LDA For Text Classification

By: Ahmed Abdelatty

1 Introduction:

In the last two decades, text classification got a lot of attention, and a lot of research have been investigating the development of better classifiers [7], [8], [9]. Feature selection proved to be very critical for making the classification more efficient and accurate, which in turn led to intensively studies of text feature extraction [10]. Latent Dirichlet allocation (LDA) is a generative statistical model, which is used as an unsupervised tool for topic modeling. LDA represents each document as a mixture of various topics, where there is a probabilistic distribution represents the probability of each word given each topic. LDA have been used as a future extraction tool for multiple NLP tasks like classification [5], [6]. In this report, we are investigating the performance of LDA in text classification, and the factors upon which the performance depends. Moreover, the report is providing a comparison between LDA and the state of the art feature extraction techniques like tf-idf, and word to vector.

2 Background:

2.1 Problem Definition:

Text classification problem can be represented by

1. Training data, which is a set of documents TD = $\{d_1, d_2, \ldots.., d_N\}$.
2. A set of classes C = $\{c_1, c_2, \ldots\ldots, c_M\}$

To solve such a problem, we need to design a function $f: D \rightarrow C$, that maps any document D to a certain class c

2.2 Latent Dirichlet allocation (LDA):

LDA is a generative probabilistic model used to represent discrete data, proven to performed well in text domain [1]. LDA is used to represent text or a document as a mixture of latent topics, and statistical dependence between every word in the corpus and every topic is captured. The graphical model representation for LDA is shown is Figure 1.

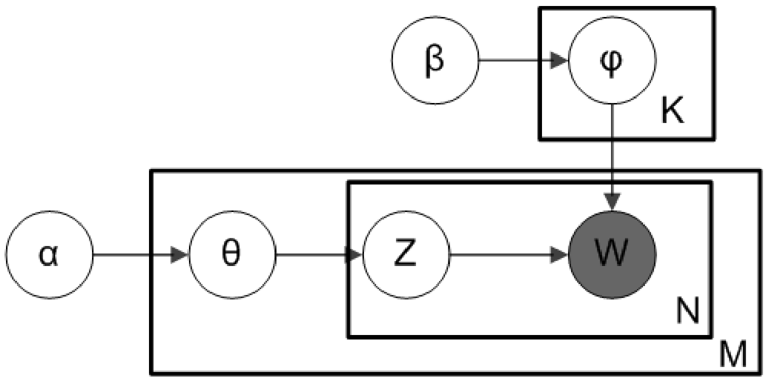

Figure 1

Where the outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document. M denotes the number of documents, N the number of words in a document. Thus:

> α is the parameter of the Dirichlet prior on the per-document topic distributions,
> β is the parameter of the Dirichlet prior on the per-topic word distribution,
> θ is the topic distribution for document *m*,
> φ is the word distribution for topic *k*,
> $z_{m,n}$ is the topic for the *n*-th word in document *m*, and
> $w_{m,n}$ is the specific word.

Using this model, the joint distribution can be calculated as follows:

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j) P(W_{j,t} \mid \varphi_{Z_{j,t}}),$$

2.3 Inference for LDA:

Variance number of inference algorithms have been developed for LDA, [1] provided an approximation of the posterior distribution using variational Bayes, [4] used collapsed Gibbs sampling. For this report we are going to give a brief explanation to collapsed Gibbs sampling. In a nut shell, collapsed Gibbs sampling, sum out or integrate out some variables and sample for other variables. For example, using collapsed Gibbs sampling in our case, will allow us to calculate the joint distribution over Z, and W as follows:

$$P(\boldsymbol{Z}, \boldsymbol{W}; \alpha, \beta) = \int_{\boldsymbol{\theta}} \int_{\boldsymbol{\varphi}} P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta)\, d\boldsymbol{\varphi}\, d\boldsymbol{\theta}$$

3 Extracting features using LDA:

As mentioned before LDA proved to be a very effective feature extractor, and dimensionality reduction technique [5], [6]. What makes LDA superior to some of the state of the art feature extraction techniques like tf-idf is its ability to create continuous vector representations of documents or words, which means LDA can preserve semantic relation. For this section, we are going to present two approaches of how LDA can be used to produce a continuous vector representation of a document.

