Unsupervised Text Segmentation Based on Latent Dirichlet Allocation and Topic Tiling

Mirela Oštrek, Luka Dulčić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{mirela.ostrek,luka.dulcic}@fer.hr

Abstract

In this paper we describe an unsupervised method for topical segmentation of text. This method represents text as a sequence of semantically coherent segments using the Bayesian topic modeling approach and one of the recently developed text segmentation algorithms. We developed and evaluated this method on synthetic Choi dataset. After the initial dataset cleanup, Latent Dirichlet Allocation model is applied to it in order to predict a certain probability distribution over topics which are then used as topic vectors for the Topic Tiling algorithm. The latter divides text in semantically coherent segments by calculating cosine similarities and depth scores between the neighboring topic vectors. Performance of this method is evaluated with both Pk and *WD* measure, and results are shown.

1. Introduction

Most people today are searching through digital repositories which contain a great number of documents such as web pages, articles, emails, forum and blog posts, and so on. Despite this need for information, finding a relevant topic for a query is extremely difficult task, unless documents are manually annotated with topics (Alpaydin, 2014). Our aim is to do this annotation automatically and use it to divide a number of documents into semantically coherent units – paragraphs. This will prove to be helpful later on, when a search engine tries to retrieve all relevant documents and to pinpoint a specific location in document which best suits user's query.

Dividing documents into semantically coherent units is commonly known as Text Segmentation (TS). TS can be divided into two sub–fields: linear TS and hierarchical TS. Linear TS sequentially analyses text and segments it based on sequential topical changes, while hierarchical TS analyses text as a whole and tries to discover topical hierarchy. Early unsupervised linear TS algorithm was *TextTiling* (Hearst, 1994). In *TextTiling*, text is divided into blocks of words which are then represented as term frequency vectors. Text is segmented based on a cosine similarity between adjacent word blocks. Locations of segments boundaries are determined by using heuristic. (Galley et al., 2003) introduced *LcSeg*, *TextTiling* based algorithm, which used tf–idf term weights for representing word blocks instead of tf representation, this approach improved results over *TextTiling*. One of the first probabilistic approaches were introduced by (Utiyama and Isahara, 2001) using Dynamic Programming. Most recent linear TS approaches use topic models based on Latent Dirichlet Allocation (LDA) (Misra et al., 2009; Riedl and Biemann, 2012c). First hierarchical TS algorithm was introduced by (Yaari, 1997), his approach is based on a cosine similarity and agglomerative clustering. (Choi, 2000) introduced C99 algorithm which, instead of computing a similarity between adjacent word blocks, computes a similarity between all word blocks and searches for a segmentation which optimizes an objective based on these similarities by greedily making a succession of best segment boundaries. More recently, first Bayesian hierarchical TS algorithm based on LDA was introduced by (Eisenstein, 2009).

In this paper, we focus on linear TS based on the work of Riedl and Biemann (2012a; 2012b; 2012c; 2012d). We use LDA model to predict a probability distribution over topics and Topic Tiling algorithm to segment given text into paragraphs. In Section 3 we give a short overview of LDA and continue by explaining Topic Tiling in Section 4. Dataset preprocessing is illustrated in Section 2. In Section 5 we present evaluation metrics and our results. Section 6 concludes the paper.

2. Dataset

2.1. Choi Dataset description

The Choi dataset (Choi, 2000) is commonly used in TS field. It is artificially generated from Brown corpus and consists of 920 documents. Each document consist of 10 segments. Document generation was performed by extracting snippets of 3–15 sentences from different documents from Brown corpus. Documents in Choi dataset are divided into 6 categories based on the number of sentences in segments. Categories are 3–5, 3–11, 3–15, 6–8, 9–11 and 12–15 where A–B indicates there are A to B sentences in each segment.

