Wicked Fast Latent Keyphrase Inference

Yu Lu (yulu3)
and
Xingyu Xiang (xxiang6)
and
Shidi Zhao (szhao41)
University of Illinois at Urbana-Champaign

For document representation, Latent Keyphrase Inference (LAKI) is developed to incorporate latent document keyphrases into document representation. The idea of representing document by its Domain Keyphrases brings LAKI very high accuracy in document classification. But however, since LAKI inferred the dependency between Domain Keyphrases and Content Units via a complex Bayesian network, it slows down the efficiency of LAKI in both training and inference process. To solve this problem, we propose WAKI (Wicked fast latent keyphrase inference) to do document keyphrase inference. In WAKI, instead of utilizing Bayesian network, we use a weight matrix to convert the bag of phrases representation into document keyphrase representation. Information on quality of phrase, semantic similarity and prior frequency of phrase are all incorporated into the weight matrix. The introduce of matrix operations improved the efficiency of LAKI by an order of magnitude in both training and inference phases while achieving higher accuracy on document classification. Moreover, since the weight matrix is sparse, the online inference time increase very slow as the number of total phrases grows. All of these make WAKI an efficient, scalable and effective architecture compared to LAKI.

Categories and Subject Descriptors:
General Terms: Latent Keyphrase Inference, Text Mining

Additional Key Words and Phrases: Phrase Mining, Document Representation, Text Mining, Segmentation, Word Embedding

1. INTRODUCTION

In natural language processing, because the raw text is unstructured, it is difficult to apply such as machine learning algorithms and data mining method to them. To be able to mining them, we need to convert raw text data into fixed length collections, more to vectors with numbers, this is called feature extraction. Thus for large set of document and text corpora data, our main goal is to represent them in some short collections of units so that we can apply tasks, like classification, and similarity study. To apply text mining, the starting point is to represent them with machine-understandable words and phrase. A document can be represented by words, topics, knowledge based concept, and document keyphrase. For example, the sentence DBSCAN is a method for clustering in process of knowledge discovery, can be represent in words: dbscan, method, clustering, process, in topics: [k-means, clustering, clusters, dbscan], in KB concept: data mining, clustering analysis, dbscan, and keyphrase: dbscan;[dbscan, density, clustering,…]. (see figure 1). With different represent styles, there are several document representation methods.

Bag-of-words, or BoW, is the traditional way for document representation. It can find known words in one specific document. These known words are called bag. Firstly, it split text and collects words form the corpus dataset. Using these features, it convert a specific document in to vectors, and decrease these sparse vectors with ignoring the frequency words, like of, the, ignoring punctuation, reducing words to their stem. We can score the vector by binary scoring method, or simply count the number of times each word appears in a document, and calculate their frequency as score, and get score matrix. However, this simple model bring some limitations. First, the known words bag need to be carefully selected, in addition, the sparsity cause the computational difficulty. Also, convert the raw text into numbers may loss a lot of information, the same word may represent different meaning, it is difficult for BoW to tell. And it needs known words collection, or knowledge based data first, if there are new words appear, BoW can not recognize it.

With these limitations, we move on the latent Dirichlet allocation (LDA), it is based on topic models instead of bag of words. [David M. Blei and Jordan 2003] From one document, we get several topics as clusters, and each cluster includes a bunch of words sharing the same topics. Each topic is a distribution over words; each document is a mixture of corpus-wide topics. LDA is a three-levels Bayesian model, which has corpus-level parameters, document-level variables, word-level variables. And for the connect of each layer, we calculated the condition possibility by Content Units with choosing topics. (see fig2). Documents are represented as latent topics graph, where each topic has a position and the possibilities going from one node towards another are calculate by a distribution over words. For LDA method, words are significantly related to topics, they calculate the fixed conditional distributions of words under specify topic. Topics are chosen by some random distribution, like multinomial distribution. However, because
it really depends on the topics you select, the number of topics is fixed and must be known ahead of time, and Dirichlet topic distribution cannot capture correlations. Thus Difficult for human to infer topic semantics, and it could not find the related or similarity between two documents.

