jLDADMM:
A Java package for the LDA and DMM topic models

Dat Quoc Nguyen

Department of Computing
Macquarie University, Australia
dat.nguyen@students.mq.edu.au

Abstract: In this technical report, we present jLDADMM—an easy-to-use Java toolkit for conventional topic models. jLDADMM is released to provide alternatives for topic modeling on normal or short texts. It provides implementations of the Latent Dirichlet Allocation topic model and the one-topic-per-document Dirichlet Multinomial Mixture model (i.e. mixture of unigrams), using collapsed Gibbs sampling. In addition, jLDADMM supplies a document clustering evaluation to compare topic models. jLDADMM is open-source and available to download at: https://github.com/datquocnguyen/jLDADMM

Keywords: Topic modeling, short texts, LDA, DMM, document clustering

1. Introduction

Topic modeling algorithms are statistical methodologies "for analyzing documents, where a document is viewed as a collection of words, and the words in the document are viewed as being generated by an underlying set of topics" [6].[1] The probabilistic topic model Latent Dirichlet Allocation (LDA) [3] is the most widely used model to discover latent topics in document collections. However, as shown in Tang et al. [13], LDA obtains poor performance when the data presents extreme properties (e.g., very short or very few documents). That is, applying topic models to short documents, such as Tweets or instant messages, is challenging because of data sparsity and the limited contexts in such texts. One approach is to combine short texts into long pseudo-documents before training LDA [5, 14, 9, 1]. Another approach is to assume that there is only one topic per document [11, 16, 15, 12], such as in the mixture of unigrams Dirichlet Multinomial Mixture (DMM) model [15].

We present in this technical report jLDADMM—a Java package for the LDA and DMM topic models. jLDADMM is released to provide alternative choices for

[1] In fact, topic models are also used for other kinds of data [2]. However, in this report we discuss topic modeling in the context of text analysis.

topic modeling on normal or short texts. jLDADMM provides implementations of the LDA topic model [3] and the one-topic-per-document DMM [11], using the collapsed Gibbs sampling algorithms for inference as described in [4] and [15], respectively. Furthermore, jLDADMM supplies a document clustering evaluation to compare topic models, using two common metrics of Purity and normalized mutual information (NMI) [8].

Our design goal is to make jLDADMM simple to setup and run. All jLDADMM components are packaged into a single file `.jar`. Therefore, users do not have to install external dependencies. Users can run jLDADMM from either the command-line or the Java API. The next sections will detail the usage of jLDADMM in command line, while examples of using the API are available at https://github.com/datquocnguyen/jLDADMM.

Please **cite** jLDADMM when it is used to produce published results or incorporated into other software. Bug reports, comments and suggestions about jLDADMM are highly appreciated. As a free open-source package, jLDADMM is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

2. Using jLDADMM for topic modeling

This section describes the usage of jLDADMM in command line or terminal, using a pre-compiled file named `jLDADMM.jar`. Here, it is supposed that Java is already set to run in command line or terminal (e.g. adding Java to the environment variable `path` in Windows OS).

Users can find the pre-compiled file `jLDADMM.jar` and source codes in folders `jar` and `src`, respectively. Users can also recompile the source codes by simply running `ant` (it is also expected that `ant` is already installed). In addition, users can find input examples in folder `test` .

File format of input corpus: Similar to file `corpus.txt` in folder `test`, jLDADMM assumes that *each line in the input corpus file represents a document*. Here, a document is a sequence of words/tokens separated by white space characters. The users should preprocess the input corpus before training the LDA or DMM topic models, for example: down-casing, removing non-alphabetic characters and stop-words, removing words shorter than 3 characters and words appearing less than a certain times.

Now, we can train LDA or DMM by executing:

```
$ java [-Xmx1G] -jar jar/jLDADMM.jar -model <LDA_or_DMM> -corpus
<Input_corpus_file_path> [-ntopics <int>] [-alpha <double>] [-beta <double>]
[-niters <int>] [-twords <int>] [-name <String>] [-sstep <int>]
```

where parameters in [ ] are optional.

- `-model`: Specify the topic model LDA or DMM
- `-corpus`: Specify the path to the input corpus file.
- `-ntopics <int>`: Specify the number of topics. The default value is 20.
- `-alpha <double>`: Specify the hyper-parameter α. Following [15, 7], the default α value is 0.1.
- `-beta <double>`: Specify the hyper-parameter β. The default β value is 0.01 which is a common setting in the literature [4]. Following [15], the users may consider to the β value at 0.1 for short texts.
- `-niters <int>`: Specify the number of Gibbs sampling iterations. The default value is 2000.
- `-twords <int>`: Specify the number of the most probable topical words. The default value is 20.
- `-name <String>`: Specify a name to the topic modeling experiment. The default value is *"model"*.
- `-sstep <int>`: Specify a step to save the sampling outputs. The default value is 0 (i.e. only saving the output from the last sample).

Examples:

```
$ java -jar jar/jLDADMM.jar -model LDA -corpus test/corpus.txt -name
testLDA
```

The output files are saved in the same folder containing the input corpus file, in this case: the folder `test`. We have output files of `testLDA.theta`, `testLDA.phi`, `testLDA.topWords`, `testLDA.topicAssignments` and `testLDA.paras`, referring to the document-to-topic distributions, topic-to-word distributions, top topical words, topic assignments and model parameters, respectively.

Similarly, we perform:

```
$ java -jar jar/jLDADMM.jar -model DMM -corpus test/corpus.txt -beta
0.1 -name testDMM
```

Output files `testDMM.theta`, `testDMM.phi`, `testDMM.topWords`, `testDMM.topicAssignments` and `testDMM.paras` are also in folder `test`.

