Topic Retrieval and Articles Recommendation

Yu Wang, Xinyu Shen, Jinzhi Wang

Abstract
How to search articles effectively is significantly important to researchers in academia. Currently researchers use search engines like Google Scholar and search by keywords (eg. machine learning, topic modeling, k-means etc.). The typical search results are large amount of articles, which match the keywords exactly but are on many different topics. It is very time and effort consuming to read through the papers manually and select out desirable ones. We propose a solution to search papers by topic. We apply the Vector Space Model and tf-idf to vectorize and model documents. Then we use k-means algorithm and Latent Dirichlet Allocation (LDA) method to train on a large document set and analyze every paper in it to generate its distribution over topics. The algorithm recommend papers to reader by analyzing whatever the reader has read and compare with the distribution of all papers in the training set. The papers with most similar distributions are those recommended to the user. In this way we realize searching in the database by topics rather than keywords. We use 1298 papers from past CS 229 course project reports as data source. Without any title or keywords matching, experimental results have demonstrated that our method can successfully extract the topics beneath the words of an article and recommend closely related ones.

Keywords
Vector Space Model, Text feature, tf-idf, k-means, LDA, Topic Distribution, Recommendation

Contents

1. Introduction

1.1 Motivation

In academia, a proper method to effectively extract topics from papers and search for papers with similar topics have long been desired. Currently, people are familiar with searching by keywords using tools like Google Scholar. However, these searching results are merely based on the matching of input keywords. But we want to search by the semantic contents, which is beyond keyword matching. Even by skimming condensed abstract section, substantial time will be consumed to acquire the topics on literature review. Especially considering the vast volume of literature on Internet, it is almost impossible to retrieve accurate topics and find best matching papers manually.

1.2 Objectives and Outline of Our Approach

The problem we are to address is that given a user input — a paragraph of interest or several journal papers which we call reading list — how to create a recommendation list of papers to help the user decide a clear order of pursuing and provide him/her with abundant information as well.

As a rough depiction of the framework of our proposed method, we will first need a collection with enough literatures, ideally across many different areas. It would be the best if we have a literature database like ScienceDirect. Then we process on the papers in the collection, including format converting, word trimming, high and low frequency word filtering and non-English word removal to obtain a training set formed by word frequency statistics. Then we will calculate topic distribution for each document, given the number of topics arbitrarily selected. When making recommendations, the algorithm analyze the reading list and compute its topic distribution. Then the algorithm judge the similarity between this distribution and every document's distribution in the training set and recommend the most similar ones.

1.3 Project Benefits

Our method can be used as an effective way of paper recommendation. First, by topic modeling, we will improve both the efficiency and accuracy of paper searching towards a particular topic. Second, by conducting topic search, it is easier to find important literature bridging two different academic fields.

2. Background and Literature Review

In our project, we use an advanced topic modeling method to help researchers explore and browse large collection of archives. Historically, topic modeling can be used to improve the accuracy of classification by their contextual information. For example: Guo et al.[1] studied the geographical location to classify different type of tours. Ka-Wing Ho[2] tried to solve the problem of genres classification of moves by considering it as a multi-label classification problem. In their work, they both used topic models to organize information.

In recent years, some research concerning topic modeling focused on how to train the dataset in the environment of twitter[3]. The authors demonstrated that topic models training quality will be influenced by length of document. Latent Dirichlet Allocation (LDA) model and author-topic(AT) model, which is an extension of LDA are used and compared in this research, and LDA had better performance in their settings. One more interesting paper explores the topic and role of author in a social network[4]. This paper introduced the author-recipient-topic (ART) model, which is based on LDA and AT models, considering the attribute of topic allocations with factors tuned from sender-recipient perspective. Another paper introduces the relational topic model (RTM), which is a hierarchical model with network structure and node attributes[5]. Their research focus was on words in each documents and they realized summarization of document network, with predicted links and words amongst them. The RTM model is built upon mixed-membership model, and it also reuses the statistical assumptions behind LDA.

Previous research and other related works will be served as reference and comparison of our research work. We incorporated some ideas, applications and structures from them as a reference.

3. Methodologies and Algorithms

3.1 Data and preprocessing

Data source currently used in the research is previous CS229 course project reports. We are using all the project reports from year 2011 to 2015 and the number of articles is 1298. Ideally online open-source research papers are our best source of data but let's restrict the data source to course reports for testing purposes.

The first step is to convert formatted PDFs to plain TXT files. The articles are published on course website in PDF format, as are most of the other research literatures. They are converted to program-readable plain TXT files with no information lost or distorted using an online tool `http://pdftotext.com/`. We then remove all numbers, signs, symbols and non-English letters and convert all English letters to lowercase. Next Porter stemming algorithm is applied to remove the morphological endings of the words in the documents. For example make, makes, making are stemmed to "mak" and surprise, surprises, surprising, surprised, surprisingly to "surpris". The last step of text processing is to remove trivial stop words like the, and, was, we etc. from the data. After all these preprocessing steps, we finally have the non-trivial, and sequential (as in the original article) English words written in plain TXT files.

