Phrase Mining for Topic Modeling

1 Introduction

Suppose we have hundreds of articles/sentences. We want to know what topics each of those articles/sentences talk about. An article describing allegations made on a pharmaceutical company may talk about topics like Government, Medicine or Business. Our goal is to assign these topics to documents. There're lots of algorithms developed to achieve this goal.

In general, three strategies on phrase mining with topic modeling:

- Strategy 1: Generate bag-of-words → generate sequence of tokens
- Strategy 2: Post bag-of-words model inference, visualize topics with n-grams
- Strategy 3: Prior bag-of-words model inference, mine phrases and impose on the bag-of-words model

In paper[1], it proposed a new methodology ToPMine that demonstrates both scalability compared to other topical phrase methods and interpretability. It does Phrase mining 1st, then does phrase-based topic modeling. This algorithm is efficient and generates high-quality topics and phrases without any training data.

2. Problem Description

For MP3 program assignment(PLSA algorithm implementation) , we need to construct the term-document matrix where each row represents a document and each column represents a vocabulary term. If a collection of documents is big, the term-document matrix will be huge. For example, a collection of documents like 'DBLP.txt" has the 10835 documents and it has 30140 vocabulary. That means the term-document matrix size is 10835x30140. Even after removing the stop words, the vocabulary size is still very big(29804). Per PLSA algorithm, we need to calculate the current log likelihood of the model using the model's update probability matrices. Given this huge matrix, the program is stuck during running.

To reduce the vocabulary size, I use the Aproio Algorithm and set the Minimum Support to various numbers. The results are shown as Fig.1. We can see that even we set the MIN_SUPPORT to 100, the vocabulary size is still over 2000(2108). We will still run into program running time problem for PLSA. It's not efficient and Scalability for this.

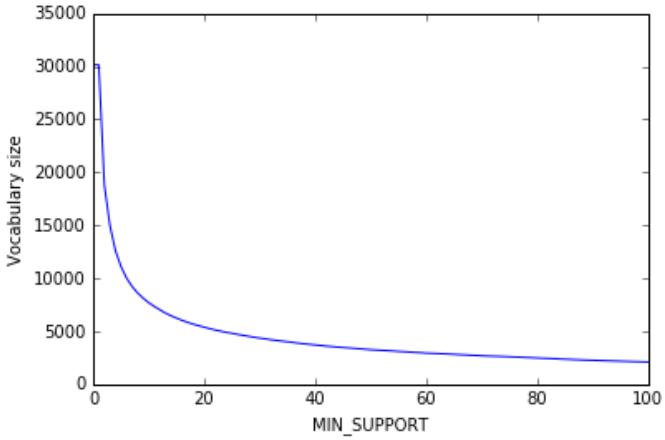

Figure 1 Vocabulary Size(bag-of-words) Vs MIN_SUPPORT

3. Phrase Mining

In paper [1], it performs frequent contiguous pattern mining to extract candidate phrases and their counts. Then it performs agglomerative merging of adjacent unigrams as guided by a significance score. This segments each document into a "bag-of-phrases".

There're two fundamental properties for efficiently mining these frequent phrases.

1. Downward closure lemma: If phrase P is not frequent, then any super-phrase of P is guaranteed to be not frequent.

2. Data-antimonotonicity: If a document contains no frequent phrases of length n, the document does not contain frequent phrase of length >n.

By doing this, the vocabulary size can be reduced significantly. For "DBLP.txt" data, the vocabulary size is 142 after frequent phrase pattern mining when the MIN_SUPPORT is set to 100. The vocabulary size(bag of phrase) vs MIN_SUPPORT is shown as Fig.2. We can see that the vocabulary size is significantly reduced, which can speed up the calculation.

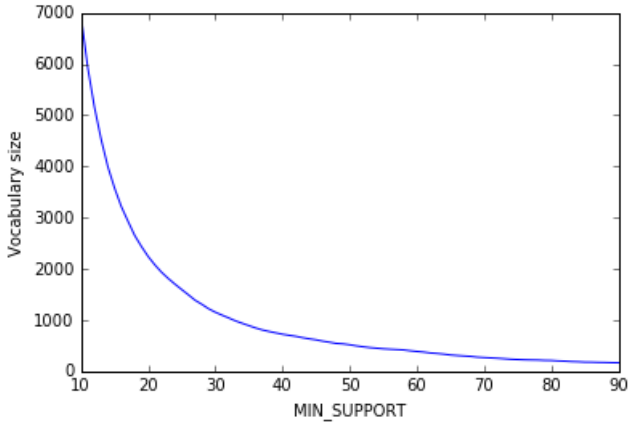

Figure 2 Vocabulary Size(bag of phrase) vs MIN_SUPPORT

4.Topic Modeling

In paper[1], it assumes tokens in the same phrase should share the same latent topic. After it generates the 'bag of phrases', it uses the 'bag of phrases' as the input for its new probabilistic model, PhraseLDA. I think we can apply the same 'bag of phrases' method to PLSA as well.

