optionIn == 0){
      System.out.println("Input AVRO file: " + inputPath);
      schema = new Schema.Parser().parse(new File(avroSchema));
         file = new File(inputPath);
         datumReader = new GenericDatumReader<GenericRecord>(schema);
         dataFileReader = new DataFileReader<GenericRecord>
         (file, datumReader);
         data = null;
   }
   //the input is a text file
   else if(optionIn == 1){
      System.out.println("Input TXT file: " + inputPath);
      scanner = new Scanner(new File(inputPath));
   }

   //the output is a sequence file
   if(optionOut == 0){
      System.out.println("Start creating the sequence file");
      org.apache.hadoop.io.SequenceFile.Writer.Option filePath
         = SequenceFile.Writer.file(new Path(outPath));
      org.apache.hadoop.io.SequenceFile.Writer.Option keyClass
         = SequenceFile.Writer.keyClass(Text.class);
      org.apache.hadoop.io.SequenceFile.Writer.Option valueClass
         = SequenceFile.Writer.valueClass(Text.class);

      SequenceFile.Writer sequenceFileWriter
         = SequenceFile.createWriter(conf, filePath,
         keyClass, valueClass);

      try {
         int count = 0;
         if(optionIn == 0){
            while (dataFileReader.hasNext()) {
               data = dataFileReader.next(data);
               String content = data.get("text_clean").toString();
                     sequenceFileWriter.append(
                  new Text("doc_" + (++count)),
                  new Text(content.replace("\n"," “))
               );
            }
         }
         else{
            while(scanner.hasNextLine()){
               String content = scanner.nextLine();
               sequenceFileWriter.append(
                  new Text("doc_" + (++count)),
                  new Text(content.replace("\n"," “))
               );
            }
         }
      } finally {
         IOUtils.closeStream(sequenceFileWriter);
      }
   }

   //the output is a text file
   else if(optionOut == 1){
      System.out.println("Start creating the text file");
      try {
         BufferedWriter txtOutput
            = new BufferedWriter(new FileWriter(new File(outPath)));
         if(optionIn == 0){
            int i = 0;
            while (dataFileReader.hasNext()) {
               data = dataFileReader.next(data);
               String content = data.get("text_clean").toString();
               txtOutput.write(content.replace("\n", " ") + "\n");
               i++;
            }
         }
         txtOutput.close();
      }
      catch (IOException e){
      }
   }
   }
 }
```

To compile and run the program, follow the steps shown below:

1. Add the JAR files of Cloudera's Hadoop and Hadoop Annotations to the class path. Cloudera

```
export     CLASSPATH=/usr/lib/hadoop/hadoop-common-2.5.0-cdh5.3.0.jar:/
home/gpB/java/hadoop-annotations-2.0.0-cdh4.0.1.jar:/home/gpB/java/
avro-1.7.7.jar:/home/gpB/java/avro-tools-1.7.7.jar:/home/gpB/java/
avro-mapred-1.7.7-hadoop1.jar:/home/gpB/java/hadoop-streaming-2.0.0-
mr1-cdh4.3.0.jar
```

   The class path must contain the following packages:

   1. Hadoop Common (*hadoop-common-2.5.0-cdh5.3.0.jar*): the path of Hadoop can be found by running the command `hadoop version`

   2. Hadoop Annotations (*hadoop-annotations-2.0.0-cdh4.0.1.jar*): it can be downloaded from https://repository.cloudera.com/content/groups/public/org/apache/hadoop/hadoop-annotations/2.0.0-cdh4.0.1/hadoop-annotations-2.0.0-cdh4.0.1.jar

   3. AVRO (*avro-1.7.7.jar*): it can be downloaded from http://mirror.metrocast.net/apache/avro/stable/java/avro-1.7.7.jar

   4. AVRO Tools (*avro-tools-1.7.7.jar*): it can be downloaded from http://mirror.sdunix.com/apache/avro/stable/java/avro-tools-1.7.7.jar

   5. AVRO MapReduce (*avro-mapred-1.7.7-hadoop1.jar*): it can be downloaded from http://mirror.metrocast.net/apache/avro/stable/java/avro-mapred-1.7.7-hadoop1.jar

6. Compile the Hadoop code. This step will create the class file.

```
    javac SequenceFileOperator.java
```

7. Create a JAR file using the class file. This will create the file *seq.jar*.

```
    jar cvf seq.jar SequenceFileOperator.class
```

8. Run the program by passing 5 arguments:
    1. An absolute path (local file system path not a HDFS path) of the input file. This path can be either a path of an AVRO or a text file
    2. An absolute path of an AVRO schema file (in the case that the input file is AVRO)
    3. An output file path. If the output is a sequence file, the output path is a path on HDFS. If the output is a text file, the output is a path on the local file system.
    4. An input file option (either 0 or 1)
        - 0 - AVRO file
        - 1 - text file
    5. An output file option (either 0 or 1)
        - 0 - Sequence file
        - 1 - text file