2.2. Preprocessing

Preprocessing of Choi dataset consists of standard tasks such as sentence segmentation, tokenization and stemming. Sentence segmentation was an easy task because every sentence in Choi dataset is divided by a new line character. Additionally we needed to remove irrelevant sentences which were empty or consisted of only stop words and other irrelevant tokens. Task of sentence segmentation is essential for this project because the Topic Tiling algorithm predicts segment boundaries based on similarities between adjacent sentences. Tokenization of sentences is done using standard *nltk*[1] tokenizer. Obtained tokens are then filtered using

[1] nltk.org

the list of irrelevant tokens which includes standard english stop words list. After filtering, tokens are stemmed using *nltk* Porter Stemmer, and finally we vectorize the tokens using *scikit–learn*[2] CountVectorizer.

Initial preprocessing pipeline described above did not produce the desired results. In further analysis of Choi dataset and segmentation model we discovered that Choi dataset is polluted with lots of irrelevant tokens which were not covered in our list of irrelevant tokens. These tokens appear in the vast majority of documents and were the main reason for the poor performance of the segmentation model. After identifying all irrelevant tokens which affected the performance, we included them in list of irrelevant tokens. Also, after reading *scikit–learn* tutorial[3] on this subject, we decided to also remove all digits and tokens which appeared in more than 95% of documents or appeared in only one document. Previously described changes significantly improved the performance.

3. LDA

LDA in text processing is an application of the Bayesian approach, namely topic modeling (Blei et al., 2003; Alpaydin, 2014). It serves its purpose for our method as it outputs the probability distribution over topics for each sentence of a given text. Topic distribution is assumed to have Dirichlet prior as in:

$$\mathrm{Dirichlet}(\theta|\alpha) = \frac{\Gamma(\alpha_0)}{\prod_i^K \Gamma(\alpha_i)} \prod_i^K \theta_i^{\alpha_i - 1}, \qquad (1)$$

where $\theta = [\theta_1, \ldots, \theta_K]^T$ and $\alpha_0 = \sum_i \alpha_i$. In equation (1) K is the number of topics and θ denotes probabilities that correspond to the proportions of different topics. Prior allows us to calculate our prior beliefs in these proportions.

LDA is parametric model and its size is fixed, but we can make this model nonparametric by making the number of topics increase as necessary and adapt them to data using a Dirichlet process (Alpaydin, 2014). However, in our project we did not automatically adjust the number of topics to data. We have chosen several fixed numbers of topics and evaluated our method against them to see which one gives the most satisfying results.

LDA model in general works in the following way:

1. Generate topics in advance.
2. Assign topic to each word.
3. Check up and update topic assignments iteratively.

LDA iterates over the third step defined number of times. This is one of the parameters for fine–tuning LDA model. Other than that, LDA model has three more parameters and those are: α , β and number of topics K. Parameter α regulates the sparseness of topic–document distribution. Lower values result in documents being represented by fewer topics. Reducing β increases the sparsity of topics, by assigning fewer terms to each topic, which is correlated to how related words need to be, to be assigned to a topic (Riedl and Biemann, 2012c).

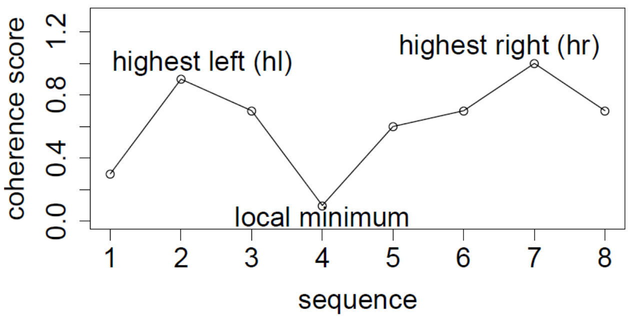


Figure 1: Illustration of the highest left and the highest right peak according to a local minimum (Riedl and Biemann, 2012c).

We conducted a research in three different ways of training our LDA model regarding semantically coherent units which are given to LDA as inputs. It is possible to send units such as sentences, paragraphs or even whole documents to LDA, all with a goal to fit our model so that it can perform well on the set of unseen documents. Inputs for predictions are always sentences because we need to obtain word–topic vectors, which are represented as probability distributions over topics for each sentence, and pass them on to the algorithm for TS.