Denote the document set by $D = \{d_1, d_2, \cdot, d_m\}$, the text feature set bt $W = \{w_1, w_2, \cdots, w_n\}$.

We apply the Vector Space Model on all the documents. Every unique stemmed word in a document is treated as a dimension. The magnitude of vector component in that dimension is the total number of word appearance in that document. Repeat this process for all the documents and we generate the text feature matrix X. The elements of matrix X is

$$X_{i,j} = \sum_{word\ in\ d_i} 1\{word = w_j\} \tag{1}$$

Now every column in matrix X represents a feature for learning algorithms to learn later. We then apply threshold on every element of X to filter out words like names, abbreviations, etc.

$$X_{i,j} = \begin{cases} X_{i,j} & X_{i,j} \geq threshold \\ 0 & X_{i,j} < threshold \end{cases} \tag{2}$$

By testing, we find 3 is a good choice for threshold. A thing worth mentioning here is after the thresholding, many columns in matrix X will become all zeros, which means the corresponding feature is considered indiscriminative. We remove all these zeros columns for the sake of both code efficiency and features' meaningness.

Before finally send X to learning algorithm, we further process matrix X using the concept of inverse document frequency to account for the intuition that a word appearing in too many incidents should be less discriminative that a word that appears only in few documents, regardless of their actually frequency in the individual documents.

$$X_{i,j} = \ln \frac{|D|}{1 + |\{d : w_j \in d\}|} \sum_{word\ in\ d_i} 1\{word = word\ j\} \tag{3}$$

The weight $\ln \frac{|D|}{1+|\{d:w_j \in d\}|}$ emphasized more the words appearing in less documents. 1 is added to avoid zero denominator.

3.2 Training Algorithms

As stated before, the final training set is presented to the training algorithm as the matrix X with rows representing training samples and columns representing the text features.

In the training part, we applied both k-means and Latent Dirichlet Allocation (LDA) to perform unsupervised learning.

Unsupervised learning is the natural choice here because the lack of label for each sample, i.e. topic classification of each course report.

At the end of training process, both k-means and LDA will analyze all reports and calculate weight distributions over topics for each of them. The weight distributions indicate the likelihood of a document belonging to each topic. The similarity between these distributions will be our criterion for recommendation.

In the following discussion, topic and cluster essentially mean the same thing. We set the topic/cluster number k to be 20.

3.2.1 k-means

In our case, the data sent to k-means algorithm is X with 1298 rows and 39588 columns. We first trained the data using k-means clustering method. We used cosine distance in our algorithm.

Denote the row vectors of X by $\{\vec{r}_1, \vec{r}_2, \cdots, \vec{r}_n\}$

$$CosDis(d_i, d_j) = 1 - \frac{\vec{r}_i \cdot \vec{r}_j}{||\vec{r}_i|| \cdot ||\vec{r}_j||} \tag{4}$$

Where d_i, d_j are i^{th} and j^{th} document.

Cosine distance or cosine similarity measures the cosine of angle between two vector. It can be easily applied to any high dimension space. Unlike Euclidean distance, it measures the difference in orientation rather than magnitude. Cosine distance is suitable for text vectors because scaling a text vectors should not change the topic of this document.

After convergence, k-means assigns each report to a cluster. The next step is to analyze the weight distribution over all topics for each report. In our vector model for documents, each vector in high dimension represents a document. The intuitive idea is that if a vector is surrounded by many vectors belonging to cluster i, then this vector should have a high weight on cluster i. In practice, we artificially set a threshold ε and define hypercone $|\vec{r} - \vec{r}_i| < \varepsilon$ to be the ε vicinity of $\vec{r}_i$. We admit there is some arbitrariness in choosing ε. We count the number of vectors assigned to each topic in the vicinity of $\vec{r}_i$. The number count can be presented as $n_1, n_2, \cdots, n_k$. After normalization, i.e. divided by the total number n of vectors in the vicinity, we have the weight distribution of document $\vec{r}_i$ over all topics. The distribution is $\frac{n_1}{n}, \frac{n_2}{n}, \cdots, \frac{n_k}{n}$. Denote the distribution matrix for all vectors by T_{kmeans}.

3.2.2 Latent Dirichlet Allocation (LDA)

LDA calculate the weight distribution using a different method. We apply the MATLAB LDA programs offered by `http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm`.

The LDA algorithm has the exact same input as k-means, i.e. the matrix X (m by n) mentioned above. Each row of X is a sample and each column is a word.