Another model Explicit Semantic Analysis (ESA), can measure the relatedness of two texts in a concept space. It is designed to represent texts as a weighted mixture of a set of natural concepts, by concept level. Thus it is easier for human to understand. ESA is a concept-based model, is actually a variant of a generalized vector space model that uses Wikipedia as its index corpus. However, there are some limitation for ESA. First, because ESA is a variant of a generalized vector space model that uses knowledge based dataset like Wikipedia, as its index corpus, but the data is always large, thus it takes a lot of computational resource to produce concept vector, generate the index matrix. Also the large index matrix requires large memory. Second, if Wikipedia has a million documents, then the concept space has a million dimensions; similarity or relatedness computations between two vectors with numerous dimensions are costly.

Because Word2Vec embeds the words and can find the joint-meanings of phrases in a sentence. In Doc2Vec method, they use word embedding method in each document, during the learning step, they average the word embedding result and use that update result to represent the document. Comes to the detail methodology, Word2Vec proposed a neural network architecture of an input layer, a projection layer parameterized by the matrix $U$ and an output layer by $V$, use conditional possibility to calculate for each target word $w_i$ in a specific document. The word vectors are then learned to maximize the log likelihood of observing the target word at each position of the document. Doc2VecC consists of an input layer, a projection layer as well as an output layer to predict the target word.(see figure 5) ceremony in this example. The embeddings of neighboring words (opening, for, the) provide local context while the vector representation of the entire document (shown in grey) serves as the global context. In contrast to Paragraph Vectors, which directly learns a unique vector for each document, Doc2VecC represents each document as an average of the embeddings of words randomly sampled from the document (performance at position p, praised at position q, and brazil at position r). The problem for this method is it sometimes difficult to tell the meaning of each cluster. And it is necessary to set a proper number of clusters.

In 2016, there is a new document representation method called latent keyphrase inference(LAKI). There are two highlights of LAKI. The first one is that it use document keyphrase as the input. They firstly make a definition for how the document Domain Keyphrase looks like.

DEFINITION 1. Extract Domain Keyphrase $K \{K_1, K_2, ..., K_M \}$ from one document $D$, for each $K$, it contains a high-dimensional vector $P \{P_1(q), P_2(q), ..., P_M(q)\}$ for any text query $q$ from the same domain.

That is how they define the relation between query and keyphrase for one specific document. From the beginning, they generated the quality phrase by using Segphrase method. They extract the feature or words from a domain-specific corpus, then they related it to a specific document and it is called document keyphrase.

Another highlight for LAKI is using the Domain Keyphrase silhouettes graph in offline Domain Keyphrase learning step. They also make a definition for "Domain Keyphrase silhouette".

DEFINITION 2. for a Domain Keyphrase $K_m$, its keyphrase silhouette $S_m$ means link of all related content units to $K_m$, and these Content Units is $T \{T_1, T_2, ..., T_L \}$
LAKI method has two parts, offline part and online training part. Firstly use SegPhrase to generate quality phrase and then generate document keyphrase. Then create silhouette graph, Bayesian to calculate the possibility and then optimize it with EM method. For online training part, they fit in a document, then give different weight to phrase vector, more weight means closer, finally is able to represent document. They use conditional distribution in Bayesian network, to calculate the possibility from parent nodes to go to children nodes. And they added on noisy-OR term in summary of the possibility of parent nodes to increase the robust of the system. To update the weight of each link, they use EM algorithm to update the weight. During expectation step, they only count the grey nodes with single parent at each time. This reduces a lot of storage consumption and transforms our task to computing. (see grey nodes with single parent at each time, This reduces a lot of computation. Also the time of online inference is positive correlated to the length of corpus. Thus for large dataset, both offline and online training step take long time. Thus to improve the efficiency and accuracy of LAKI, we develop a method called wicked latent keyphrase inference (WAKI).

Fig. 5. Sketch of LAKI method.

Fig. 6. Sketch of EM algorithm.

In this section, we will present the overall architecture of WAKI. Before introducing the structure, we would like to first motivate our WAKI architecture to do the Domain Keyphrases representations.

WAKI is the first paper to propose using Domain Keyphrases for document representation. The idea of Domain Keyphrases representation helps WAKI achieve very impressive performance on both document classification. But however the Bayesian network approach use in LAKI brings high complexity to do model. Which in turn result in the low efficiency in both offline training and online inference. Thus, we remove the key component, Bayesian network in LAKI and replace it with a Content Unit Weight Matrix. Also, to improve the efficiency, instead of doing the Domain Keyphrases Silhouetting which adding extra burden in turning document into Domain Keyphrases representation, we directly find the silhouettes for Content Unit in the offline training. These changes would bring very high efficiency in Domain Keyphrases representation while preserving all the factors considered in LAKI.