```
    hadoop jar seq.jar SequenceFileOperator <Arguments>
```

9.3. Compile and Run the Hadoop Program to Invoke LDA

This program run LDA on the input sequence file, dump output vectors to text files, and convert the output tex files to AVRO files. Notice that the AVRO files are stored in the local file system.The source code of the program is shown in List 5.

LIST 5 THE PROGRAM FOR LDA INVOCATION ON MAHOUT

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapred.jobcontrol.*;
import org.apache.hadoop.util.ToolRunner;
import org.apache.mahout.clustering.lda.cvb.CVB0Driver;
import org.apache.mahout.common.AbstractJob;
import org.apache.mahout.common.HadoopUtil;
import org.apache.mahout.text.SequenceFilesFromDirectory;
import org.apache.mahout.utils.vectors.RowIdJob;
import org.apache.mahout.utils.vectors.VectorDumper;
import org.apache.mahout.vectorizer.SparseVectorsFromSequenceFiles;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.util.*;
import java.io.File;
import java.io.FileNotFoundException;

//AVRO libs
import org.apache.avro.io.*;
import org.apache.avro.Schema;
import org.apache.avro.Schema.Parser;
import org.apache.avro.generic.GenericRecord;
import org.apache.avro.generic.GenericDatumWriter;
import org.apache.avro.generic.GenericDatumReader;
import org.apache.avro.generic.GenericData;
import org.apache.avro.file.*;
import org.apache.avro.mapred.FsInput;

public class LDAJob extends AbstractJob {
  private static final Logger log = LoggerFactory.getLogger(Job.class);
  static int numTopics = 5;
  static int numTopTopics = 3;
  static int numWords = 10;
  static double doc_topic_smoothening = 0.0001;
  static double term_topic_smoothening = 0.0001;
  static int maxIter = 1;
  static int iteration_block_size = 10;
  static double convergenceDelta = 0;
  static float testFraction = 0.0f;
  static int numTrainThreads = 4;
  static int numUpdateThreads = 1;
  static int maxItersPerDoc = 1;
  static int numReduceTasks = 10;
  static boolean backfillPerplexity = false;
  static String subFix = "";
  static ArrayList<String> topics = new ArrayList<String>();
  static ArrayList<String> topWords = new ArrayList<String>();
  static String avroSchemaFile = "lda.avsc";
  static String avroOutFile = "lda.avro";
  static String avroInFile = "";
  static String avroSchemaInFile = "";
  static String hdfsFolder = "lda-optimized";

  public static void main(String[] args) throws Exception {

   if(args.length >= 4){
     Configuration conf = new Configuration();
     HadoopUtil.delete(conf, output);
```

```
      //set arguments
      numTopics = Integer.parseInt(args[0]);
      maxIter = Integer.parseInt(args[1]);
      subFix = args[2];
      hdfsFolder = args[3];

      if(args.length > 4){
         avroSchemaFile = args[4];
         avroOutFile = "avro_" + subFix + ".avro";
         avroInFile = args[5];
         avroSchemaInFile = args[6];
      }

      String[] ldaArgs = { "-DbaseFileLocation=" + baseFileLocation };

      ToolRunner.run(new LDAJob(), ldaArgs);
      System.out.println("done");
   }
   else{
      System.out.println("Usage\n\thadoop jar lda.jar LDAJob "
         + "<number of topics> <number of iterations> " +
         + "<output subfix> <HDFS working directory> " +
         + "[<schema file: lda.avsc> <avro input file> " +
         + "<avro schema input file>]");
   }
   }

   public int run(String[] arg0) throws Exception {

   Configuration conf = getConf();
   String folder = hdfsFolder;
   String ldaSeq = folder + "/lda-seq";
   String ldaVectors = folder + "/lda-vectors";
   String ldaMatrix = folder + "/lda-matrix";
   String ldaOut = folder + "/lda-out";
   String ldaTopicOut = folder + "/lda-topic-out";
   String topicWordDist = "topic-word-" + subFix  + ".dist";
   String docTopicDist = "doc-topic-" + subFix + ".dist";

   log.info("Deleting all the previous files.");
   HadoopUtil.delete(conf, new Path("temp"));
   HadoopUtil.delete(conf, new Path(ldaVectors));
   HadoopUtil.delete(conf, new Path(ldaMatrix));
   HadoopUtil.delete(conf, new Path(ldaOut));
```

```
    HadoopUtil.delete(conf, new Path(ldaTopicOut));

    log.info("Step 2: converting the seq to vector.");
    System.out.println("starting seq To Vector job");
    String[] seqToVectorArgs = {
       "-i", ldaSeq, “-o", ldaVectors, "-wt", "tf", “--namedVector"
    };
    ToolRunner.run(new SparseVectorsFromSequenceFiles(), seqToVectorArgs);
    System.out.println("finished seq to vector job");

    log.info("Step3:   convert   SequenceFile<Text,   VectorWritable>   to
    SequenceFile<IntWritable, VectorWritable>");
    System.out.println("starting rowID job");
    String[] rowIdArgs = {
       "-i", ldaVectors + "/tf-vectors", "-o", ldaMatrix
    };
    ToolRunner.run(new RowIdJob(), rowIdArgs);
    System.out.println("finished rowID job");

    log.info("Step4: Run the LDA algo");
    System.out.println("starting caluclulating the number of terms");
    System.out.println("finished calculating the number of terms");
    long seed = System.nanoTime() % 10000;
    System.out.println("starting the CVB job");
    CVB0Driver drive = new CVB0Driver();
    String[] cvbArgs = {
          "-i", ldaMatrix + “/matrix”, "-o", ldaOut,
          "-k", numTopics + "", "-x", "1", "-dt", ldaTopicOut,
          "-dict", ldaVectors + "/dictionary.file-0"
    };
    drive.run(cvbArgs);
    System.out.println("finished the cvb job");

    log.info("Step5: vectordump topic-term");
    System.out.println("starting the vector dumper for topic term");

    //topic-word distribution
    String[] topicTermDumperArg = {
       "-i", ldaOut,  "-d", ldaVectors + "/dictionary.file-0",
       "-o", topicWordDist, "-dt", "sequencefile", "-sort",
       ldaOut, "-vs", numWords + ""
    };

    VectorDumper.main(topicTermDumperArg);
    System.out.println("finisher the vector dumper for topicterm");
```

```java
System.out.println("starting the vector dumper for doc topic");
//doc-topic distribution
String[] docTopicDumperArg = {
   "-i", ldaTopicOut + "/part-m-00000",
   "-d", ldaVectors + "/dictionary.file-0",
   "-dt", "sequencefile", "-o", docTopicDist
};
VectorDumper.main(docTopicDumperArg);
System.out.println("finish the vector dumper for doctopic dumper");

log.info("Step6: convert data to AVRO file");
System.out.println("convert data to AVRO file");
Scanner topicWordScanner = new Scanner(new File(topicWordDist));

System.out.println("++++++ List of topics ++++++");
while(topicWordScanner.hasNextLine()){
   String line = topicWordScanner.nextLine();
   line  = line.substring(1, line.length() - 1);
   String words = "\"words\": [";
   String[] temp = line.split(",");
   int i = 0;
   int j = 0;
   for(String t: temp){
      String word = t.split(":")[0];
      if(!word.toLowerCase().equals("rt")){
         words += "\"" + word + "\"";
         if(i == 0){
            topWords.add(word);
         }
         words += ",";
         i++;
      }
      if(j == (numWords - 1)){
         words = words.substring(0, words.length() - 1);
      }
      j++;
   }
   words += "]";
   System.out.println(words);
   topics.add(words);
}
topicWordScanner.close();
```

```
    //params for write AVRO
    Schema schema = new Parser().parse(new File(avroSchemaFile));
    File file = new File(avroOutFile);
    DatumWriter<GenericRecord>           datumWriter           =           new
    GenericDatumWriter<GenericRecord>(schema);
    DataFileWriter<GenericRecord>         dataFileWriter         =         new
    DataFileWriter<GenericRecord>(datumWriter);
    dataFileWriter.create(schema, file);

    //params for read AVRO
    Schema schemaIn = new Schema.Parser().parse(new File(avroSchemaInFile));
    File fileIn = new File(avroInFile);
    DatumReader<GenericRecord> datumReader
       = new GenericDatumReader<GenericRecord>(schemaIn);
    DataFileReader<GenericRecord> dataFileReader
       = new DataFileReader<GenericRecord>(fileIn, datumReader);
    GenericRecord dataIn = null;

    Scanner docTopicScanner = new Scanner(new File(docTopicDist));
    while(docTopicScanner.hasNextLine()){
       String line = docTopicScanner.nextLine();
       line = line.substring(1, line.length() - 1);
       String[] temp = line.split(",");
       int i = 0;
       ArrayList<Prob> probs = new ArrayList<Prob>();
       for(String t: temp){
          String[] top = t.split(":");
          if(top.length == 2){
             Prob p = new Prob(top[0], Double.parseDouble(top[1]));
             p.setId(i);
             probs.add(p);
             i++;
          }
       }
       Collections.sort(probs);
       String ldaVectorString = "";
       String ldaTopicString = "";
       int id = 0;
       for(int j = 0; j < numTopTopics; j++){
          id = probs.get(j).id;
          ldaVectorString += "{" + topics.get(id) + ", \"prob\": "
             + probs.get(j).prob + "}";
          ldaTopicString += topWords.get(id);
          if(j < numTopTopics - 1){
             ldaVectorString += " | ";
```

```
              ldaTopicString += " | ";
          }
       }

       dataIn = dataFileReader.next(dataIn);

       //create new record
       GenericRecord record = new GenericData.Record(schema);
       record.put("doc_id", dataIn.get("doc_id").toString());
       record.put("lda_topics", ldaTopicString);
       record.put("lda_vectors", ldaVectorString);
       dataFileWriter.append(record);
    }
    dataFileWriter.close();

    return 0;
  }
  }
```

