TOPIC MODELING: CLUSTERING OF DEEP WEBPAGES

Muhunthaadithya C[1], Rohit J.V[2], Sadhana Kesavan[3] and
Dr. E. Sivasankar[4]

[1, 2, 3]Department of CSE, NIT Trichy
4Assistant Professor, Department of CSE, NIT Trichy-620015,
Tamilnadu, India
muhunth93@gmail.com
jvrohit1201@gmail.com
sadhanakesavan@gmail.com

ABSTRACT

The internet is comprised of massive amount of information in the form of zillions of web pages.This information can be categorized into the surface web and the deep web. The existing search engines can effectively make use of surface web information.But the deep web remains unexploited yet. Machine learning techniques have been commonly employed to access deep web content.

Under Machine Learning, topic models provide a simple way to analyze large volumes of unlabeled text. A "topic" consists of a cluster of words that frequently occur together. Using contextual clues, topic models can connect words with similar meanings and distinguish between words with multiple meanings. Clustering is one of the key solutions to organize the deep web databases.In this paper, we cluster deep web databases based on the relevance found among deep web forms by employing a generative probabilistic model called Latent Dirichlet Allocation(LDA) for modeling content representative of deep web databases. This is implemented after preprocessing the set of web pages to extract page contents and form contents.Further, we contrive the distribution of "topics per document" and "words per topic" using the technique of Gibbs sampling. Experimental results show that the proposed method clearly outperforms the existing clustering methods.

KEYWORDS

Latent Dirichlet Allocation, Latent Semantic Analysis, Deep Web, Cosine Similarity, Form Content and Page Content.

1. INTRODUCTION

1.1 Deep Web

The internet is a huge repository for an extremely wide range of information. Most of us access this information by querying through standard search engines. However, a portion of this World

David C. Wyld et al. (Eds) : ACITY, DPPR, VLSI, WiMNET, AIAA, CNDC - 2015
pp. 19–27, 2015. © CS & IT-CSCP 2015 DOI : 10.5121/csit.2015.51302

Wide Web content is not explored by any standard search engines[1]. This content is "masked" and is referred to as the deep web[2].

The deep web is significantly gigantic. According to a July 2000 white paper [3], the deep web is 500 times larger than the surface web and indeed, continues to proliferate this magnitude[3].From the surveys presented in [4],[5] we can get a lucid understanding of the surface web. There are links buried far down on sites thriving on the surface web, which direct us to the news and history of the Deep web. It is quite common to come across deep web pages that have links terminating with the extension '.onion'. Such web pages require us to access them through browsers named 'Tor', which connect to the servers containing the repository and get access consent[6].

Today, the 60 largest Deep Web sites contain around 750 terabytes of data, surpassing the size of the entire Surface Web 40 times. 95% of the Deep Web is publically accessible, which is free of cost. This is why search engines, such as Google, index well over a trillion pages on the World Wide Web, but there is information on the web that common search engines don't explore. Most of this constitutes databases of information that need to be searched directly from the specific website. A small pocket of the deep web is filled with hyper-secret communities who flock there to escape identification from authorities [7].

1.2 Topic Modeling

Topic models provide a simple way to analyze large volumes of unlabeled text. A topic consists of a cluster of words that frequently occur together. Using contextual clues, topic models can connect words with similar meanings and distinguish between uses of words with multiple meanings. Topic models express the semantic information of words and documents by 'topics'[8].

1.1 Clustering

Clustering is a division of data into groups of similar objects where each group consists of objects that are similar between themselves and dissimilar to objects of other groups.

From the machine learning perspective, clustering can be viewed as unsupervised learning of concepts. We can cluster images, patterns, shopping items, feet, words, documents and many more fields. Clustering has wide scope of application in data mining, text mining, information retrieval, statistical computational linguistics, and corpus based computational lexicography.

Clustering has the following advantages**:**

1. Clustering improves precision and recall in information retrieval.
2. It organizes the results provided by search engines.
3. It generates document taxonomies.
4. It also generates ontologies and helps in classifying collocations.

A good clustering algorithm needs to have the characteristics, mentioned below:

1. Scalability
2. High dimensionality

3. Ability to discover clusters of arbitrary shape

Considering the above factors and enormous size of documents, we have analyzed that Latent Dirichlet allocation outperforms the efficiency of other existing clustering algorithms [9].