The workflow of WAKI from training the raw text to infer the Document Keyphrases online is shown in figure 5. Given a domain-specific text corpus, the first step in WAKI is to mining the salient terms from the given corpus. But however, solving this task is not the objective of this work. So we will use an existing algorithm, Segphrase to identify the Domain Keyphrases and Content Unit in the corpus. Meanwhile, we expect to identify the quality of Content Units and Domain Keyphrases. This step can also be done by utilizing Segphrase. It would generate the quality score of each phrase during the training process, which will indicate the informativeness and distinctness of phrases. Then, Segphrase will be used to do the segmentation of each document of the corpus. After the segmentation, we can turn each document into the representation of Bag of Word. Based on this representation, we can calculate the prior frequency of each Content Unit. This will serve as an important building block for generating the Content Unit Weight Matrix. In order to define the similarity between Content Units and Domain Keyphrases and between Content Units themselves, word embedding needs to be done on Content Units and Domain Keyphrases. We finish this task by feed the segmentation result generated by Segphrase to Word2vec and turn all of the Content Units into vector representations. These vector representations will later be used to do Content Unit Silhouetting. i.e. For each Content Unit, finding a set of Domain Keyphrases related to the given Content Unit and use a vector to represent this Content Unit weight by its similarity between Domain Keyphrases.

After having all this building block, we would proceed to build a Content Unit Weight Matrix. This is the key technical contribution and innovation of WAKI architecture. Content Unit Weight Matrix is an \( m \times n \) matrix where \( m \) is the number of Content Units and \( n \) is the number of Domain Keyphrases. It is built based on the prior frequency of each Content Unit, phrase quality score of each Content Unit and silhouette of each Content Unit. The target of offline training in WAKI is to build this matrix. In the phase of online inference, whenever a document query comes in, all we need to do is to transform the document into the representation of bag-of-words and multiply it by this matrix to get the Document Keyphrases representation.
3. CONTENT UNIT EXTRACTION AND SILHOUETTING

3.1 Content Unit Extraction and Segmentation

The first step for offline training is to extract the Content Unit and Domain Keyphrases then do the segmentation on the training corpus, i.e., mining the salient terms in the given corpus. As we previously said, this is not the objective of this work. There are numerous works in discovering salient phrases, e.g., Segphrase and Autophrase. In order to make a fair comparison between the Bayesian model approach in LAKI and our Content Unit Weight Matrix, we adopt the same method adopted by LAKI, Segphrase, to finish this task.

More specifically, we would first use Segphrase to get the list of phrases appears in the corpus together with their quality scores. Then we would set a threshold $t$ on the quality score. For those quality scores higher than $t$, we treat it as Domain Keyphrases and use it for document representation. For those phrases with a quality score lower than $t$, we would treat it as Content Unit and use it as the building block of a document.

A segmentation would be done for each of the document in the training corpus. This segmentation will serve two purposes: cal-
If the following inequalities hold:

\[
\text{Content Unit weight Matrix.}
\]

**Inequity.**

vector representation \( V \) between Content Unit \( C \)

**between a bag of Domain Keyphrases which are similar in semantic to Content Unit** \( C \). The value of row \( S_i \) on each dimension of \( S \) refers to the similarity between Content Unit \( C \) and Domain Keyphrase \( D_i \). Then the value \( S_i \) of each appearance of the ith Content Unit in the documents contributes to the weight of each Domain Keyphrase. Since it appears frequently across various domains, it would also easily satisfy our second condition for filtering Content Unit silhouettes: conditional probability. Thus it may appear in many of the Content Unit silhouettes and contribute a lot to various Domain Keyphrases. This would make it much harder for us to distinguish documents from different domains with frequent "data" appearance. So we need to lower the weight of words like "data". They carry very low quantities of information. This can be reflected by its quality score done by many existing works.

Consider the factors above and combine it with the information we got from previous training process, we can finally give the formal definition of our Content Unit Matrix.