4. Topic Tiling

With the aim of being able to segment the given textual document into semantically coherent units, we applied the Topic Tiling algorithm to it. In the previous section we have discussed LDA model and how it outputs probability distribution over topics for each sentence of a given text. Those outputs are actually called topic vectors and they serve their purpose as inputs for the Topic Tiling algorithm. In contrast to some older algorithms for TS such as *TextTiling*, Topic Tiling algorithm does not use real words, but it uses topic distribution over words in a sentence.

In the Topic Tiling algorithm, each sentence of a text is represented by a topic vector. Neighboring topic vectors are then compared in terms of similarity. We decided to use cosine similarity because it is efficient for our task and it is not computationally too expensive. Cosine similarity between two topic vectors is called the coherence score. Values close to zero indicate marginal relatedness of topic vectors, while values close to one denote considerable relatedness. The next step is calculating the depth scores d_p with the following expression:

$$\mathrm{d_p} = \frac{1}{2}(hl(p) - c_p + hr(p) - c_p), \qquad (2)$$

where $hl(p)$ denotes the highest peak on the left side of the current depth score point p, $hr(p)$ denotes the highest peak on the right side of the p, and c_p is coherence score for p (Riedl and Biemann, 2012c). Figure 2 shows highest left and right peak for a certain local minimum. Finally, depth scores are searched for M local minimums, and boundaries

[2] `scikit-learn.org`

[3] `scikit-learn.org/stable/auto_examples/applications/topics_extraction_with_nmf_lda.html`

are set between topically different segments of text – paragraphs. Number of paragraphs M can be chosen in two different ways:

1. Number is fixed according to developer's "intuition" or already known fact.

2. Number is "calculated" according to certain condition performed on depth scores.

First case is commonly used if we already know how many thematic paragraphs textual document should consist. Second case counts all depth scores which meet the following condition as boundaries:

$$\mathrm{d_p} > \mu - x \cdot \sigma. \tag{3}$$

Condition (3) must be met for depth score to be marked as boundary between two paragraphs. For that specific purpose, standard mean μ and standard deviation σ are calculated using depth scores as data. In that way, a threshold for depth score being marked as a boundary is defined and number of paragraphs is selected dynamically. In condition (3) there is also x variable present. In our research, we have chosen a few values for x ranging roughly from 0 to 1 to see which is the optimal value of x for dynamical segmentation. Results are shown in the next section.

5. Evaluation

5.1. Pk and *WD* Measure

Early articles on TS (Hearst, 1994) used precision and recall for evaluating segmentation models. These methods are nowadays considered inappropriate for this task because the distance between the false positive segment boundary and the correct one is not considered at all. For this reason, Pk (Beeferman et al., 1999) and *WD* (Pevzner and Hearst, 2002) measures were developed. Descriptions of Pk and *WD* measures below are presented following the description in (Riedl and Biemann, 2012c).

Pk measure uses a sliding window of k tokens which is moved over the text to calculate segmentation penalties. First we generate pairs: $(1, k), (2, k+1), (3, k+2), ..., (n-k, n)$ where n is the length of the document. Then for each pair (i, j) it is checked whether positions i and j belong to the same segment or not. This is done separately for estimated and gold standard boundaries. If the gold standard and estimated segments do not match, a penalty of 1 is added. Finally, an error score is computed by normalizing penalty score by the number of pairs $(n - k)$, which produces a number between 0 and 1. A score of 0 indicates a perfect match between a gold standard and estimated boundaries. The value of parameter k is usually set to a half of the average segment length, given by the gold standard.