Notice that the code is modified from [15].

To compile and run the program, follow the steps shown below:

1. Add the necessary JAR files to the class path.

```
export  CLASSPATH=home/gpB/java/hadoop-annotations-2.0.0-cdh4.0.1.jar:/
usr/lib/mahout/mahout-examples-0.9-cdh5.3.0-job.jar:/home/gpB/java/
slf4j-1.7.12/slf4j-log4j12-1.7.12.jar:/home/gpB/java/slf4j-1.7.12/
slf4j-api-1.7.12.jar:/usr/lib/hadoop/lib/*:/usr/lib/hadoop/.//*:/usr/
lib/hadoop-hdfs/./:/usr/lib/hadoop-hdfs/lib/*:/usr/lib/hadoop-hdfs/.//
*:/usr/lib/hadoop-yarn/lib/*:/usr/lib/hadoop-yarn/.//*:/usr/lib/
hadoop-mapreduce/lib/*:/usr/lib/hadoop-mapreduce/.//*The   class   path
must contain the following packages:
```

1. Hadoop Annotations (*hadoop-annotations-2.0.0-cdh4.0.1.jar*): it can be downloaded from https://repository.cloudera.com/content/groups/public/org/apache/hadoop/hadoop-annotations/2.0.0-cdh4.0.1/hadoop-annotations-2.0.0-cdh4.0.1.jar

2. Mahout (*mahout-examples-0.9-cdh5.3.0-job.jar*): it is a Mahout for Cloudera

3. Logging Tools (*slf4j-log4j12-1.7.12.jar*): it can be downloaded from http://www.slf4j.org/dist/slf4j-1.7.12.tar.gz

    4. Hadoop class path: it can be obtained by using the command `hadoop classpath`

5. Compile the Hadoop code. This step will create the class file.

```
javac LDAJob.java
```

6. Create a JAR file using the class file. This will create the file *lda.jar*.

```
jar cvf lda.jar LDAJob.class
```

7. Before running the program, set *HADOOP_CLASSPATH* using the command `export HADOOP_CLASSPATH=$CLASSPATH`. Run the program by passing the following arguments:
    1. Number of topics
    2. Number of iterations
    3. A suffix of the output file
    4. A name of HDFS working directory: this folder is required to be created before the execution and it has to contain the input sequence file, namely "*lda-seq*"
    5. An AVRO schema file for LDA (see List 3 for more information)
    6. An input in AVRO format (the program gets the document IDs from this file)
    7. An AVRO schema file for the AVRO in 6.

```
hadoop jar lda.jar LDAJob <Arguments>
```

9.4. Read AVRO File

Since we are curious about what is contained in the AVRO file, we use the tool from the AVRO project (https://avro.apache.org) to read the file.

1. Download the following files from Cloudera and the AVRO project
    1. *hadoop-streaming-2.0.0-mr1-cdh4.3.0.jar* (https://repository.cloudera.com/artifactory/cloudera-repos/org/apache/hadoop/hadoop-streaming/2.0.0-mr1-cdh4.3.0/hadoop-streaming-2.0.0-mr1-cdh4.3.0.jar)
    2. *avro-1.7.7.jar* (http://mirror.metrocast.net/apache/avro/stable/java/avro-1.7.7.jar)
    3. *avro-mapred-1.7.7-hadoop1.jar* (http://mirror.metrocast.net/apache/avro/stable/java/avro-mapred-1.7.7-hadoop1.jar)

    Store in files in the working directory.

2. Launch the following command to read the input AVRO file. The program will convert the binary content in the AVRO file to the JSON string and store it in the output file on HDFS

```
    hadoop jar hadoop-streaming-2.0.0-mr1-cdh4.3.0.jar \
        -D mapred.job.name="avro-streaming" \
        -D mapred.reduce.tasks=0 \
        -files avro-1.7.7.jar,avro-mapred-1.7.7-hadoop1.jar \
        -libjars avro-1.7.7.jar,avro-mapred-1.7.7-hadoop1.jar \
        -input <HDFS path of the AVRO file> \
        -output <HDFS path of the output file> \
        -mapper org.apache.hadoop.mapred.lib.IdentityMapper \
        -inputformat org.apache.avro.mapred.AvroAsTextInputFormat
```

3. Read the output by using the following command

```
    hadoop fs -cat <HDFS path of the output file>/part-00000
```

10. Developers manual

This section provides the details on how to install software and packages that are necessary for this project. 10.1-10.2 is the tutorial for installation of Solr. Since we need to implement LDA on Solr, it is necessary for us to installing Solr at the beginning of the project. Tutorial of crawling webpages using Python script is in 10.3, which is a important technique to get webpage collections. 10.4 is talking about installation of hadoop, on which we need to implement LDA. 10.5 is the instruction for Mahout installation. We use Mahout LDA to extract topics from bother tweet and webpage collections. Finally, another package, Nutch, is discussed about, and an example for extracting webpages based on URL from tweet collection is given, which is in 10.6 and 10.7. The order for introducing packages are based on time we first use. So we think it is most helpful to present in this order. Notice that an operating system used in the following examples is GNU/Linux #1 SMP Debian 3.16.5-1. The name of the host computer is *micloud2*.

