Mini Search Engine for Research Papers Database

Phu Duong
Department of Computer Science
Iowa State University
Ames, Iowa 50011
dtmyphu@iastate.edu

Motivation and Goal

Searching documents over internet has been playing an important role in our current digital life. The emergence of world wide web has resulted the exponential increase of web data which contain billion of heterogeneous documents. Search engines (e.g., Google[1], Bing[2]) are now the essential systems that facilitate users to quickly find a specific document over web environment by accurately matching users' queries with web documents.

In this project, we aim to design and implement a mini search engine that enables users to search research papers based on the keywords entered. Searching results will be the top K documents which are "similar" to the query. We first design the search engine framework and algorithms used in processing textual documents and we then implement our system. We apply our system on computer science research paper database which include about 37000 papers. In the following sections, our design and algorithms are developed based on the structrure of the data set (i.e., the research paper data set).

Search Engine Framework

In this section, we describe the search engine framework. In particular, we show step by step how to build a search engine for a database.

Processing paper database

The data from research papers are textual data. For each paper, we extract the title, abstract, author and publish date. Only title and abstract are used for searching purpose. In particular, we process the database in four steps.

- Paper Text Data: Assume that we have a collection of paper (thousands). Each paper is in raw format containing all text data. In this scope of this project, we extract title and abstract as the textual features which are used for searching.
- Paper Pre-Processing: The textual data of the paper is preprocessed using some textual processing technique like stopword removal (e.g., is, are, you, him), stemming (i.e., going$\rightarrow$ go) and creating dictionaries.
- Word Distribution Format: This format is done by using the term (belonging to a dictionary) frequency of each paper. We then normalize the term frequency of each paper to make a paper into a distribution over words.
- Paper Feature Database: Word distributions of the papers are saved in the database for the purpose of doing the similarity calculation with the query.

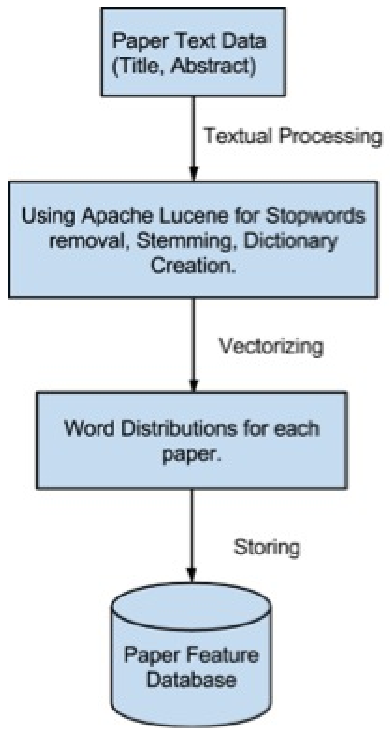
Paper Text Data (Title, Abstract)
Textual Processing
Using Apache Lucene for Stopwords removal, Stemming, Dictionary Creation.
Vectorizing
Word Distributions for each paper.
Storing
Paper Feature Database

Figure 1: Processing Paper Database

Processing the user query

Query submitted by a user is under raw text data. We do feature extraction on the query before comparing it with documents in database. The procedure of extracting features from

[1]`www.google.com`
[2]`www.bing.com`

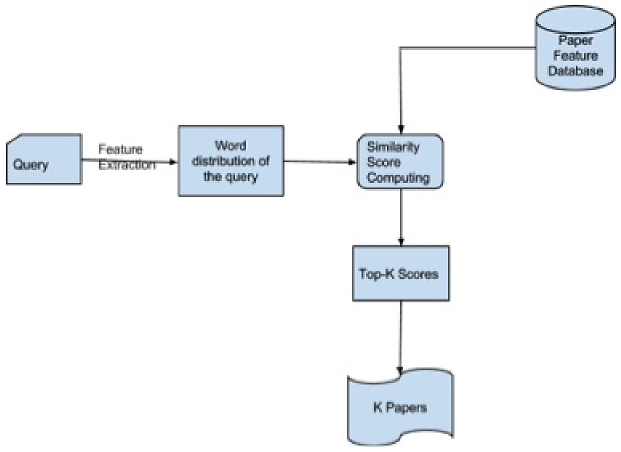

Figure 2: Processing the User Query

a query is quite similar to the one applied to a paper. Here are the steps of processing a query.

- Query Feature Extraction: Each query will be converted into a word distribution like one paper in the previous subsection.
- Similarity Score Computing: For each paper in the database, we calculate the similarity score with the query. The similarity metric used here is Jensen Shannon Divergence (Manning and Schütze 1999). The smaller the divergence between the two distributions of the query and the paper the higher their similarity
- Searching Results: We choose the top K papers that approximately similar to the query and output to the user.

Algorithms

We use the title and/or abstract of papers in the database to represent each paper by a *bag of words*. Let $W = \{w_1, w_2, \ldots, w_l\}$ be the set of distinct words occurring in the title and/or abstract of papers. We define the word distribution P_k of a paper d_k as the conditional probability distribution $(p_k(w_1), \cdots, p_k(w_l))$ where $p_k(w_i) = P(w_i|d_k) = n(d_k, w_i)/\sum_{r=1}^{r=l} n(d_k, w_r)$ over the distinct word set W, where $n(d_k, w_i)$ is the number of occurences of word w_i in the text data of the paper d_k.