The algorithm outputs another matrix Z (n by k), with its rows represent words and columns represent topic. $Z_{i,j}$ is a number between 0,1 and it indicates how likely word i (w_i) belongs to topic j ($j \in [0,k]$). With matrix X and Z, we can calculate the weight distribution over topics for all documents. First we need to normalize X by row and get $\hat{X}$. Then $T_{LDA} = \hat{X}Z$.

Denote the rows in matrix T_{kmeans} and T_{LDA} by wd_i.

3.3 Recommendation Algorithm

As mentioned in the motivation part, we want to extract reading interest and topic preference from the reports the reader has already read and recommend according to similarity between weight distribution over topics.

In practice, given the reading list containing papers the reader has read, we can calculate the word frequency vector $\vec{r}$. For k-means algorithm, we put $\vec{r}$ in the high dimension space and count the number of vectors in each topic in its ε vicinity. The procedure is the same as above. For LDA algorithm, we normalize $\vec{r}$ by its total number of words and get $\hat{\vec{r}}$, then $\hat{\vec{r}}Z$ is the weight distribution over topics.

Denote the weight distribution of this new reading list by wd. By comparing the new distribution wd to row vectors wd_i in matrix T_{kmeans} and T_{LDA}, we are able to rank the reports in the training set by their similarity to wd and recommend those with highest similarity.

To judge the similarity between wd and wd_i, we use both KL-divergence and cosine distance.

4. Results and Discussion

4.1 k-means clusters visualization

The actual algorithm used all 1298 reports from 2011-2015 and the cluster number was set to 20. The scale of this set is too large to be clearly presented in the final report, so we run a clustering on 2015 reports. By doing so we can take advantage of the classification available on the course website `http://cs229.stanford.edu/projects2015.html`, and compare them with clustering results. Note the course website only provide classification for 2015 reports.

4.1.1 Clusters

Table 1. Sample Number in clusters

Cluster	Website number	k-means number
Finance & Commerce	42	48
General Machine Learning	42	cannot tell
Natural Language	40	45
Life Sciences	38	30
Computer Vision	33	37
Audio & Music	25	24
Physical Sciences	23	51
Athletics & Sensing Devices	18	20
Theory & Reinforcement	14	cannot tell
Recommendation	N/A	24

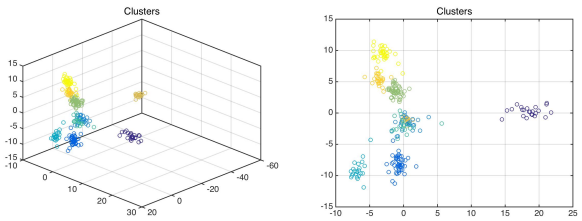
After all the preprocessing, matrix X for 2015 reports has 273 rows and 17202 columns. We tried various cluster numbers ranging from 1 to 15. Experiments show that cluster number between 8 to 10 give the best results. We judge the clustering results by extracting the most frequency words in each cluster that are not present in other clusters and manually determining if they are closely related. It turned out that all the 2015 project reports are already classified into 9 groups according to the course website. Thus we clustered the reports into 9 categories and compared the number of samples in each cluster with results on the website. They are shown in Table 1. The most frequent words in each cluster are shown in Table 2. As one can see from both Table 1 and

Table 2. Most frequent words in each cluster

Finance & Commerce	sale store price stock market trade company strategy employee return
Natural Language	bill document token sentenc answer tf idf question text recip link code comment
Life Sciences	gene patient tumor cancer thyroid cell diseas diabet surviv brain tissu
Computer Vision	image color pixel cnn face convolut layer descriptor object recognit visual neural webcam
Audio & Music	song music audio sound chord note voic speaker genr melodi frequenc guitar
Physical Sciences	Sensor activ packet jet traffic motion damag simul quantum structur memori seismic acceleromet
Atheletics & Sensing Devices	game player team season win nba polici quarterabck hero football borad defens statist injuri
Recommend System	user review busi rate movi restaurant yelp recommend item star factor

Table 2, k means algorithm clusters narrow topics well, which usually comes with "landmark" words, like "sale" and "stock" to "Finance and Commerce"; "gene" and "cancer" to "Life Sciences"; "song" and "music" to "Audio & Music", etc. K means can not perform well for the "General Machine Learning" and "Theory & Reinforcement" categories. This is because all the documents are CS229 course projects and every report more or less mention the key words like "svm", "learning", "supervised". When we conduct clustering on this document set, key words to machine learning area will become completely indiscriminative. What we learn here is that we have to provide a training set that contains different examples in some aspects to cluster well.

In general, the training is satisfying and clustering is rather obvious. All 273 samples are automatically labelled by k means algorithm. Labeling can be manually done in future for higher precision.