10.1.Solr Installation

This section presents the details on how to install Solr.

1. Solr requires Java 1.7 or later version. So before installing Solr, checking the version of Java is necessary. Run the `java -version` command in a terminal. It will show the content as follows:

```
java version "1.7.0_65"
OpenJDK Runtime Environment (IcedTea 2.5.3) (7u71-2.5.3-2)
OpenJDK 64-Bit Server VM (build 24.65-b04, mixed mode)
```

   If the version of Java is earlier than 1.7, the later version of JDK (Java SE Development Kit) can be downloaded from the Oracle website: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

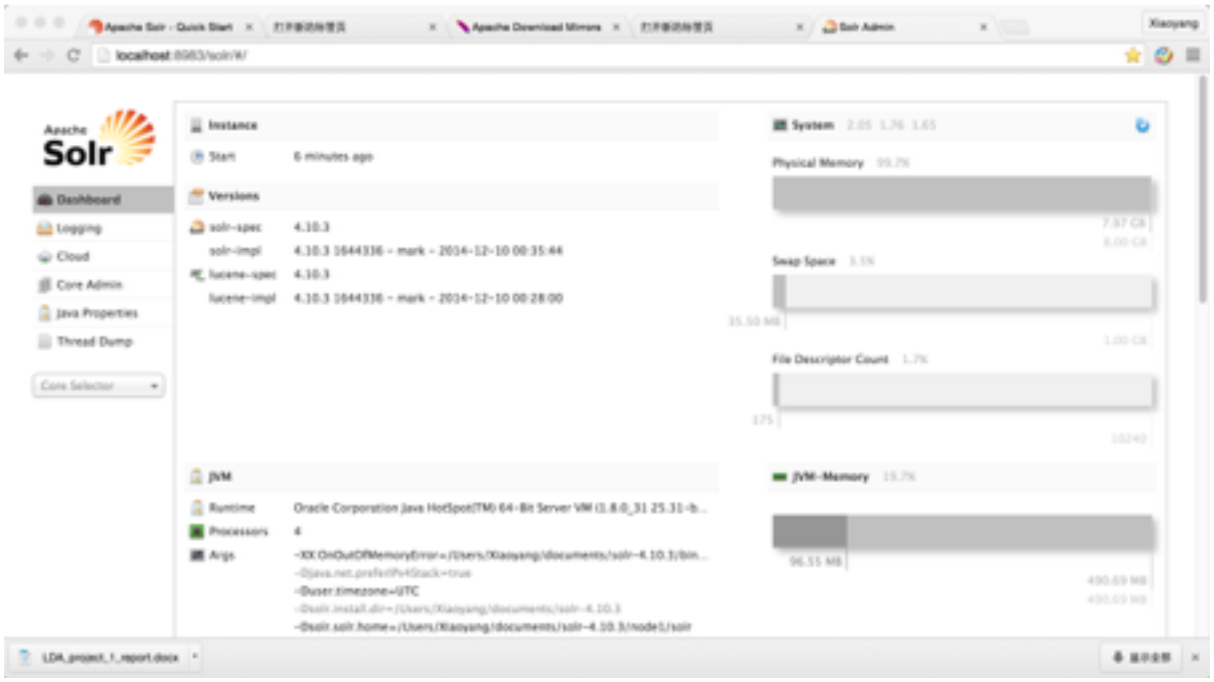
2. The latest version Solr-4.10.3 is available at the Apache website: http://www.apache.org/dyn/closer.cgi/lucene/solr/4.10.3.

3. After downloading the Solr package, unzip it and put the files in the 'Documents' directory.

4. Launch Solr by running `bin/solr start` in terminal. The following message will be shown:

```
Waiting to see Solr listening on port 8983 [\]
Started Solr server on port 8983 (pid=13952). Happy searching!
```

5. Type this address in the browser: http://localhost:8983/solr/ to ensure that Solr is properly installed. If Solr is successfully installed, the Solr Admin user interface (as shown in Figure 13) will be presented.



Instance
Start 6 minutes ago

Versions
solr-spec 4.10.3
solr-impl 4.10.3 1644336 - mark - 2014-12-10 00:35:44
lucene-spec 4.10.3
lucene-impl 4.10.3 1644336 - mark - 2014-12-10 00:28:00

JVM
Runtime Oracle Corporation Java HotSpot(TM) 64-Bit Server VM (1.8.0_31 25.31-b...
Processors 4
Args -XX:OnOutOfMemoryError=/Users/Xiaoyang/documents/solr-4.10.3/bin...
-Djava.net.preferIPv4Stack=true
-Duser.timezone=UTC
-Dsolr.install.dir=/Users/Xiaoyang/documents/solr-4.10.3
-Dsolr.solr.home=/Users/Xiaoyang/documents/solr-4.10.3/node1/solr

System 2.05 1.76 1.65
Physical Memory 99.7% 7.97 GB 8.00 GB
Swap Space 3.5% 35.50 MB 1.00 GB
File Descriptor Count 1.7% 175 10240
JVM-Memory 19.7% 96.55 MB 490.69 MB 490.69 MB

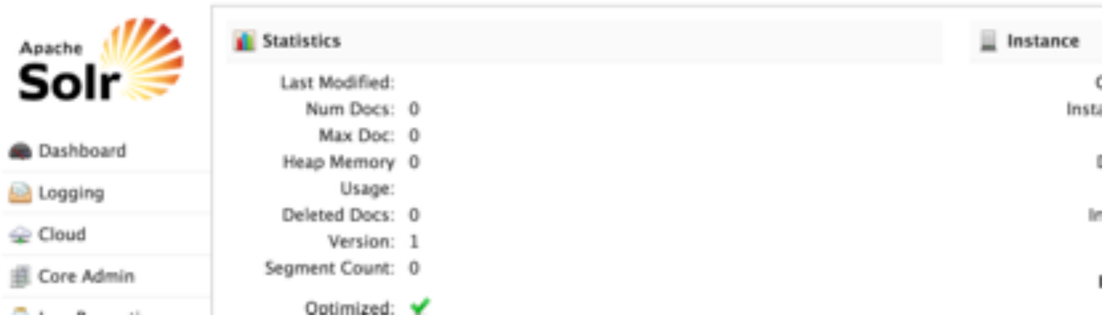
FIGURE 13 SOLR ADMIN SCREEN

10.2. Importing Example Documents to Solr

This section describes how to import documents including tweets and webpages to Solr. We use SimplePostTool to import the data set to Solr.