We use the Jensen-Shannon divergence $D(P_k||P_j)$ between their word distributions P_k and P_j to calculate the distance between two papers d_k and d_j. The smaller the divergence between the two word distributions of the two papers the larger their similarity. The Jensen-Shannon divergence between two probability distributions P_k and P_j is defined as:

$$D(P_k||P_j) = \frac{1}{2}\left[\sum_i p_k(w_i)\log\left(\frac{2p_k(w_i)}{p_k(w_i)+p_j(w_i)}\right) + \sum_i p_j(w_i)\log\left(\frac{2p_j(w_i)}{p_k(w_i)+p_j(w_i)}\right)\right]$$

Data set

We use the data set from Andrew McCallum group[3]. Specifically, we use the data set Cora Research Paper Classification (McCallum). In this data set we extract about 37000 papers where each paper has contain authors, title, abstract, published date.

Preprocess data

We first extract the title, abstract of each paper and use them as the textual data of one paper. We then use Apache Lucence package [4] to do stopword removal, stemming. We make use of three searching criterion. They are searching based on title, abstract, and both title and abstract. We remove the infrequent term for each case. In particular, terms whose frequency less than 10, 20, and 20 in title, abstract, title-abstract, respectively are removed.

For each case, we have a separate dictionary. Specifically, the dictionaries for title, abstract, title+abstract have 2972, 7493, and 7755 terms, repectively.

Result

In this section, we show the demonstration of the mini search engine where K is set to be 10 and searching based on the title only.

- $Query$ = "Hierarchical Mixtures of Experts"

1. Title: *Adaptively Growing Hierarchical Mixtures of Experts* - Author: Jurgen Fritsch, Michael Finke, Alex Waibel
2. Title: *Hierarchical mixtures of experts and the EM algorithm* - Author: Michael I. Jordan Robert A. Jacobs
3. Title: *Constructive Algorithms for Hierarchical Mixtures of Experts* - Author: S.R.Waterhouse A.J.Robinson
4. Title: *Classification Using Hierarchical Mixtures of Experts* - Author: S.R.Waterhouse A.J.Robinson
5. Title: *Hierarchical Mixtures of Experts and the EM Algorithm* - Author: Michael I. Jordan and Robert A. Jacobs
6. Title: *A Hierarchical Community of Experts*- Author: Geoffrey E. Hilton Brian Sallan and Zoubin Ghahramani
7. Title: *Bayesian Methods for Mixtures of Experts* - Author: Steve Waterhouse David MacKay Tony Robinson
8. Title: *What Experts Deny, Novices Must Understand* - Author: Michael Miller Don Perlis and AV Williams Bldg
9. Title: *Mixtures of Experts Estimate A Posteriori Probabilities* - Author: Perry Moerland
10. Title: *C IDIAP Martigny Valais Suisse Some Methods for Training Mixtures of Experts* - Author: Mucto P Perry Moerlnd

- $Query$ = "Reinforcement Learning Algorithms for Average-Payoff Markovian Decision Processes"

[3] `http://people.cs.umass.edu/~mccallum/data.html`

[4] `https://lucene.apache.org/`

1. Title: *Reinforcement Learning Algorithms for Average-Payoff Markovian Decision Processes* - Author: Satinder P. Singh
2. Title: *Learning to Solve Markovian Decision Process* - Author: Satinder P. Singh
3. Title: *Generalized Markov Decision Processes: Dynamic-programming and Reinforcement-learning Algorithms* - Author: Csaba Szepesvari Michael L. Littman
4. Title: *Learning Decision Theoretic Utilities Through Reinforcement Learning* - Author: Stensmo, M. and Sejnowski, T. J. (). Magnus Stensmo Terrence J. Sejnowski
5. Title: *HQ-Learning: Discovering Markovian Subgoals for Non-Markovian Reinforcement Learning* - Author: Marco Wiering Jurgen Schmidhuber
6. Title: *Reinforcement learning for realistic manufacturing processes* - Author: Stephan ten Hagen and Ben Krose
7. Title: *Neurocontrol by Reinforcemen Learning* - Author: G. Schram B.J.A. Krose flfl R. Babuska A.J. Krijgsman
8. Title: *Applicability of Reinforcement Learning* - Author: Paul E. Utgoff Paul R. Cohen
9. Title: *Learning Without State-Estimation in Partially Observable Markovian Decision Processes* - Author: Satinder P. Singh Tommi Jaakkola Michael I. Jordan
10. Title: *Hierarchical Control and Learning for Markov Decision Processes* - Author: Ronald Edward Parr

Summary and Discussion

In this study, we design framework and algorithms and implement a mini search engine to search on research paper database. We first do preprocessing on the textual data and then we do feature extraction from the papers. The submitted query is also first preprocessed and extracted feature before comparing with the documents in the database. We use Jensen-Shannon divergence to compute the similarity between the document in the database and the query. The top K (user specified) documents that are similar to the query are returned to the user.

There are some limitations exist in this study. Each time the user submits the query, then the process of comparing query to all documents in the database is required. This will be take time when the database become larger. The work can be improved by proposing a mechanism to cache the previous query results to enhance the performance of the search engine. Another solution is to do the pre-clustering the whole database into several smaller topics and we only do comparing query with documents on clusters (subsets) of documents that have the topics closely related to the query.

Future work

Future work of this study is that we will enhance the performance of the search engine by tackle the limitation mentioned above. Another interesting thing is answering the question: How good are the results to the query? In order to answer this question, we will propose metrics which are used to measure the goodness of the results. Some simple intuitions are calculating the number of occurences of keywords/phrases/ which both appear in query and research. Another thing is that besides Jensen-Shannon divergence, we can try on different similarity metrics.

References

Manning, C. D., and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambrige, Massachusetts.

McCallum, A. Cora research paper classification, http://people.cs.umass.edu/ mccallum/data/cora-classify.tar.gz.