3.1 Topics distribution as features (LDA-td):

In this approach, for each document d, using a trained LDA model, and using inference as in **Section 2.3** we can learn **f,** where **f** is a feature vector such that:

$$\mathbf{f}[\mathrm{i}] = \mathrm{p}(\mathrm{d}\ |\mathrm{topic_i}\ )\ \ ,1 <= \ \mathrm{i}\ <= \ \text{number of topics}$$

Where $\mathrm{p}(\mathrm{d}\ |\mathrm{topic_i}\ )$ is the conditional probability of document d given $\mathrm{topic_i}$. using the notation of **(figure 1)** f is basically equals to θ.

3.2 Words distribution as features (LDA-wd):

This approach is slightly different, in which we calculate the feature vector of a document d as the mean of the topic distribution vectors of the words in d, **f** can be represented as follows:

$$\mathbf{f}[\mathrm{i}] = \frac{1}{n}\sum_{1=j}^{n} \mathrm{p}\left(\mathrm{w_j^d}\ |\mathrm{topic_i}\ \right)\ \ ,1 <= \ \mathrm{i}\ <= \ \text{number of topics}$$

Where $\mathrm{w_j^d}$ is the word with index j in document d, n is the number of words in d, and $\mathrm{p}\left(\mathrm{w_j^d}\ |\mathrm{topic_i}\ \right)$ is the conditional probability of word $\mathrm{w_j^d}$ given $\mathrm{topic_i}$

5 Text Classification

The classification task is divided into three stages, the first stage or the preprocessing stage, in which we deleted all the stop-words. The second stage is concerned with extracting the useful features of the text. For the third stage, we use off-the-shelf classifiers like SVM, and KNN.

6 Experiments

To compare the semantic extraction of each of our methods, the experiments have been carried out on two data sets R8 of Reuters 21578, and 20 newsgroups. Where the class of the former are more semantically related. More about the datasets can be found at [12], [13]. Moreover, both datasets were divided into 80% (training set), and 20% (test set).

6.1 Feature extraction Methods

In this sub-section we present all the methods used as a feature extractor, to be compared with the methods provided at (**Section 3**)

6.1.1 Term frequency

Term frequency tf(t,d), is simply the representation of a document d as a vector of the raw count of a term/words in a document.

6.1.2 Term frequency–Inverse document frequency

Term frequency–Inverse document frequency (tf–idf) represents document d as a vector of the weighted count of a term/words in a document. There are many ways to calculate the weighted count of a term [11], the most popular one however is as follows:

$$\mathbf{tfidf}(\boldsymbol{t}, \boldsymbol{d}, \boldsymbol{D}) = \mathbf{tf}(\boldsymbol{t}, \boldsymbol{d}) \cdot \mathbf{idf}(\boldsymbol{t}, \boldsymbol{D})$$

Where,

$$\mathbf{idf}(\boldsymbol{t}, \boldsymbol{D}) = \mathbf{log}\,\frac{\boldsymbol{N}}{|\{\boldsymbol{d} \in \boldsymbol{D} : \boldsymbol{t} \in \boldsymbol{d}\}|}$$

Such that, N is the total number of documents, $|\{\boldsymbol{d} \in \boldsymbol{D} : \boldsymbol{t} \in \boldsymbol{d}\}|$ is the number of documents where the term t appears.

6.1.3 Word to Vector

Word to Vector is technique, which uses Deep Neural Nets, to create continuous vector representations for words. Like LDA, Word to Vector is preserving semantic relations. One way to use Word to Vector as a feature extractor is calculate the mean of the vector representation of all the words represented in a document d. To be more specific we used Stanford pretrained word to vector representation (glove)[14],where every word is represented with 100 features.

6.2 Results

In this section, we compare all previously mentioned feature extraction techniques, by applying both KNN, and SVM on the features extracted by them. Moreover, we are demonstrating how LDA training time difference by changing the number of topics.