1. Prepare SimplePostTool by running `export CLASSPATH=dist/solr-core-4.10.2.jar` in terminal to run SimplePostTool.

2. Download *paris_shooting-First200.csv* and *Webpages.zip* from VT Scholar. Before indexing, the screen shot of Solr is shown in Figure 14.



Statistics
Last Modified:
Num Docs: 0
Max Doc: 0
Heap Memory Usage: 0
Deleted Docs: 0
Version: 1
Segment Count: 0
Optimized: ✓

FIGURE 14 SOLR ADMIN SCREEN BEFORE INDEXING

It shows the number of documents is 0.

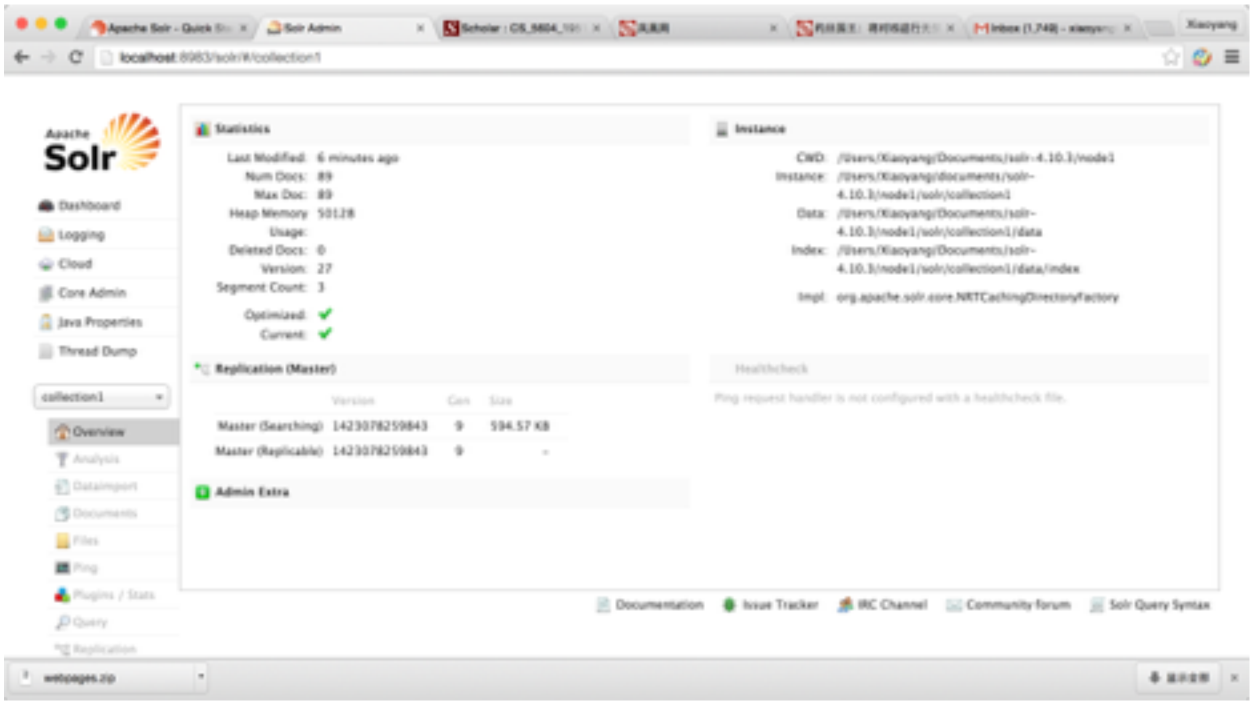
3. Use the command line `java org.apache.solr.util.SimplePostTool example/exampledocs/*.xml` to import all the XML files from *example/exampledocs* into Solr. The terminal shows the following message:

```
SimplePostTool version 1.5
Posting files to base url http://localhost:8983/solr/update using
content-type application/xml..
POSTing file gb18030-example.xml
…
POSTing file sd500.xml
POSTing file solr.xml
POSTing file utf8-example.xml
POSTing file vidcard.xml
14 files indexed.
```

4. Import the webpages downloaded from VT Scholar using the command line `java -Dauto org.apache.solr.util.SimplePostTool ~/Desktop/Webpages`. The terminal shows the following message:

```
Indexing directory /Users/Xiaoyang/Desktop/Webpages (47 files,
depth=0)
POSTing file 0.txt (text/plain)
…
…
POSTing file 9.txt (text/plain)
47 files indexed.
```

Before indexing tweets which are in the CSV format, we index *book.csv* in *example/exampledocs*. The Solr screenshot is shown in Figure 15.



Statistics

```
   Last Modified: 6 minutes ago
       Num Docs: 89
        Max Doc: 89
    Heap Memory: 50128
          Usage:
   Deleted Docs: 0
        Version: 27
  Segment Count: 3
      Optimized: ✓
        Current: ✓
```

Replication (Master)

	Version	Gen	Size
Master (Searching)	1423078259843	9	594.57 KB
Master (Replicable)	1423078259843	9	-

Admin Extra

Instance

```
      CWD: /Users/Xiaoyang/Documents/solr-4.10.3/node1
 Instance: /Users/Xiaoyang/documents/solr-
           4.10.3/node1/solr/collection1
     Data: /Users/Xiaoyang/Documents/solr-
           4.10.3/node1/solr/collection1/data
    Index: /Users/Xiaoyang/Documents/solr-
           4.10.3/node1/solr/collection1/data/index

     Impl: org.apache.solr.core.NRTCachingDirectoryFactory
```

Healthcheck

Ping request handler is not configured with a healthcheck file.

FIGURE 15 SOLR ADMIN SCREEN AFTER INDEXING (WEBPAGES)

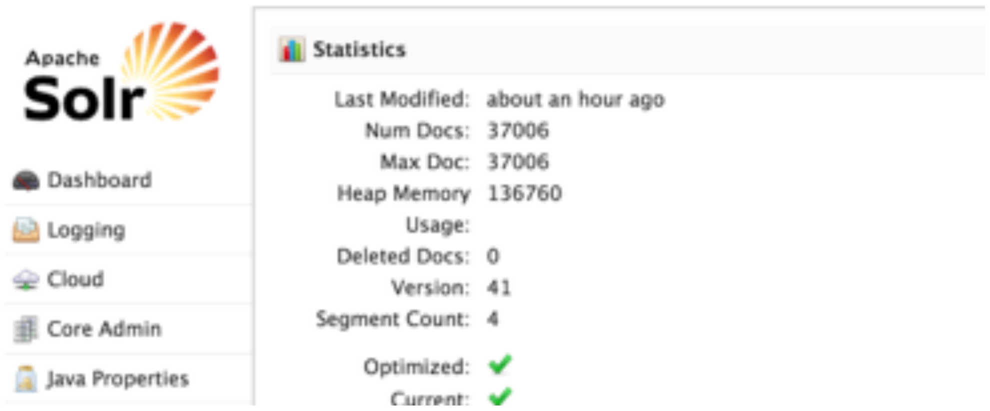
The number of documents is 89.