2. RELATED WORK

Several deep web clustering approaches exist. Here, we briefly discuss the motivation for our work. The content in [11] proposes to cluster deep web pages by extracting the page content and the form content of a deep web page. Considering the extracted parts as words in a document, clustering is done. The results prove to be better than those of several previous models. However, for some datasets, words were clustered under irrelevant topics.

3. PROPOSED METHOD

A topic consists of a cluster of words that frequently occur together. Topic models can connect words with similar meanings and distinguish between uses of words with multiple meanings.A variety of topic models have been used to analyze the content of documents and classify the documents. For example, Latent Semantic Analysis compares documents by representing them as vectors, and computing their dot product. In our paper, we have used another such algorithm: Latent Dirichlet Allocation [10].

3.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a generative probabilistic topic model for collections of discrete data. The generative model describes that the documents are generated using the following algorithm. For each document:

1. Firstly, a distribution over topics is chosen randomly.
2. Now, for each word to be generated in the document,
   a. A topic is chosen from the distribution created in Step 1.
   b. Next, a word is chosen randomly from the corresponding distribution over vocabulary for the topic.

The Dirichlet prior on per document topic distribution is given by

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_i - 1} \ldots . \theta_k^{\alpha_k - 1} \quad (1)$$

Joint distribution of topic mixture θ, a set of N topic z, a set of N words w

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \quad (2)$$

α -hyper parameter on the mixing proportions (K-vector or scalar if symmetric).

β -hyper parameter on the mixture components (V-vector or scalar if symmetric).

Θ_m-parameter notation for p(z|d=m), the topic mixture proportion for document m. One proportion for each document, $\Theta = \{\vartheta\sim m\}$ M m=1 (M × K matrix).

Φ_k-parameter notation for p(t|z=k), the mixture component of topic k. One component for each topic, $\Phi = \{\phi\sim k\}$ K k=1 (K × V matrix).

N_m- Document length (document-specific), here modeled with a Poisson distribution with constant parameter ξ.

$Z_{m,n}$- Mixture indicator that chooses the topic for the nth word in document m.

$W_{m,n}$ - Term indicator for the nth word in document m.

In order to retrieve topics in a corpus, this generative process is reversed. The two distributions: distribution over topics and distribution over vocabulary for the topics are discovered by the reverse process. This process uses Gibbs Sampling, which is commonly used as a means of statistical inference. It is a randomized algorithm (i.e. an algorithm that makes use of random numbers, and hence may produce different results each time it is run). Gibbs sampling generates a Markov chain of samples, each of which is correlated with nearby samples.

3.2 Parsing

LDA is a document clustering algorithm. To give input to LDA as documents, we need to parse the input XML/HTML files. As mentioned earlier, the Page Contents and the Form Contents are extracted from the web pages.

1. Page Content: Values and textual content visible on the web page.
2. Form Content: The value of the attributes in the form.

Most of the deep web pages are blocked by forms (e.g. login forms). To take advantage of this, we take into consideration the form attribute values, which can reveal important information about the type of deep web pages. Thus, this approach covers a large portion of deep web database for clustering.

3.3 Visiting Inner Links

For some web pages, it is possible that there is very little information related to the topic it comes under. However, the webpage may contain links to other related sites, or even its home page. If we were able to visit the inner HTML links in the webpage, we could gather further information about the topic the webpage talks about. For this purpose, we perform parsing individually on all the links inside the webpage. This way, we also gather more relevant test data than what is already available.

However, is also possible to have irrelevant links inside the webpage. For example, there might be a quick link to a search engine, or advertisements. Since the web pages discovered by visiting the inner links are not so reliable, we must give them lesser weightage than the original data. We achieve this by increasing the frequency of the words in the original webpage, i.e., we repeat the words in the original webpage for a specific number of times.