5.Conclusion

Before topic modeling, we should consider phrase mining 1st. We mine phrases by finding frequent sequential patterns from the documents. A phrase is a sequence of words. With a sequential pattern mining algorithm we can easily find these sequences. Once we get this 'bag-of-phrases', we can use it as the input to the LDA or PLSA topic modeling to generate the topics.

ToPMine demonstrates scalability on large datasets and interpretability in its extracted topical phrases beyond the current state-of-the-art methods.

Reference

[1]Ahmed ElKishky, Yanglei Song, Chi Wang, Clare R. Voss, Jiawei Han "Scalable Topical Phrase Mining from Text Corpora", Proceedings of the VLDB Endowment, Vol. 8(3), pp. 305 - 316, 2014

[2] P. Schone and D. Jurafsky.” Is knowledge-free induction of multiword unit dictionary headwords a solved problem?” Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing, pages 100-108, 2001.

Appendix.

For DBLP.txt data, after performing the topmine phrase mining algorithm, we can get 142 phrase and the run time is 19.559295177459717 seconds.

Phrase Data:
[('paper presents', 817), ('paper propose', 786), ('paper present', 759), ('real time', 729), ('experimental results', 563), ('results show', 555), ('real world', 549), ('state art', 448), ('large scale', 435), ('data sets', 412), ('run time', 387), ('high level', 372), ('data mining', 365), ('experimental results show', 361), ('paper describes', 352), ('access control', 345), ('paper proposes', 318), ('object oriented', 314), ('high performance', 306), ('energy consumption', 284), ('query processing', 281), ('sensor networks', 272), ('large number', 263), ('case study', 243), ('proposed approach', 241), ('wide range', 238), ('model checking', 238), ('data structures', 236), ('sensor network', 226), ('experiments show', 223), ('ad hoc', 222), ('proposed method', 218), ('data structure', 215), ('simulation results', 212), ('machine learning', 212), ('web services', 209), ('power consumption', 206), ('previous work', 204), ('information retrieval', 202), ('paper describe', 202), ('execution time', 201), ('address problem', 196), ('results demonstrate', 192), ('based approach', 190), ('peer peer', 190), ('web pages', 188), ('operating system', 188), ('shared memory', 188), ('improve performance', 187), ('show proposed', 185), ('data streams', 185), ('source code', 183), ('data set', 183), ('wireless sensor networks', 179), ('software development', 177), ('recent years', 173), ('web search', 172), ('sensor nodes', 172), ('search engines', 170), ('paper introduce', 169), ('type system', 167), ('low level', 165), ('model based', 163), ('semantic web', 161), ('xml data', 160), ('software engineering', 159), ('experimental evaluation', 159), ('search engine', 158), ('control flow', 156), ('embedded systems', 155), ('high quality', 155), ('design implementation', 152), ('proposed algorithm', 149), ('information systems', 149), ('database systems', 148), ('paper study', 147), ('content based', 144), ('worst case', 144), ('software systems', 144), ('web applications', 140), ('approach based', 139), ('load balancing', 138), ('fine grained', 137), ('distributed systems', 137), ('present approach', 136), ('demonstrate effectiveness', 135), ('data sources', 134), ('paper show', 134), ('system performance', 134), ('paper introduces', 132), ('user interface', 132), ('polynomial time', 131), ('programming language', 131), ('high dimensional', 130), ('end end', 129), ('xml documents', 129), ('data stream', 128), ('algorithm based', 128), ('domain specific', 128), ('response time', 128), ('general purpose', 128), ('time series', 126), ('mobile devices', 125), ('data flow', 124), ('association rules', 124), ('feature selection', 120), ('case studies', 119), ('method based', 119), ('higher order', 115), ('virtual machine', 115), ('search results', 115), ('static analysis', 113), ('design space', 113), ('orders magnitude', 112), ('ad hoc networks', 112), ('support vector', 112), ('real life', 111), ('existing approaches', 110), ('paper address', 108), ('wide variety', 108), ('lower bound', 108), ('main memory', 108), ('nearest neighbor', 108), ('open source', 108), ('query language', 107), ('quality service', 107), ('web page', 106), ('extensive experiments', 105), ('small number', 105), ('fault tolerance', 105), ('image retrieval', 104), ('file system', 104), ('show approach', 104), ('query optimization', 104), ('compile time', 103), ('data model', 103), ('training data', 102), ('performance evaluation', 101), ('paper addresses', 101), ('paper investigate', 101), ('propose approach', 100), ('concurrency control', 100)]