5. Use the command line to import the tweets in the CSV format (*paris_shooting-First200.csv*). However, the operation fails. We then open the file '*paris_shooting-First200.csv*'. We find double quotations, which we highlight in the content.

```
"twitter-search","RT @IngridSunyer: Rise up against violence! #CharlieHebdo http://t.co/quamNmHAwd","","kidninjaltd","552943243197812736","2346787985","en","<a href=\"http://twitter.com\" rel=\"nofollow\">Twitter Web
```

For the SimplePostTool, the double quotations are used to separate fields. The highlighted ones are included in the content of the fields. The SimplePostTool cannot distinguish the difference between the double quotes in the content and the doubles quote that is used for separating fields. So the way to solve this problem is to replace the double quotes by single quotes. The new tweets file uploaded by Mohamed (the TA), which is *z_542_qp.csv*.

6. Import the file '*z_542_qp.csv*'. The screenshot of Solr admin page after successfully importing the tweets is shown in Figure 16.



Last Modified:	about an hour ago
Num Docs:	37006
Max Doc:	37006
Heap Memory Usage:	136760
Deleted Docs:	0
Version:	41
Segment Count:	4
Optimized:	✓
Current:	✓

FIGURE 16 SOLR ADMIN SCREEN AFTER INDEXING (TWEETS)

Now, the number of documents is 37006.

10.3. Crawling Web Contents using Python Script

1. The Python Script, which is from Mohamed, can be downloaded from VT Scholar. Before using the script, we need to download a couple of packages. The script's name is *tweet_URL_archivingFile.py*.

2. Install Python on a computer. Python is available at the website: https://www.python.org/downloads/. We use Python version 3.4.3, which is the latest version. It is easy to be installed by following the installation manual provided on the website.

3. *Request* and *Beautifulsoup4* modules are required to run the script. We use *Pip* to install these two modules. *Pip* can be downloaded from the website: https://pypi.python.org/pypi/pip. Follow the steps provided on the website to install *Pip*.

4. Launch the command `pip install requests & $pip install beautifulsoup4` to install the modules

5. Make a new directory to host the web files. Enter the directory and use the command `python tweet_URL_archivingFile.py z428t2.csv` to get web contents from URLs.

6. The script will run on the computer for a couple of minutes to produce crawling results. In the terminal window, we will see the output as following:

```
tweets is read from File
short Urls extracted:  223776
cleaned short URLs:  223655
Unique short URLs:  206574
Freq short URLs (>10):  257User Manual
```

```
Unique Orig URLs expanded:  179
Bad URLs:  12
Webpages text saved
```

In the output directory, we get 179 text files (using the default setting).

10.4. Hadoop Installation

This section demonstrates how to install Hadoop on a cluster. The cluster used in this example consists of 4 physical machines (1 master and 4 slaves). All machines are identical. In our example, *micloud2* is the master while *micloud3*, *micloud4*, and *micloud5* are the slaves.

10.4.1.SSH Access Set Up

In order to allow the nodes in the cluster to communicate with each other without a password, the public SSH key connection is used.

1. Open a terminal on the **master** computer and open the directory *~/.ssh*

2. Use the command `ssh-keygen -t rsa` to create a key pair (private key and public key). The command will prompt the following message:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/skx/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/skx/.ssh/id_rsa.
Your public key has been saved in /home/skx/.ssh/id_rsa.pub.
```

   The files *id_rsa* (private key) and *id_rsa.pub* (public key) will be generated in the *~/.ssh* directory

3. Use the command `ssh-copy-id -i ~/.ssh/id_rsa.pub <slave name>` to copy the key to the slaves. The example is shown below:

```
kwang@micloud2:~/.ssh$ ssh-copy-id -i ~/.ssh/id_rsa.pub micloud3
```

4. Check the set up by using the command `ssh  <slave  name>`. If the public SSH key connection is set up properly, the master will be able to access the slave machine. The example is shown below:

```
kwang@micloud2:~/.ssh$ ssh-copy-id -i ~/.ssh/id_rsa.pub micloud3
```

10.4.2.Hadoop Installation

Follow the steps shown below to install Hadoop on the cluster:

1. Download Hadoop version 1.2.1 from the Apache website: http://mirror.nexcess.net/apache/hadoop/common/hadoop-1.2.1/hadoop-1.2.1-bin.tar.gz

2. Extract the file in a preferred directory on the **master** computer (in this case, the master node is *micloud2* and the working directory is */home/kwang*) using the command `tar -zxvf hadoop-1.2.1-bin.tar.gz`

3. Enter the Hadoop configuration directory to modify configuration files. For Hadoop 1.2.1, the configuration directory is *hadoop-1.2.1/conf*.

   In the following configuration examples, modify the content *highlighted in yellow* based on your cluster's configuration.

   - Replace `micloud2` with your master computer's name (host name or IP address)
   - Replace `kwang` with your Hadoop's user name

   3.1. Edit the file '*core-site.xml*' using the following XML:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <property>
       <name>fs.default.name</name>
       <value>hdfs://micloud2:9020</value>
    </property>
    <property>
       <name>hadoop.proxyuser.kwang.hosts</name>
       <value>*</value>
       </property>
    <property>
       <name>hadoop.proxyuser.kwang.groups</name>
       <value>*</value>
    </property>
    <property>
       <name>hadoop.tmp.dir</name>
       <value>/tmp/hadoop-kwang</value>
       <description>A  base  for  other  temporary  directories.</
description>
     </property>
   </configuration>
```

   3.2. Edit the file '*hdfs-site.xml*' using the following XML:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
```

```
<configuration>
    <property>
       <name>dfs.replication</name>
       <value>1</value>
    </property>
</configuration>
```

3.3. Edit the file '*mapred-site.xml*' using the following XML:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <property>
       <name>mapred.job.tracker</name>
       <value>micloud2:9070</value>
    </property>
</configuration>
```

3.4. Edit the file 'master'. This file contains a name or IP address of the master node. In this case, it is *micloud2*.

```
micloud2
```

3.5. Edit the file '*slaves*'. This file contains the list of slave nodes. In our example, the slave nodes are *micloud3*, *micloud4*, and *micloud5*. The content of the *slaves* file is shown as follow:

```
micloud3
micloud4
micloud5
```

3.6. Append the following line to the file 'hadoop-env.sh'.

```
export JAVA_HOME=${JAVA_HOME}
```

4. Set a Hadoop environment variable (*HADOOP_INSTALL*) in the file *~/.bash_profile*. Append the following setting to the file:

```
export HADOOP_INSTALL=/home/kwang/hadoop-1.2.1
```

Notice that, in our example, Hadoop is installed in the directory */home/kwang*.