WD measure, according to its authors Pevzner and Hearst (2002), is an enhancement over Pk measure, where the drawback of Pk is that it is unaware of the number of segments between pairs (i, j). Pk and *WD* measures are very similar, both use sliding window of k tokens. The first step in *WD* is the same as in Pk; dividing document into $(n-k)$ pairs. Then, for each pair (i, j), number of segments between position i and j is counted separately for gold standard and estimated segments. If the count for the gold standard and estimated segments does not match, a penalty of 1 is added. Finally, error score is obtained by normalizing penalty score by the number of pairs $(n - k)$ which produces number between 0 and 1. A score of 0 indicates a perfect match between the gold standard and estimated boundaries.

In practice, Pk and *WD* measures are highly correlated. For this reason, we used only Pk measure for validation of model to reduce the overhead of the parameter grid search, but used both measures for model testing.

5.2. Evaluation Setup

First of all, it is important to note that we were unable to conduct a proper cross–validation which is usually mandatory for methods with a lot of parameters like this one. Main reason for this is that the training of LDA model is very computationally demanding, and since we had no high computation hardware available, this was a considerable limitation. Because of the latter reasons we were forced to reduce parameters grid search space and to even neglect some parameters for which we used recommended values (Riedl and Biemann, 2012c). Cross–validation was performed with simple train–validation–test split of dataset with ratio 60:20:20. We were unable to use k–fold cross validation because of its high computational demand.

Following parameters are subject to optimization in this segmentation model:

- K : Number of topics used in the LDA model. Commonly used values vary from 50 to 500.
- α : Document topic prior in LDA model. Recommended value is 0.1.
- β : Word topic prior in LDA model. Recommended values are 0.1 or 0.01.
- i : Inference iterations in LDA model. Recommended value is 70 to 120.
- x : Multiplier of standard deviation in (3). Commonly used value is 0.5.

Parameters β and i were not part of the grid search. Recommended values $\beta = 0.01$ and $i = 70$ were used. The grid search included parameters K, α and x with the following values:

- K: $\{60, 80, 100, 120, 130, 140, 150, 160\}$.
- α : $[0.1, 1]$ with step of 0.1.
- x : $[0.1, 0.7]$ with step of 0.1.

Beside these parameters, we can also consider LDA input as another parameter. We can input documents, segments or sentences into LDA as mentioned in the article above. This this parameter was not included in a grid search because it would extend grid parameter space considerably. Instead, we estimated performance on a smaller subset of

Table 1: Segmentation results with different LDA input types. Segmentation model was trained on 100 documents and tested on 50. Results may slightly vary due to the stochastic nature of LDA and dataset split, but in general, segment as basic input unit always outperforms the other two.

Input type	Pk	*WD*
sentence	0.46	0.53
segment	**0.17**	**0.22**
document	0.36	0.41

Table 2: Segmentation results on validation set. For simplicity only scores regarding parameter x are shown with respect of optimal parameters K and α.

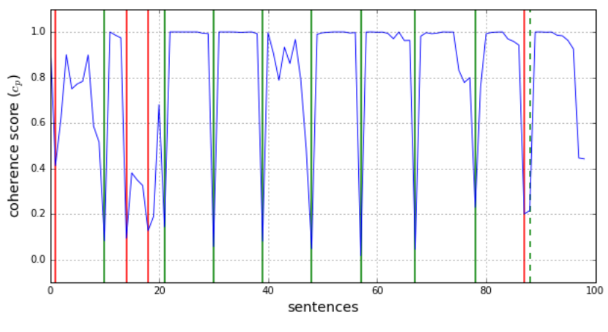
K=150, α=0.1	Pk
x = 0.1	0.093
x = 0.2	**0.087**
x = 0.3	0.095
x = 0.4	0.104
x = 0.5	0.108
x = 0.6	0.122
x = 0.7	0.143

documents with fixed parameters (K=100, α=0.1, β=0.01, i=70, x=0.5). Input consisting of segments as basic units significantly outperformed other two input types, therefore we used segments as the basic input units for model evaluation. Table 1 shows scores for different input types. Table 2 shows segmentation results on the validation set.

