INFORMATION SYSTEMS IN MANAGEMENT

Information Systems in Management (2016) Vol. 5 (2) 205−214

A SEARCH OF SIGNIFICANT PHRASES FOR BUILDING TOPIC MODELS IN TEXT DOCUMENTS

PIOTR OŻDŻYŃSKI, DANUTA ZAKRZEWSKA

Institute of Information Technology, Lodz University of Technology

A huge amount of documents in the digitalized libraries requires efficient methods for exploring contained there information. "Topic modeling" is considered as one of the most effective among them. In spite of commonly used approaches for finding occurrences of single words, in the paper building topic models based on phrases is pondered. We propose a methodology, which enables to create a set of significant word sequences and thus limiting the search area to phrases which contain them. The methodology is evaluated on experiments performed on real text datasets. Obtained results are compared with those received by using LDA algorithm.

Keywords: topic model, frequent sequences, LDA

1. Introduction

As the amount of available data grows, the problem of managing the information becomes more crucial. The digitized libraries cover not only hundreds of years of human knowledge firstly written on paper, but also being constantly published over the internet documents such as articles, blogs or opinions. Nowadays there have been arising huge collections, which effective exploring and browsing require structured ways of interaction.

As one of the recently developed approaches there should be mentioned topic modeling, where each document can be labeled with topic names. The method consists in mining the collections through the underlying and constantly reappearing

topics. Such approach can be used for searching documents similar to those of interest, what plays important role in information retrieval tasks.

Most topic modeling algorithms use text corpora as a "bag of words" and rely only on occurrences of single words. As human interpretation of a text is based on the recognition of the meaning of phrases rather than on the separated words, current topic modeling methods use phrases instead of words to build a model.

In the paper a set of algorithms which allows to find significant word sequences is proposed. The basic assumption of the considered approach is ability to find the most informative sequences and omitting meaningless phrases from an analyzed set of text documents. Having the set of selected phrases together with documents in which they occurred would allow to find topics by analyzing only these phrases instead of the whole document. What is more, topics can be assigned in hierarchical order according to its significance in the document.

The key idea of the presented algorithms consists in considering a graph structure which facilitates the analysis of phrases and connections between them. Using the graph makes possible to select sequences of words which are characteristics for different topics and lets to tie their source documents.

The remainder of the paper is organized as follows. In the next section the topic modeling research is described. Then the proposed methodology including algorithms of frequent sequences searching and grouping similar documents is depicted. In the following section results of experiments carried out on real document collections are discussed. Finally some concluding remarks and future research are presented.

2. Topic modeling

The goal of topic modeling is to automatically discover the topics from a collection of documents. The documents themselves are observed, while the per-document topic distributions, and the per-document per-word topic assignments are hidden structure. The central computational problem for topic modeling is to use the observed documents to infer the hidden topic structure. This can be thought of as "reversing" the generative process.

Topic models can be interpreted as probability distributions on terms. This approach is presented in [1] where authors treat a set of documents as a combination of terms from the selected universe. A corpus model C is a result of the imposition of terms U, topics T, styles S and a probability distribution D.

$$C = (U, T, S, D) \tag{1}$$

Papadimitriou et al. [1] assumed that documents are not represented by terms but by the underlying (latent, hidden) concepts referred to by the terms. They pro-

posed the information retrieval method, known as *Latent Semantic Indexing*, which aims at capturing hidden document structure by using techniques from linear algebra.

Blei et al. proposed *Latent Dirichlet Allocation* (LDA) method [2, 3] based on a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. Formally a topic is defined to be a distribution over fixed vocabulary. Authors assume that topics are specified before any data has been generated. Then for each document in the collection words are obtained in two stage process:

1. the distribution over topics is randomly chosen;
2. (a) the topic is randomly chosen and (b) words from the corresponding distribution over the vocabulary are being selected.

This statistical model reflects the intuition that documents represent multiple topics. Each document exhibits the topics with different proportion (step #1); each word in each document is drawn from one of the topics (step #2b), where the selected topic is chosen from the per-document distribution over topics (step #2a). The idea of LDA method is presented in Figure 1.

Another strategy for topic modeling is considered in [4], where a framework named KERT has been presented. The technique is based on finding phrases instead of single words. To find topical keywords LDA method is used. Next frequent sequences are generated using an efficient pattern mining algorithm FP-growth [5]. Candidate topical keyphrases are selected from those which contain topical keywords. Then the phrase qualities are evaluated using a characteristic function, and ranked accordingly. Top ranked phrases are selected as a representation of the topic.