6.2.1 R8 Results

	tf	tf-idf	Word2vec	LDA-td	LDA-wd
KNN	86.4%	82.6%	94.2%	88.7%	46.4%
SVM	96.29%	97.3%	95.4%	97.2%	50.1%

Please note that the results given for both LDA-td, and LDA-wd are the best results we got after tuning the number of topics. More on who the accuracy and the training time variance is presented in (figure 2,3). As we can see, LDA-td performs very well, however LDA-we performed bad. Another thing to note, training time isn't always being proportional to the number of topics, this can be for many reasons. The most important reason, as mentioned before LDA's parameters are learned through approximation which means they doesn't have to be corresponding to a global optimal, which in turn means the algorithm will terminate ones a local optima is found.

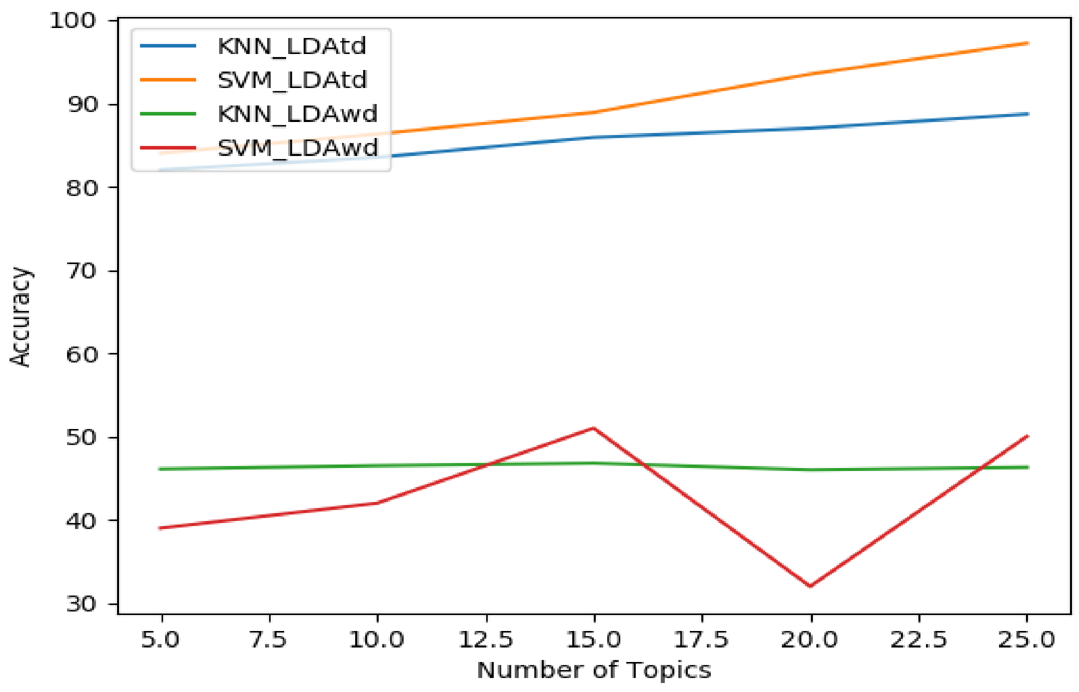

Figure 2