5. Format Hadoop's name node using the command `$HADOOP_INSTALL/bin/hadoop namenode –format`

6. Start Hadoop using the command `$HADOOP_INSTALL/bin/start-all.sh.`

7. Use the command `jps` to check whether Hadoop have been launched properly

   The following is the output result from the master node (*micloud2*)

```
15294 Jps
14843 NameNode
15017 SecondaryNameNode
15136 JobTracker
```

   The following is the output result from the slave node (*micloud3, micloud4,* and *micloud5*)

```
10349 TaskTracker
10453 Jps
10233 DataNode
```

10.5. Mahout Installation

Mahout is required in our project since it provides the LDA function, which is based on CVB (Collapsed Variational Bayes) algorithm. To install Mahout, make sure that you have installed Maven 3 (mvn) and Subversion (svn) on your master machine.

1. Open a directory that you prefer to install Mahout. Our working directory in this example is */home/kwang*
2. User Subversion to check out Mahout's source code using the command `svn co http://svn.apache.org/repos/asf/mahout/trunk mahout`
3. Go to *mahout/trunk* directory and build Mahout by using the command `mvn install`
4. Once the code is successfully built, the message 'BUILD SUCCESS' will be shown in the terminal

10.6. Nutch Installation and Tutorial

Nutch is an open source web crawler software project. It will be used in collaboration with Python scripts to get webpages from URLs. There are several ways to install Nutch, and we setup Nutch from a binary distribution. A couple of steps are needed to get it ready.

1. Download source package and unzip it. The version we use is 1.9.
2. Verify Nutch installation. Under the directory of nutch, run command `bin/nutch` we get the following output:

```
Usage: nutch COMMAND
where COMMAND is one of:
  readdb              read / dump crawl db
……
  plugin              load a plugin and run one of its classes main()
  CLASSNAME           run the class named CLASSNAME
```

It shows we install Nutch successfully.

After installation, the tutorial can be found from this websites: http://wiki.apache.org/nutch/NutchTutorial#Introduction . And we use the bash script to crawl webpages the command is like the following: `Usage: bin/crawl <seedDir> <crawlDir> <solrURL> <numberOfRounds>`

or `$ apache-nutch-1.9/bin/crawl seedsURLs.txt crawlTest/ http://localhost:8983/solr 2` for us under the Nutch directory

10.7. Crawl Webpages from Small Collection of Tweets

1. Clean up tweets collection z106t.csv by using Python script provided by noise reduce team. Before cleaning, the tweets are like following:

```
   "@ABC: 3 American service members killed this morning in Afghanistan
suicide attack: http://t.co/BTdK38uiUL" 😔 Rest In Peace.
yvette_nicoleee  359688930569879554   Tue Jul 23 14:58:18 +0000 2013
   "@ABC: 3 American service members killed this morning in Afghanistan
suicide attack: http://t.co/KWOeSSUlOc" rest in peace ❤❤❤ 😇😇😇
Ken_Z_ie  359686196970012672  Tue Jul 23 14:47:26 +0000 2013
```

After cleaning, the tweets will become

```
   3 americans die in suicide bomb attack time we pulled all soldiers
out of all war zones an leave the evil *profanity* to it
#toomanyheroesdead  rowbsapril84 359854157059985408  Wed  Jul  24
01:54:51 +0000 2013
   25 People Killed 29 Injured in Suicide Bomb Attack in Northern Iraq
Terrorist Attack ubAlert http://t.co/7q4TylLYR0    mike1029
359819653209534464   Tue Jul 23 23:37:45 +0000 2013
   25 People Killed 29 Injured in Suicide Bomb Attack in Northern Iraq
#terrorist #ubAlert http://t.co/1YV2B1JJQK mike1029 359819613145530368
      Tue Jul 23 23:37:35 +0000 2013
```

2. Execute Python script tweet_shortToLongURL_file to get a list of URLs by using command `python tweet_shortToLongURL_File.py z106t.csv` the script will output a file(seedsURLs_z106t_Cleaned.txt) with a list of URLs

3. modify crawl script of Nutch (with the help from noise reduce team). There are two files needed to be modified. One is the file bin/crawl, and another is conf/nutch-site.xml.

```
   if false
then
#'note'that'the'link'inversion'Q'indexing'routine'can'be'done'within't
he'main'loop #'on'a'per'segment'basis
……
   ''if'['$?'Qne'0'] '''then'exit'$? ''fi
fi
   done
```

Adding the comments highlighted in red.

```
   <property>
         <name>http.agent.name</name>
         <value>cs5604s15_lda</value>
          <description>HTTP 'User-Agent' request header. MUST NOT be
empty -
          please set this to a single word uniquely related to your
organization.
         </description>
       </property>

       <property>
         <name>http.robots.agents</name>
         <value>cs5604s15_lda</value>
          <description>HTTP 'User-Agent' request header. MUST NOT be
empty -
          please set this to a single word uniquely related to your
organization.
         </description>
       </property>
```

4. Execute Nutch by using command : `bin/crawl ../urls/ output/ http://preston.dlib.vt.edu:8980/solr/1 1` and the webpages can be found in directory: ../output/segments/20150411200422/content/part-00000

11. Conclusion

Based on the background knowledge and design, we have implemented LDA in the whole project. We extracted topics from both tweet and webpage collections. Results are presented in Chapter 8 of this report. Based on this results, we use several methods to do evaluations. Base on the evaluation, we improve LDA performance using several ways. We also cooperate with other teams, such as hadoop team, Solr team, to make LDA run in the whole project. We are certain that performance of the information retrieval system of the IDEAL project would be enhanced pleasurably.

In addition, we have introduced tutorials of related packages, which will be helpful for others who want to do similar projects. We also completed a program to evoke LDA from Java and provided the source code. Readers could use the program to apply LDA more easily. A user manual and a developer manual are also present based on our experience to give technique details of our work.

12. Acknowledgements

We are really glad because we manage to complete the task of LDA project. We would like to express our sincere gratitude to Dr. Fox for guidance and encouragement and also for teaching. We also sincerely thank our TAs, Sunshin and Mohamed, for supporting and help. Last but not least, we wish to express appreciation to our classmates for the fruitful co-operation. Without their help, we cannot complete the project, and we cannot have such a fantastic experience.