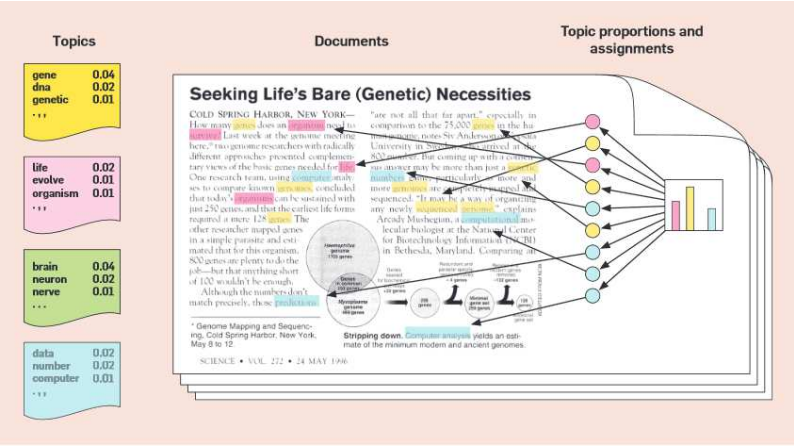

Figure 1. Latent Dirichlet Allocation (LDA) [3]

The similar approach is used in a method called *Scalable Topical Phrase Mining from Text Corpora* [6]. The technique is also based on the frequent sequences The main difference comparing to the KERT method consists in generating frequent sequences before using the modified LDA method. The last one is applied to find topical keyphrases. This modification is called PhraseLDA.

3. Methodology

In the current research we consider the method based on the similarity of documents represented as vectors of sequences. The proposed technique consists of the two following steps:

- finding most frequent sequences;
- finding documents most similar to the selected one.

Both of them are described in the following subsections.

3.1. Finding most frequent sequences

As a word sequence we will consider an ordered list of consecutive words. Sequences A and B are equal if they have the same length and in both sequences the same words are at the same positions. The length of the sequence is the number of words within.

Definition. A sequence B = (b_1, b_2, b_3, ..., b_m), m = | B | is contained in the sequence A = (a_1, a_2, a_3, ..., a_k), k = | A |, (B is a subsequence of A) if there exists a number d such that

$$0 \le d \le k - m \wedge \forall i \in \{1,..., m\} : b_i = a_{i+d} . \quad (2)$$

The sequence consisting of n words will be referred to the name of the n-gram. In particular, the bigram and trigram will mean the sequence of the two and three words. In this paper the word sequence will be used interchangeably with the word phrase.

In the algorithm, new frequent sequence of length $n + 1$ is built on the basis of the existing sequence of length n and an information about bigrams location. This approach is derived from the observation that if a sequence of length $n + 1$ is frequent, all subsequences of the sequence are also frequent. It is used in the algorithm “apriori” [7], which is a reference for many other searching frequent patterns algorithms. Thus, searching for frequent sequences of length $n + 1$ we assume that they consist of frequent sequences of length n.

Firstly, a set of analyzed documents is normalized. At this stage punctuation marks and numbers are removed, all the uppercases are converted to lowercases.

In the first step of the algorithm a unique identifier is assigned to each word in the whole set. The identifier is an integer number. Thus, a collection of text docu-

ments is converted into a set of numerical sequences. Each next occurrence of the word is replaced with the identifier given to the first occurrence.

In the second step of the algorithm, there is built a data structure, which stores all pairs of consecutive words as well as additional information about their positions. The occurrence of each pair is associated with a specific document and positions of pairs being an offset from the beginning of the document. To simplify the notation both identifiers are stored as a single integer. Using binary notation older bytes store the document index while younger ones store a bigram position. The structure created this way is kept as a map. The key consists of a pair of numbers connected with the bigram. As all the bigrams are indexed the structure is called the inverted bigram index. Since longer *n*-grams will be represented by the same data structure it is required to remember also a sequence length.

This method of storing bigrams positions is equivalent to storing document as a sequence of words. Both forms can be converted without loss of information. It is also possible to combine the first and the second step to produce the word's indexes and bigram indexes in one pass.

To indicate only frequent sequences support threshold should be input as a parameter. Further steps of the algorithm will be performed for only these *n*-grams for which the number of occurrences is greater or equal to the threshold. Thus it is practical to sort bigram keys in a descending order.

The bigram's location number of keys will depend on the data contained in the text. The number of bigrams can be equal to the square of the number of unique words in a set of documents. However, if all the words are unique the number of possible bigram's positions cannot be greater than the size of the document set. Let us assume that the set consists of k words and each is unique, then the maximum number of bigrams is less than k because each bigram has its location in the document.

The starting set of frequent sequences is the set of bigrams so $n = 2$. For each sequence of length n (denoted by $Q_i(n)$) a list of candidate sequences of length $n + 1$ ($Q_i(n + 1)$) is created. All n-grams whose first $n - 1$ words are the same as the last $n - 1$ words of the starting n-gram are searched. Since this operation will be repeated many times it is reasonable to hold such a map of n-grams in a memory. The key of such a map would be the $(n - 1)$-gram and the value would be an array of n-grams starting with this sequence [8].