3.4 Gibbs sampling

Gibbs sampling is one of the class of sampling methods known as Markov Chain Monte Carlo. We use it to sample from the posterior distribution, p(Z|W), given the training data W represented in the form of

$$W = \begin{bmatrix} \{w_1, \ldots, w_{N_1}\}, & \text{words in the 1st document} \\ \{w_{N_1+1}, \ldots, w_{N_1+N_2}\}, & \text{words in the 2nd document} \\ \ldots & \ldots \\ \{w_{1+\sum_{j=1}^{D-1} N_j}, \ldots, w_{\sum_{j=1}^{D} N_j}\} & \text{words in the } D\text{-th document} \end{bmatrix},$$

Gibbs sampling is commonly used as a means of statistical inference, especially Bayesian inference. It is a randomized algorithm (i.e. an algorithm that makes use of random numbers, and hence may produce different results each time it is run), and is an alternative to deterministic algorithms for statistical inference such as variational Bayes or the expectation-maximization algorithm (EM). Gibbs sampling generates a Markov chain of samples, each of which is correlated with nearby samples.

3. IMPLEMENTATION

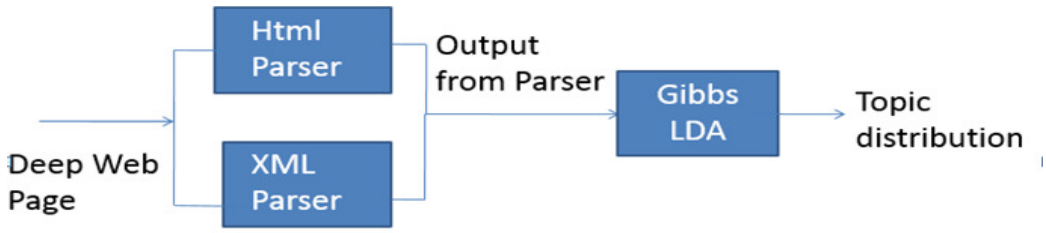

Fig 2: Module Diagram

We split the implementation into two parts:

1. Parsing
2. LDA

4.1 Parsing

Input: Deep web interface webpage (HTML/XML)

Output: Document with Form Contents and Page Contents as words.

4.1.1 Algorithm

1. From the input HTML/XML, gather all the textual content. Textual content refers to all the *text* that is visible on the webpage.
2. For each <form> element, gather all the values of the attributes for each of the child tags. For example, in the following we wish to extract the *"Search by Departure City"* and *"departure city"*.

<form>
<attrgroup>
<attr name="Search by Departure City" ename="departure city">
…
…
</attrgroup>
</form>

3. The words extracted from *Step 1* and *Step 2* constitute the *extracted words.* Write the *extracted words* into the output document.
    a. If the webpage is the original webpage interface, repeat the extracted word for a fixed number of times (say N), and write to the output document.
    b. If the webpage is among the ones discovered by visiting the inner links, write them to the document as it is.
4. Remove all the stop-words. Stop-words constitute certain regularly used words which do not convey any meaning, like *as, the, of, a, on,*etc.

4.1.2 Output Format

Google App Engine has been used for creating the web-tool for this project. The parser code written in Python takes in multiple HTML/XML files for input and gives the output after extracting the required words. Further, in order to remove grammatical constructs like ‘a’, ‘the’, ‘on’, we include a file of *stop-words*. All the words in this file are checked against the words obtained from the App Engine (Python code), and removed.

The input format for LDA is as follows:
<Number of documents: N>
<Document 1 - space separated words>
<Document 2 - space separated words>
…
…
<Document N - space separated words>
This output file from the python parser code produces the output in the above mentioned format. This output file is then given as input to the LDA code.

4.2 Gibbs LDA

Input:

1. A single file constituting the set of documents received after the Parsing stage. Multiple web pages parsed and given as input to LDA.

2. Number of Topics to be discovered.

Output: Clusters - A set of topics with most occurring words in each topic.

4.2.1Algorithm

As described earlier, we feed the Gibbs sampled data to our LDA model to extract topics from the given corpus.

5. EXPERIMENTAL RESULTS

In order to evaluate the performance of our approach we tested it over dataset of TEL-8 as mentioned in [8], UIUC Web integration repository [12]. The repository contains web search forms of 447 sources originally classified into 8 domains. The term TEL-8 refers to eight different web source domains belonging to three major categories (Travel, Entertainment and Living). Travel group is related to car rentals, hotels and airfares. Entertainment group has books, movies and music records interfaces. Finally the living group contains jobs and automobiles related query interfaces.

5.1 Performance measure

To evaluate the performance of our method, we calculated Precision of the outputs generated.In a classification task, the precision for a class is the number of true positives divided by the total number of elements labeled as belonging to the positive.