13. References

[1] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. JMLR 3:993–1022. 418, 520, 525, 528

[2] Wei, Xing, and W. Bruce Croft. 2006. LDA-based document models for ad-hoc retrieval. In Proc. SIGIR, pp. 178–185. ACM Press. DOI: doi.acm.org/10.1145/1148170.1148204. 418, 522, 532

[3] Solr Manual, (Retrieved: Feb 26, 2015) http://lucene.apache.org/solr/quickstart.html

[4] Anthes, Gary. "Topic models vs. unstructured data." Communications of the ACM 53.12 (2010): 16-18.

[5] Lucene, (Retrieved: Feb 26, 2015) http://lucene.apache.org/solr/quickstart.html

[6] Bishop, Christopher M. Pattern recognition and machine learning. Vol. 4. No. 4. New York: springer, 2006.

[7] Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. Vol. 1. Cambridge: Cambridge university press, 2008.

[8] Wikipedia, (Retrieved: Feb 26, 2015) http://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

[9] Introduction to Latent Dirichlet Allocation, (Retrieved: Feb 26, 2015) http://blog.echen.me/2011/08/22/introduction-to-latent-dirichlet-allocation/

[10] Blei, David M; Ng, Andrew Y.; Jordan, Michael. Latent Dirichlet allocation. Journal of Machine Learning Research. 2—3-01, 3(4-5): pp.993-1022.

[11] Ding, Weicong, Mohammad H. Rohban, Prakash Ishwar, and Venkatesh Saligrama. Topic discovery through data dependent and random projections. arXiv preprint arXiv:1303.3664 (2013). http://arxiv.org/pdf/1303.3664.pdf

[12] Pons-Porrata, Aurora, Rafael Berlanga-Llavori, and José Ruiz-Shulcloper. "Topic discovery based on text mining techniques." Information Processing & Management 43, no. 3 (2007): 752-768. http://www.sciencedirect.com/science/article/pii/S0306457306000914

[13] Barry de Ville, Gurpreet S. Bawa. Topic Discovery, Tracking, and Characterization of Social Media Conversation for Point of Origin and Dissemination Discovery: Structural Precursors to Topic Determination in Text Corpora. SAS Institute Inc., Cary, NC. SAS Global Forum, Orlando, Florida, April 2012. http://support.sas.com/resources/papers/proceedings12/304-2012.pdf

[14] Hauffa, Jan, Tobias Lichtenberg, and Georg Groh. "Towards an NLP-Based Topic Characterization of Social Relations." In 2012 International Conference on Social Informatics (SocialInformatics), pp. 289-294. IEEE, 2012. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6542453

[15] How to Print Mahout LDA CVB topic, (Retrieved: April 20, 2015) http://stackoverflow.com/questions/16994529/how-to-print-mahout-lda-cvb-topic

[16] Chang, Jonathan, et al. "Reading tea leaves: How humans interpret topic models." Advances in neural information processing systems. 2009.

14. Appendix

14.1. Appendix A: Questionnaire

1. Please select the words which are not related to others according to the information you get from the set of words showing below

   malaysia,flight,mh370,search,370,missing,airlines,rt,found,says

   - malaysia
   - flight
   - mh370
   - search
   - 370
   - missing
   - airlines
   - rt
   - found
   - says

2. Please select the words which are not related to others according to the information you get from the set of words showing below

   airlines,mh370,missing,plane,rt,flight,jet,,370,malaysia

   - airlines
   - mh370
   - missing
   - plane
   - rt
   - flight
   - jet
   - 370
   - malaysia

3. Please select the words which are not related to others according to the information you get from the set of words showing below

   rt,airlines,crash,new,from,flight,after,ukraine,site,news

   - rt
   - airlines
   - crash
   - new
   - from
   - after
   - ukraine
   - site
   - news

4. Please select the words which are not related to others according to the information you get from the set of words showing below

   airlines,rt,search,malaysia,flight,mh370,after,australia,area,indian

   - airlines
   - rt
   - search
   - malaysia
   - flight
   - mh370
   - after
   - australia
   - area
   - indian

5. Please select the words which are not related to others according to the information you get from the set of words showing below

   malaysia,airlines,flight,jet,mh,plane,search,from,missing,crash

   - malaysia
   - airlines
   - flight
   - jet
   - mh
   - plane
   - search
   - from
   - missing
   - crash

6. Please select the words which are not related to others according to the information you get from the set of words showing below

   rt,airlines,pic,found,plane,missing,update,jet,mh370

   - rt
   - airlines
   - pic
   - found
   - plane
   - missing
   - update
   - jet
   - mh370

7. Please select the words which are not related to others according to the information you get from the set of words showing below

   malaysia,flight,plane,search,missing,mh,rt,370,jet,airlines

   - malaysia
   - flight
   - plane
   - search
   - missing
   - mh
   - rt
   - 370
   - jet
   - airlines

8. *Please select the words which are not related to others according to the information you get from the set of words showing below*

   plane,mh370,flight,missing,found,rt,airlines,pic,shocking

   - *plane*
   - *mh370*
   - *flight*
   - *missing*
   - *found*
   - *rt*
   - *airlines*
   - *pic*
   - *shocking*

9. Do you think the following set of words can describe the Malaysia airlines plane crash event?

   malaysia,flight,370,jet,mh,crash,from,airlines,over,plane

   - Yes
   - No

10. Do you think the following set of words can describe the Malaysia airlines plane crash event?

    malaysia,rt,missing,mh370,plane,370,airlines,flight,australia,about

    - Yes
    - No

11. Do you think the following set of words can describe the Malaysia airlines plane crash event?

    malaysia370,airlines,flight,search,mh370,plane,rt,370,found,pic

    - Yes
    - No

12. Do you think the following set of words can describe the Malaysia airlines plane crash event?

    malaysia,rt,airlines,plane,found,pic,update,crash,shocking

    - Yes
    - No

13. Do you think the following set of words can describe the Malaysia airlines plane crash event?

    airlines,malaysia,plane,flight,mh,pic,search,missing,mh370,new

    - Yes
    - No

14. Do you think the following set of words can describe the Malaysia airlines plane crash event?

    malaysia,airlines,plane,rt,search,flight,mh370,370,found,from

    - Yes
    - No

15. Do you think the following set of words can describe the Malaysia airlines plane crash event?

    rt,mh370,malaysia,370,new,airlines,have,area,search,we

    - Yes
    - No

16. Do you think the following set of words can describe the Malaysia airlines plane crash event?

    malaysia,rt,airlines,mh,search,plane,crash,flight,missing,about

    - Yes
    - No