5.3. Results

Cross–validation showed that the optimal parameters are: K=150, α=0.1 and x=0.2. Pk measure score for optimal model on the test set is 0.09, and the *WD* measure score is 0.1. These results might be suboptimal because of the limitations we have faced when conducting model selection. Also these results may vary ±0.02 because of the stochastic nature of LDA model and datasets split. Figure 2 shows segmentation of one randomly chosen document from test set.

Results of our segmentation model could be improved by extending a parameter grid search and by including neglected parameters. Also, splitting segments based on a coherence score of only adjacent sentences does not always devise desired results. From more detailed analysis of some false positive boundaries which were very close to true positive ones, it is clear that the model lacks a little more context to correctly decide where to put boundary. In other words, comparing just one sentence on the left and one on the right side was not enough to make the correct decision. For this reason, we should devise topic vectors for block of sentences, not only one sentence. This would give the model more knowledge when deciding where to put segment boundary, but it would introduce window parameter ω denoting the number of sentences in the block.



```
Pk 0.094737
WD 0.147368
```

Figure 2: Plot of document segmentation. The blue graph shows cosine similarities between sentences, solid green lines indicate correct boundaries, solid red lines indicate false boundaries and dashed green lines indicate false negative boundaries. x–axis indicate between which sentences is coherence score plotted, eg. value 0 indicate coherence score between sentences in document at index 0 and 1, value 1 between sentences 1 and 2, and so on.

6. Conclusion

Throughout this article we have dealt with one of the unsupervised methods for TS which is based on the LDA topic model and Topic Tiling algorithm. In the previous section, we have carefully examined obtained results of our method on Choi dataset. The performance that ensued from cross–validation process was fairly satisfying, so we deem this state of the art method to be worthy of its label, but we also imply the need to test it out on sets of real world textual documents. By doing that it would be possible to make proper parameter adjustments and gain new clues on possible correlations between them. For further work, we would like to experiment with topic vectors, analyze it more in depth and try to gain more understanding of how they are generated. Also, it would be very interesting to generate sentence vectors with new methods like *word2vec* which achieve very promising results in a variety of natural language processing areas recently.

References

E. Alpaydin, 2014. *Introduction to Machine Learning*, pages 449–450, 480–482. MIT Press.

Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical models for text segmentation. *Machine learning*, 34(1-3):177–210.

D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, pages 993–1022.

F. Y. Y. Choi. 2000. Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 26–33, Seattle, WA, USA.

Jacob Eisenstein. 2009. Hierarchical text segmentation from multi-scale lexical cohesion. In *Proceedings of Hu-*

man Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 353–361. Association for Computational Linguistics.

Michel Galley, Kathleen McKeown, Eric Fosler-Lussier, and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 562–569. Association for Computational Linguistics.

M. A. Hearst. 1994. Multi-paragraph segmentation of expository text. *In Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 9–16.

Hemant Misra, François Yvon, Joemon M Jose, and Olivier Cappe. 2009. Text segmentation via topic modeling: an analytical study. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1553–1556. ACM.

L. Pevzner and M. A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, pages 19–36.

M. Riedl and C. Biemann. 2012a. How text segmentation algorithms gain from topic models. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2012)*, Montreal, Canada.

M. Riedl and C. Biemann. 2012b. Sweeping through the topic space: Bad luck? roll again! In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP held in conjunction with EACL 2012*, Avignon, France.

M. Riedl and C. Biemann. 2012c. Text segmentation with topic models. In *Journal for Language Technology and Computational Linguistics (JLCL)*, volume 27, pages 47–70.

M. Riedl and C. Biemann. 2012d. Topictiling: A text segmentation algorithm based on lda. In *Proceedings of the Student Research Workshop of the 50th Meeting of the Association for Computational Linguistics*, Jeju, Republic of Korea.

Masao Utiyama and Hitoshi Isahara. 2001. A statistical model for domain-independent text segmentation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 499–506. Association for Computational Linguistics.

Yaakov Yaari. 1997. Segmentation of expository texts by hierarchical agglomerative clustering. *arXiv preprint cmp-lg/9709015*.