**Definition 3 Content Unit Silhouette.** Given a Content Unit \( C \), its silhouette \( S = S_1, S_2, ..., S_k \) is a vector that compromise a bag of Domain Keyphrases which are similar in semantic to \( C \). Denoting this bag of Domain Keyphrases as \( D = D_1, D_2, ..., D_k \). Then the value \( S_i \) on each dimension of \( S \) refers to the similarity between Content Unit \( C \) and Domain Keyphrase \( D_i \).

Based on the definition, the first step of Content Unit silhouetting is to find the corresponding bag of Domain Keyphrases. For a given Content Unit \( C \) and a Domain Keyphrase \( D \), and its corresponding vector representation \( V_C \) and \( V_D \). We take \( D \) into the silhouettes of \( C \) if the following inequalities hold:

\[
\cos(V_C, V_D) > t_v
\]

and

\[
\text{Pr}(V_D, V_C) > \text{Pr}_v
\]

Where \( t_v \) and \( \text{Pr}_v \) is the corresponding threshold set for each inequity.

Then for each \( T_i \) define in the definition, we set \( S_i \) to \( \cos(V_C, V_D) \) as the similarity value.

4. **Content Unit Weight Matrix**

In this section, we would present the methodology in building our Content Unit weight Matrix.

**Content Unit weight Matrix** is a \( m \times n \) matrix where \( m \) is the number of Content Units and \( n \) is the number of Domain Keyphrases. The value of row \( i \) and column \( j \) indicate how much of each appearance of the ith Content Unit in the documents contribute to the weight of the jth Domain Keyphrase. In order to correctly transform the bag-of-words representation into Domain Keyphrases representation, the following factor need to be taken into consideration.

(1) A document is more likely belongs to a key phrase if many of its phrases are close semantically to this key phrase. That's why we need to do Content Unit silhouetting. However, the vector representation of one Content Unit may close too many domain keyphrases. So in order to make our Domain Keyphrases more distinguishing, we add a very strict restriction on selecting Content Unit silhouetting. Domain keyphrases that do not fall into the silhouettes of a certain Content Unit will be considered has zero similarity to it. This strategy will help us generate a relatively sparse vector for Domain Keyphrases, which gives us more intuition while checking the correctness of WAKI and speed up the matrix operations.

(2) Different phrases have different normal frequencies in the document. e.g. Phrase "data" is very frequent across most of the documents and phrase "data science" is not. Assume that "data" appear around 100 times on average across all the documents. And "data science" appears once on average across all the documents. Then in a given document, it's totally different on the situation you see "data" twice and see "data science" twice. Because appear twice is much lower than average when it comes to "data" and much higher than average when it comes to "data science". So we calculate the prior frequency of each Content Unit and use as a negative factor in the weight of Content Unit Matrix.

(3) Different phrases process different quantities of information. e.g. Phrase "data" may appear frequently across various domain: database, data mining, natural language processing etc. However, "data" has a closer meaning to many of Domain Keyphrases. Since it appears frequently across various domains. So it would also easily satisfied our second condition for filtering Content Unit silhouettes: conditional probability. Thus it may appear in many of the Content Unit silhouettes and contribute a lot to various Domain Keyphrases. This would make it much harder for us to distinguish documents from different domains with frequent "data" appearance. So we need to lower the weight of words like "data". They carry very low quantities of information. This can be reflected by its quality score done by many existing works.

Consider the factors above and combine it with the information we got from previous training process, we can finally give the formal definition of our Content Unit Matrix.

**Definition 4. Given a set of Content Units** \( C = C_1, C_2, ..., C_m \) and a set of Domain Keyphrases \( D = D_1, D_2, ..., D_n \). Together with their vector representations \( V_{c_1}, V_{c_2}, ..., V_{c_m} \) and \( V_{d_1}, V_{d_2}, ..., V_{d_n} \), their quality score \( S_{c_1}, S_{c_2}, ..., S_{c_m} \) and \( S_{d_1}, S_{d_2}, ..., S_{d_n} \), and their prior frequency \( F_{c_1}, F_{c_2}, ..., F_{c_m} \) and \( F_{d_1}, F_{d_2}, ..., F_{d_n} \). A Content Unit Matrix \( M \) is a \( m \times n \) matrix with its elements \( M_{i,j} \) satisfied:

\[
M_{i,j} = \frac{\max(\sqrt{S_{c_i} \cdot 0.1}) \times \cos(V_{c_i}, V_{d_j})}{\log(F_{c_i} + 1) + 1}
\]

5. **Experimental Study**

5.1 Data sets

In order to evaluate efficiency and accuracy on the WAKI and make comparison with the LAKI, we run the experiments on an 8 vCPUs machine with 30 GB memory. The algorithms were implemented in Python and C++.