The definition of related terms is shown below:

1. TN / True Negative: case was negative and predicted negative
2. FP / False Positive: case was negative but predicted positive

[13] Precision = True Positives / (True positives + False Positives)

We tried various combinations of these 8 domains, and created 4 small datasets and 2 large datasets. We compared the approach used in [8] and our approach over the same dataset, and found that our method performs well in most cases.

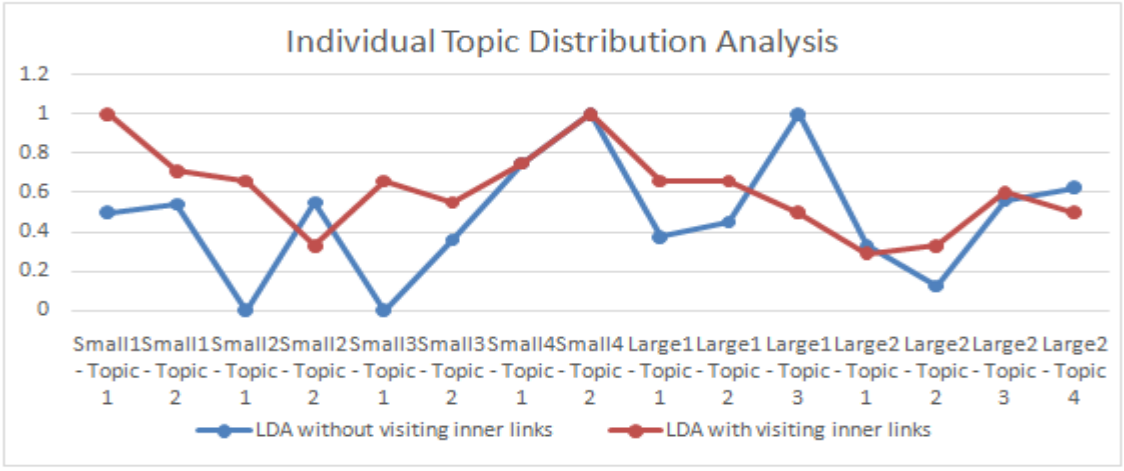

Fig 3: Individual Topic Distribution Analysis. Small refers to small dataset, and Large refers to large dataset. Several topics are discovered within for each dataset.

Though, this analysis clearly shows that our method performs better, this does not give the complete picture. For a better overall picture of the total topic distribution in each of the datasets, we use overall precision for each dataset.

We use the same formula for precision used above, and average it for the over the number of topics. Therefore, we get an averaged precision value for each dataset.

Averaged Precision = Sum of all precision values / Number of topics discovered

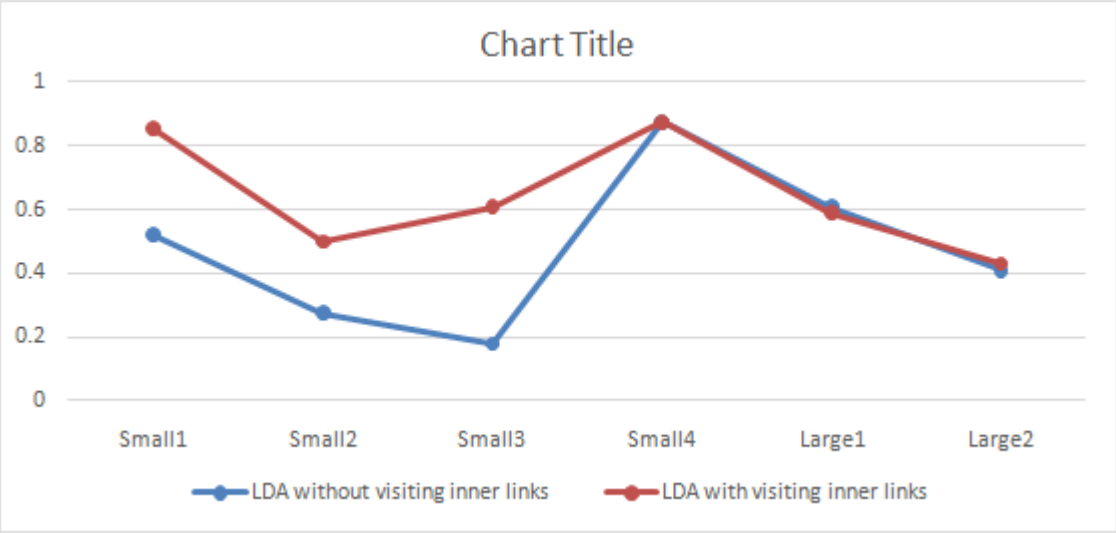

Fig 4: Performance evaluation for datasets

The above chart further strengthens the proof that the new method of visiting inner HTML links works better.