The list of potential sequences of the length greater by one is formed by joining two n-grams with the same subsequence of the length of $n - 1$. The new sequence has to be more frequent than the specified support threshold. It is therefore necessary to count the number of times the sequence occurs in the text. There is no need to search all the set of documents. It is enough to compare an array of positions of the sequence $Q_i(n)$ and the position of the n-gram that expands this sequence. If a candidate sequence $Q_i(n + 1)$ occurs in the text, the position of the ending is by one greater than the index of the starting n-gram. The example of such relationship is illustrated in Figure 2.

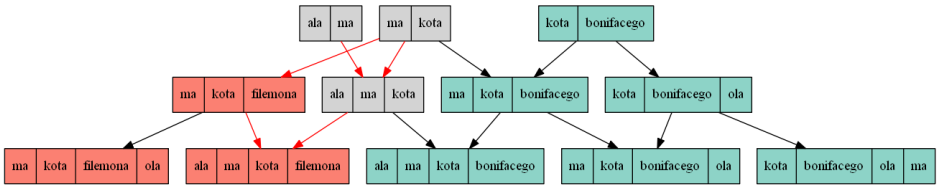

Figure 2. A structure of connections between joined *n*-grams

After completion of the cycle of the algorithm for the next *n* all data is stored in the structure like the one presented in Figure 2. Each *n*-gram is defined by specifying the array of starting indices, the sequence length, the first and the last word. Moreover each *n*-gram has references to the starting and the ending subsequence of the length $n - 1$. At the same time the references to longer sequences are remembered in the data structure. This graph-like structure allows to analyse the links between documents and sequences as well as the links between sequences of different lengths.

```
algorithm to convert text to numbers from a document set is
input: a set of normalized documents (DS)
output: words to numbers dictionary,
        an array of arrays of numbers (AN)
for each document (Di) in DS do
AN: array <- add Di as the array of numbers

algorithm to create data structures is
input: an array of arrays of numbers (AN)
output: pairs to location arrays map for bigrams
for each array ANi in AN do
  for each pair <wj, wj+1> in AN[i] do
    if the pair not exists then
      create a pair key and an array of locations
    add location (i, j) to an array related to the current pair key

algorithm to growth sequences is
    input: the map of n-grams
    output: the map of (n+1)-grams
for each n-gram Ng(left) do
  for each n-gram Ng(right) do
    if Ng(right) starts with last n-1 words of Ng(left) then
      (compare location arrays)
        if Ng(right) is consecutive to Ng(left) then
          add (n+1)-gram to an result set
```

Figure 3. A frequent sequences searching algorithm

The searching process ends if there cannot be found any longer frequent sequences. Recreating of the sequence of words is possible by reading from a document or from a reverse bigram index.

From among the set of sequences only the longest ones are chosen. They are used as the document representation, which will be applied in the next step for grouping documents. A frequent sequences searching algorithm is presented in Fig. 3.

3.2. Finding groups of similar documents

After building the *n*-gram structure all the documents are represented as sets of sequences of different lengths. The document with the greatest number of sequences is chosen. Then the Hamming distances [9] between the document and all the others are calculated. The expected number of resulting groups determines the length of a subgroup. Documents are ordered according to the distances between them and when the distance gets closer to the expected size of the subset, the group is formed. The steps are repeated until the input set is all divided into subsets. An algorithm for grouping similar documents is presented in Fig. 4.

```
algorithm convert a document to a list of n-grams is
    input: a set of n-grams with references to documents
    output: a set of documents represented only by n-grams
for each n-gram (Ng) do
  for each document (doc) containing Ng do
     doc<- add Ng to list

algorithm to group documents containing a similar set of sequences is
     input: a document array, each as a vector of n-grams,
            W approximate number of documents in an output group
     output: a set of groups of similar documents
while input array is not empty do
  select one document D(first)
  find 2xW documents with lower Hamming distance from D(first)
  select the threshold to cut the number of documents nearest W
  create a group
  add this group to output set
  remove each document from the input array

algorithm to select topic phrases is
    input: a list of sets of phrases
           K number of phrases in each set
    output a list of sets of topic phrases
for each set of phrases do
  find K most frequent phrase
  remove other phrases from this set
```

Figure 4. An algorithm for grouping similar documents

Finally, an analysis of phrases in each document group is done. The most frequent phrases are indicated as topic models.

4. Results

To evaluate the proposed methodology experiments on real document datasets were carried out. Two collections of about 20000 documents were considered:

- Ohsumed collection, which includes medical abstracts from *Medical Subject Headings* categories of the year 1991 ([10]);
- 20Newsgroups corpus, which contains 19997 articles for 20 categories taken from the Usenet newsgroups collection ([11]).