3. Topic inference on new/unseen corpus

To infer topics on a new/unseen corpus using a pre-trained LDA/DMM topic model, we perform:

```
$ java -jar jar/jLDADMM.jar -model <LDAinf_or_DMMinf> -paras
<Hyperparameter_file_path> -corpus <Unseen_corpus_file_path> [-niters
<int>] [-twords <int>] [-name <String>] [-sstep <int>]
```

- `-paras`: Specify the path to the hyper-parameter file produced by the pre-trained LDA/DMM topic model.

Examples:

```
$ java -jar jar/jLDADMM.jar -model LDAinf -paras test/testLDA.paras
-corpus test/unseenTest.txt -niters 100 -name testLDAinf
```

```
$ java -jar jar/jLDADMM.jar -model DMMinf -paras test/testDMM.paras
-corpus test/unseenTest.txt -niters 100 -name testDMMinf
```

4. Using jLDADMM for document clustering evaluation

We treat each topic as a cluster, and we assign every document the topic with the highest probability given the document [7]. To get the Purity and NMI clustering scores, we perform:

```
$ java -jar jar/jLDADMM.jar -model Eval -label <Golden_label_file_path>
-dir <Directory_path> -prob <Document-topic-prob/Suffix>
```

- `-label`: Specify the path to the ground truth label file. Each line in this label file contains the golden label of the corresponding document in the input corpus. See files `corpus.LABEL` and `corpus.txt` in folder `test`.
- `-dir`: Specify the path to the directory containing document-to-topic distribution files.
- `-prob`: Specify a document-to-topic distribution file or a group of document-to-topic distribution files in the specified directory.

Examples:

```
$ java -jar jar/jLDADMM.jar -model Eval -label test/corpus.LABEL -dir
test -prob testLDA.theta
```

```
$ java -jar jar/jLDADMM.jar -model Eval -label test/corpus.LABEL -dir
test -prob testDMM.theta
```

The above commands will produce the clustering scores for files `testLDA.theta` and `testDMM.theta` in folder `test`, separately.

The following command:

```
$ java -jar jar/jLDADMM.jar -model Eval -label test/corpus.LABEL -dir
test -prob theta
```

will produce the clustering scores for all document-to-topic distribution files with their names ending in `theta`. In this case, they are `testLDA.theta` and `testDMM.theta`. It also provides the *mean* and *standard deviation* of the scores.

To improve evaluation scores, the users might consider combining the LDA and DMM models with word embeddings [10], with the source code at HERE.

References

[1] Bicalho, P., Pita, M., Pedrosa, G., Lacerda, A., Pappa, G. L., 2017. A General Framework to Expand Short Text for Topic Modeling. Information Sciences 393 (C), 66–81.

[2] Blei, D. M., 2012. Probabilistic Topic Models. Communications of the ACM 55 (4), 77–84.

[3] Blei, D. M., Ng, A. Y., Jordan, M. I., 2003. Latent Dirichlet Allocation. Journal of Machine Learning Research 3, 993–1022.

[4] Griffiths, T. L., Steyvers, M., 2004. Finding scientific topics. Proceedings of the National Academy of Sciences of the United States of America 101 (Suppl 1), 5228–5235.

[5] Hong, L., Davison, B. D., 2010. Empirical Study of Topic Modeling in Twitter. In: Proceedings of the First Workshop on Social Media Analytics. pp. 80–88.

[6] Jordan, M. I., Mitchell, T. M., 2015. Machine learning: Trends, perspectives, and prospects. Science 349 (6245), 255–260.

[7] Lu, Y., Mei, Q., Zhai, C., 2011. Investigating task performance of probabilistic topic models: an empirical study of PLSA and LDA. Information Retrieval 14, 178–203.

[8] Manning, C. D., Raghavan, P., Schütze, H., 2008. Introduction to Information Retrieval. Cambridge University Press.

[9] Mehrotra, R., Sanner, S., Buntine, W., Xie, L., 2013. Improving LDA Topic Models for Microblogs via Tweet Pooling and Automatic Labeling. In: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 889–892.

[10] Nguyen, D. Q., Billingsley, R., Du, L., Johnson, M., 2015. Improving Topic Models with Latent Feature Word Representations. Transactions of the Association for Computational Linguistics 3, 299–313.

[11] Nigam, K., McCallum, A., Thrun, S., Mitchell, T., 2000. Text Classification from Labeled and Unlabeled Documents Using EM. Machine learning 39, 103–134.

[12] Surian, D., Nguyen, D. Q., Kennedy, G., Johnson, M., Coiera, E., Dunn, G. A., 2016. Characterizing Twitter Discussions About HPV Vaccines Using Topic Modeling and Community Detection. Journal of Medical Internet Research 18 (8), e232.

[13] Tang, J., Meng, Z., Nguyen, X., Mei, Q., Zhang, M., 2014. Understanding the Limiting Factors of Topic Modeling via Posterior Contraction Analysis. In: Proceedings of the 31st International Conference on International Conference on Machine Learning. pp. 190–198.

[14] Weng, J., Lim, E.-P., Jiang, J., He, Q., 2010. TwitterRank: Finding Topic-sensitive Influential Twitterers. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining. pp. 261–270.

[15] Yin, J., Wang, J., 2014. A Dirichlet Multinomial Mixture Model-based Approach for Short Text Clustering. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 233–242.

[16] Zhao, W. X., Jiang, J., Weng, J., He, J., Lim, E.-P., Yan, H., Li, X., 2011. Comparing Twitter and Traditional Media Using Topic Models. In: Proceedings of the 33rd European Conference on Advances in Information Retrieval. pp. 338–349.