(a) Clusters in 3d space **(b)** Clusters in 2d space

Figure 1. Clusters

Table 3. Recommendation list

Reading List	
Machine Learning Applied to the Detection of Retinal Blood Vessels	Supervised DeepLearning For MultiClass Image Classification
Top 3 Recommendation List	
k-means	**LDA**
Implementing Machine Learning Algorithms on GPUs for Real-Time Traffic Sign Classification	Pedestrian Detection Using Structured SVM
Equation to LaTeX	FarmX: Leaf based disease identification in farms
Object classification for autonomous vehicle navigation of Stanford campus	Identifying Gender From Images of Faces

4.1.2 Visualization

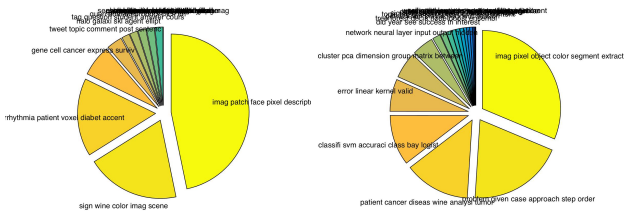
We also plotted the high dimension vectors in 3D space after dimension reduction using PCA algorithm. The clusters in 3D space are shown in Figure 1a, where the coordinates are the reduced dimension and each color marks a different cluster. Figure 1b is a 2D side view of Figure 1a. As one can see, 6 of the 9 clusters are separated well but the rest 3 are not as good.

4.2 Recommendation

We provide recommended papers based on the reading history of users. Two pieces of articles are randomly selected from the dataset as user's reading list (testing data). By using methodology stated in Methodology section, we generated corresponding reading list papers compound distribution for both k-means model and LDA model. Figure 2 illustrates the weight distribution over topics for the reading list. The largest few sections relate to topics like image, classification, and supervised machine learning. Consistency between these two method can be observed here.

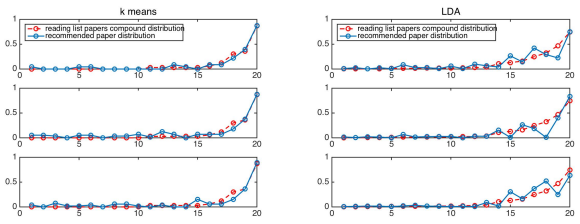
The recommendation algorithm returned three papers for both of the models, as shown in Table 3.

We can expect, just from the title, that the two input papers are related to image detection or classification using super-



(a) k-means **(b)** LDA

Figure 2. Weight distribution of reading list



(a) k-means **(b)** LDA

Figure 3. Weight distribution of reading list and recommend reports

vised machine learning approaches. And the returned result from both models share similar topics, indicated by keywords in titles, like "traffic sign classification", "object classification", "pedestrian detection", "SVM", "identification", "images of faces", etc. Noticing that one of the recommended papers is named "Equation to LaTex", this result demonstrated the distinct advantage of our model over regular search engines - it discovers the topic of documents without replying too much on keywords like "image", "machine learning" or "detection". This paper actually describes how to detect equations in PDF documents and convert them to LaTeX codes by machine learning approaches, which is a good reference for the user who intends to inquire topics like image processing with machine learning methods.

Figure 3 illustrates the comparison between the weight distribution over topics of reading list papers and recommended paper. The red lines are the distribution of the reading list and blue lines represent recommended reports. As shown in the plots, documents recommended by k-means method have a very similar distribution with the reading list's distribution, while distributions of documents recommended by LDA deviate more, which may indicate more variance error. But this does not necessarily mean k-means perform better because the way of counting paper numbers in the vicinity of reading list has an averaging effect and might facilitate the similarity between different document's distribution.

To conclude, k-means and LDA model respectively on distribution over topics of documents and words. The recommendation algorithm following each of them recommends different results but they all satisfy our expectations. Our algorithms successfully achieve our goal of topic extraction and article recommendation.

Acknowledgments

The three of us would like to thank Professor Duchi for teaching this great class and offering the opportunity of working on a project as a team. We also want to thank all the TAs, who have always been helpful in both the project and homeworks. Last but not least, we thank all our classmates who we have sought help from.

References

[1] Alan Guo, Chanh Nguyen, and Taesung Park. Building a better tour experience with machine learning.

[2] Ka-Wing Ho. Movies genres classification by synopsis.

[3] Liangjie Hong and Brian D Davison. Empirical study of topic modeling in twitter. In *Proceedings of the first workshop on social media analytics*, pages 80–88. ACM, 2010.

[4] Andrew McCallum, Andres Corrada-Emmanuel, and Xuerui Wang. Topic and role discovery in social networks. *Computer Science Department Faculty Publication Series*, page 3, 2005.

[5] Jonathan Chang and David M Blei. Hierarchical relational models for document networks. *The Annals of Applied Statistics*, pages 124–150, 2010.