We conducted the experiments on two datasets. One is an academic dataset called Aminer, another dataset is Wine reviews. The multi-label classification task is performed for both datasets and both methods, equal number of documents are sampled for each label to avoid imbalanced class issue.

The following venues are predicted for the Aminer dataset:

- acm transactions on programming languages and systems (toplas)
The following origins are predicted for the Wine reviews dataset:

—US
—France
—South Africa
—Austria
—Argentina

5.2 Quantitative Evaluation

5.2.1 Experimental Setting. The evaluation metric follows the methods in the LAKI paper [Liu et al. 2016] and the document classification task is performed. It is critical to get rights choices of the features for the classification task and we expect that performing classification on the vectorial representation of document could reflect the power of WAKI. For the Aminer, we sampled 3000 documents in total, 83.3% of data is used for training while 16.7% of the data is used for testing. For the Wine reviews, we sampled 10000 documents in total, 70% of data is used for training whereas 30% of the data is used for testing. Please see Figure 8

5.2.2 Metric. We use SVM to conduct the document classification task. The purpose of the SVM is to find a hyperplane which could maximize the margin of the support vectors. [Cortes and Vapnik 1995] SVM is very efficient when the number of features are huge and has descent results in previous work. Thus, the SVM is used for the document classification task.

There are several parameters, such as kernel, regularization, and gamma, have been tuned when we conduct the experiments for training the classifier:

—Kernel: We used the linear kernel and radial basis function (rbf) kernel to perform the task.
—Regularization: This terms specify the extent of how much you want to avoid misclassification for each training sample. It also helps to avoid overfitting. In our experimental setting, the regularization term is 1. The following figure 9 illustrates the effect of different values for the regularization term.
—Gamma: This term specifies whether the points are far away from plausible line can have a big influence on the calculation. In our setting for the rdf kernel, the gamma equals 0.7. The following figure 10 illustrates the influence of the gamma.

5.2.3 Compared methods. LAKI: Latent Keyphrase Inference (LAKI) is a data-driven model that could represent documents. The way it represents the document is a vector of closely related domain keyphrases rather than single words or existing concepts in the knowledge base.

5.2.4 Efficiency comparison. We conducted the experiment on an 8 vCPUs machine with 30 GB memory and found our method is much more efficient than LAKI. During the offline training time, WAKI is 3 to 4 times faster compared with the LAKI and it also hundreds of times faster than LAKI during the serving time. This might due to LAKI uses a very complex Bayesian network and it takes much longer time for training and inference. Since our method has significant improvement in efficiency which indicates WAKI is promising for real world applications. Please see Figure 11

5.2.5 Prediction accuracy comparison. We used the SVM with several sets of parameters to perform the prediccion task. The highest accuracy for per dataset per method is reported as following. We tried SVM with linear kernel, SVM with rbf kernel and the regularization parameter for both kernel is 1. It is observed that the accuracy of our method slightly outperforms LAKI with amazing speed. Both of the LAKI and WAKI on the AMiner dataset

ACM Transactions on Graphics, Vol. , No. , Article , Publication date: .
achieved higher accuracy than Wine reviews and this indicts the underlying knowledge of wine review is less structured compared with Aminer which aligns the finding in LAKI paper. Please see Figure 12.

6. FUTURE WORK

—Scalability testing: We could scale up the data size for online training and the query size for the online training. Record the running time for different data size which can evaluate whether our approach could linearly scale up.

—Phrase relatedness: This is a metric mentioned in the LAKI paper [Liu et al. 2016] and the purpose of this metric is to examine how well the generated relatedness scores correlate with the gold scores for the phrase pairs. However, the gold score is based on human judgements and would take longer time to conduct this evaluation metric.

ACKNOWLEDGMENTS

We thank Jialu Liu and Jingbo Shang for advising our project. Also, we acknowledge the instructor and TAs of the CS512 course for their valuable feedback and suggestions to this project.
REFERENCES