Table 1. Results of presented method and *LDA* algorithm

Proposed method	LDA (Mallet 2.0.7) ([8])
half, injury, without, *acquired immunodeficiency* syndrome, factor, objective, light, electron microscopy, analysis, still, diagnosis, small, previously, *aids*, bacterial, therefore, routine, samples, *clinical*, organism, measurements main results, complex, matched, *positive*, organisms, seen, *human immunodeficiency virus hiv infection*, showed, endoscopic, independent	*infection hiv virus human aids immunodeficiency* infected type related *acquired* infections disease *positive* viral htlv dna cases hpv *clinical*
minutes, importance, *animals*, injury, *rat*, low, *rats*, acute, levels, macrophages, vivo, observations, failure, within, elevation, h, *effect*, p less, mechanism, vitro, also, fold, *activity*, blood, plasma, mechanisms, accumulation, tnf, pathogenesis, significantly	induced *rats effect* model *animals* mice *activity* response increase effects platelet *rat* factor experimental role control increased release tissue
clinical, due, induced, year old man, either, connective *tissue*, authors, adenocarcinoma, together, d, rare, staining, examination, intact, describe, tumor cells, lower, phenomenon, immunohistochemical, means, distribution, secondary, sections, hypercalcemia, *results*, tumor, certain, levels, also, seen	bone *results tissue* anterior laser eyes posterior degrees retinal visual fractures technique knee average fracture hip cervical total spine
pathogenesis, similar, healthy, obtained, studied, response, h, healthy controls, i e, markedly, mechanism, thus, peripheral blood, evidence, circulating, known, several, alpha, median, monitoring, myasthenia gravis, among, found, infected, ml, data, prophylaxis, activation, healthy subjects, peak	injury cerebral brain nerve trauma system central spinal injuries multiple cord motor neurologic loss nervous function potential normal recovery

The proposed method and *LDA* implementation taken from Mallet 2.0.7 framework [8] were used to find topics in each corpus. Parameters were set to archive 20 topic groups. Comparison of the obtained results were done by qualitative analysis. In Table 1 some topics generated by both methods were presented. Each word appearing in both results is in Italic. From among 20 groups there are 2 for Ohsumed collection and 3 for 20 newsgroups documents which have disjoined results. It means that generated sequences contain no word from a Mallet result words set.

The main advantage of the sequences over words is the ability to archive the proper order of words as a result. Some sequences can be not so obvious to easily combine them from single words. For example words *immunodeficiency* and *acquired* forms a sequence *acquired immunodeficiency syndrome* and *human immunodeficiency virus hiv infection*.

5. Conclusion

In the paper a framework for topical keyphrase generation is introduced. The algorithm is based on analyzing a graph-like structure which holds information about relations between frequent sequences and their positions in the set of documents. Filtering only the most significant sequences reduces the data size. In spite of the most topic modeling techniques the proposed method does not use *LDA* algorithm. Such approach allows to reduce computational complexity of the algorithm.

Qualitative analysis showed that the obtained results differ from the ones received by using *LDA*, however in most of the cases the similarity between groups seems to be significant.

Future research will consist in further exploring the *n*-gram graph to extract more significant sequences and to reduce the ones that may be considered as a noise. The different choice of distance measures is planned to be considered as well as broader range of datasets for testing the method.

REFERENCES

[1] Papadimitriou C., Raghavan P., Tamaki H.; Vempala S. (2000) *Latent Semantic Indexing: A probabilistic analysis*, Journal of Computer and System Sciences, Vol. 61 (2), 217–235

[2] Blei D., Ng A, Jordan M. (2003) *Latent Dirichlet allocation*, Journal of Machine Learning Research, 3, 993–1022

[3] Blei D. (2012) *Probabilistic topic models*, Communications of the ACM, 55 (4), 77–84

[4] Danilevsky M., Wang C., Desai N.,, Ren X., Guo J., Han J. (2014) *Automatic Construction and Ranking of Topical Keyphrases on Collections of Short Documents*, SDM'14

[5] Han J., Pei J., Yin Y., Mao R. (2004) *Mining frequent patterns without candidate generation: A frequent-pattern tree approach*, Data Min. Knowl. Discov., 8 (1), 53–87

[6] El-Kishky A., Song Y., Wang C., Voss C., Han J. (2014) *Scalable Topical Phrase Mining from Text Corpora*, Proceedings of the VLDB Endowment, Vol. 8 (3), 305–316

[7] Agrawal R., Srikant R. (1995) *Fast algorithms for mining association rules in large databases*, In Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[8] Machine Learning for Language Toolkit http://mallet.cs.umass.edu/

[9] Hamming R.W. (1950) *Error detecting and error correcting codes*, The Bell System Technical Journal, Vol. 29 (2)

[10] ftp://medir.ohsu.edu/pub/ohsumed

[11] http://www.ai.mit.edu/people/jrennie/20Newsgroups/