6. SUMMARY AND CONCLUSION

The requirement for an efficient and scalable clustering algorithm for deep web pages is fulfilled by our approach. The results show that sampling of data is useful for clustering heterogeneous deep Web sources. Our proposed Gibbs-LDA based technique is more efficient than the existing techniques.

It gives more accurate results when links in the web pages are parsed and their web page content are also taken into account. As LDA produce soft clusters it assigns probability to each document for each cluster. Hence, our tool is suitable for the scenario where the sources are sparsely distributed over the web.

In order to increase the weightage for the content in our base webpage over the one that we parse in our web page, we have increased the frequency of all the texts that appear in base webpage, which increases the space complexity. New innovative methods could be proposed in future which could possibly reduce space.

ACKNOWLEDGEMENTS

We would like to thank our Project guide, Dr. E. Sivasankar and Ms.Selvi Chandran, Research Scholar, CSE, NIT-Trichy. Their input, guidance and encouragement made this project possible and a success.

REFERENCES

[1] The Deep Web: Surfacing hidden value. Accessible at http://brightplanet.com, (2000).
[2] Madhavan, J., Cohen, S., Dong, X. L., Halevy, A. Y., Jeffery S. R.,Ko, D. and Yu, C (2007),"Web scale data integration", Conference on Innovative Data System Research (CIDR), 342–350.
[3] Tor: The Second-Generation Onion Router. Accessible at www.onion-router.net/Publications/tor-design.pdf
[4] Vincenzo Ciancaglini, Marco Balduzzi, Max Goncharov, and Robert McArdle, "Deepweb and Cybercrime: It's Not All About TOR", Trend Micro
[5] David M. Blei, Probabilistic topic models (2012), communications of the acm, Vol.55, No.4
[6] Estivill-Castro, Vladimir (2002), "Why so many clustering algorithms: a position paper", ACM SIGKDD Explorations Newsletter 4.1
[7] Blei, D., M., Ng, Y., A. and Jordan, M., I. (2003)," Latent Dirichlet Allocation", Journal of Machine Learning Research.
[8] Umara Noor, Ali Daud, Ayesha Manzoor (2013)," Latent Dirichlet Allocation based Semantic Clustering of Heterogeneous Deep Web Sources", In: Intelligent Networking and Collaborative Systems (INCoS), 132- 138.
[9] Unsupervised Learning. Accessible at http://mlg.eng.cam.ac.uk/zoubin/papers/ul.pdf, (2004).
[10] Thanaa M. Ghanem and Walid G. Aref. (2004), "Databases deepen the web". IEEE Computer, 37(1), 116–117.
[11] Dennis Fetterly, Mark Manasse, Marc Najork, and Janet Wiener (2004) "A large-scale study of the evolution of web pages." In Proceedings of the 12th International World Wide Web Conference, 669–678.
[12] Steve Lawrence and C. Lee Giles (1999), "Accessibility of information on the web." Nature, 400(6740), 107–109
[13] A Theoretical and Practical Implementation Tutorial on Topic Modeling and Gibbs Sampling. Accessible at http://u.cs.biu.ac.il/~89-680/darling-lda.pdf, (2011)
[14] The ElementTree XML API. Accessible at https://docs.python.org/2/library/xml.etree.elementtree.html
[15] The Bayesian Approach. Accessible at https://www.cs.cmu.edu/~scohen/psnlp-lecture6.pdf
[16] Google App Engine: Platform as a service. Accessible at https://cloud.google.com/appengine/docs
[17] Stop Words. Accessible at http://www.webconfs.com/stop-words.php
[18] The UIUC Web integration repository. Accessible at http://metaquerier.cs.uiuc.edu/repository
[19] Precision and recall .Accessible at en.wikipedia.org/wiki/Precision_and_recall.
[20] Clustering: Overview and K-means algorithm. Accessible at http://www.cs.princeton.edu/courses/archive/spr11/cos435/Notes/clustering_intro_kmeans_topost.pdf