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. The Java package 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.

1. Introduction

jLDADMM is released to provide alternative choices for topic modeling on normal or short texts. Probabilistic topic models, such as Latent Dirichlet Allocation (LDA) [2] and related models [1], are widely used to discover latent topics in document collections. However, applying topic models for short texts (e.g. Tweets) is more 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. Another approach is to assume that there is only one topic per document.

jLDADMM provides implementations of the LDA topic model [2] and the one-topic-per-document Dirichlet Multinomial Mixture (DMM) model (i.e. mixture of unigrams) [7]. The implementations of LDA and DMM use the collapsed Gibbs sampling algorithms for inference as described in [3] and [8], respectively. Furthermore, jLDADMM supplies a document clustering evaluation to compare topic models, using two common metrics of Purity and normalized mutual information (NMI) [5].

jLDADMM is available to download at https://github.com/datquocnguyen/jLDADMM

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).

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

File format of input corpus: Similar to file `corpus.txt` in the `test` folder, 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 alpha. The default value is 0.1. See experimental details in [8, 4].
- `-beta <double>`: Specify the hyper-parameter beta. The default value is 0.01 which is a common setting in the literature [3]. Following [8], the users may consider to the beta value of 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 in the `test` folder. 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 in the `test` folder.

3. Using jLDADMM for document clustering evaluation

Here, we treat each topic as a cluster, and we assign every document the topic with the highest probability given the document [4]. 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 the `test` folder.
- `-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
```

They will produce the clustering scores for files `testLDA.theta` and `testDMM.theta` in the `test` folder, 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 topic models with word embeddings [6], with the source code at HERE.

4. Citation

Please cite jLDADMM in all publications reporting on results obtained with the help of jLDADMM:

Dat Quoc Nguyen. jLDADMM: A Java package for the LDA and DMM topic models. 2015. [.bib]

References

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

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

[3] 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.

[4] 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.

[5] Manning, C. D., Raghavan, P., Schütze, H., 2008. Introduction to Information Retrieval. Cambridge University Press.
[6] 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.
[7] Nigam, K., McCallum, A., Thrun, S., Mitchell, T., 2000. Text Classification from Labeled and Unlabeled Documents Using EM. Machine learning 39, 103–134.
[8] 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.