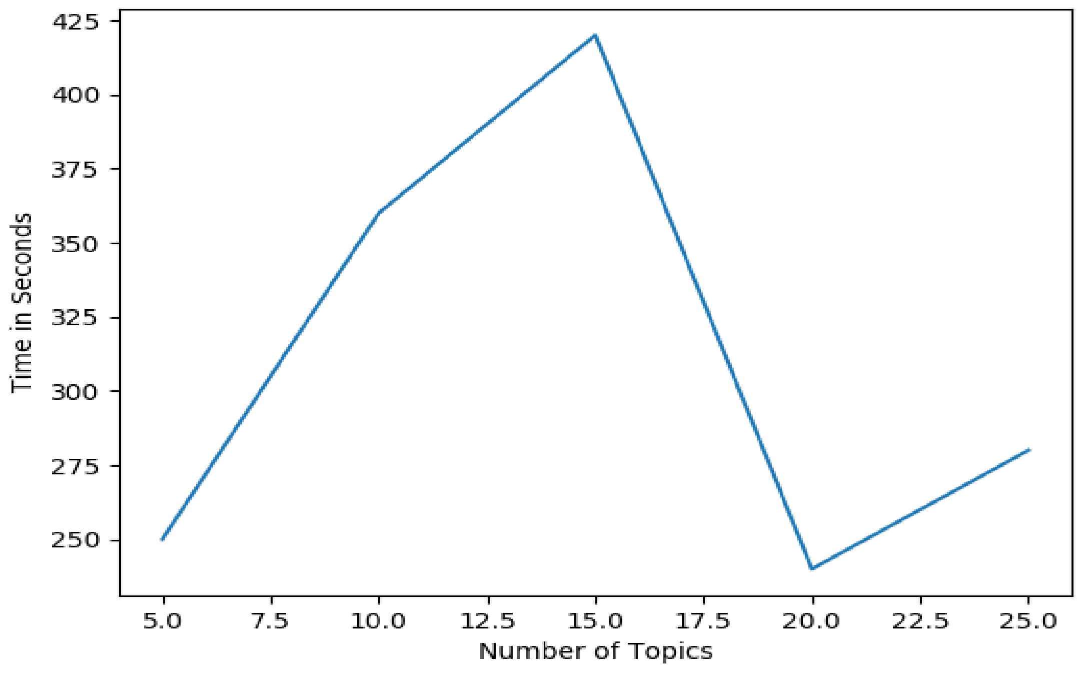

Figure 3

6.2.2 20_Newsgroup Results

	tf	tf-idf	Word2vec	LDA-td	LDA-wd
KNN	37.4%	72%	60.6%	41.4%	7.4%
SVM	76.4%	84.1%	63.57%	44.9%	8.1%

Please note that this dataset has been selected very carefully to demonstrate when LDA or similar semantic feature extractors are not expected to perform as well as tf-idf. The main reason for that is some of the classes at this dataset are very semantically related (e.g. comp.sys.ibm.pc.hardware / comp.sys.mac.hardware), where both are related to Computer Hardware. This closely semantic relation, made it hard for both LDA, and Word2Vec to outperform tf-idf.

The last note to take away is that LDA-wd proved to be a bad performer on both datasets. More on who the accuracy and the training time variance is presented in (figure 4,5).

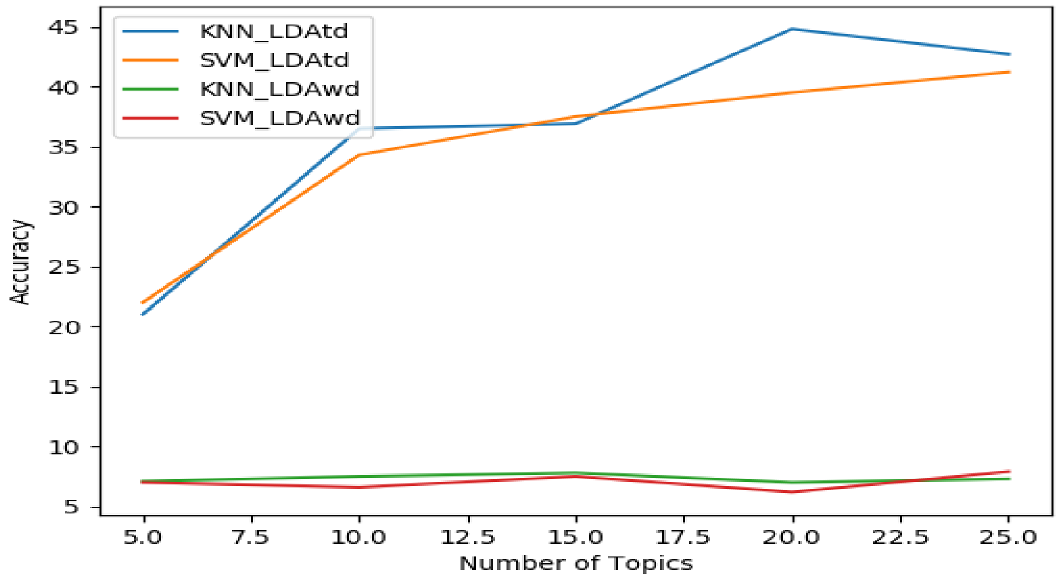

Figure 4

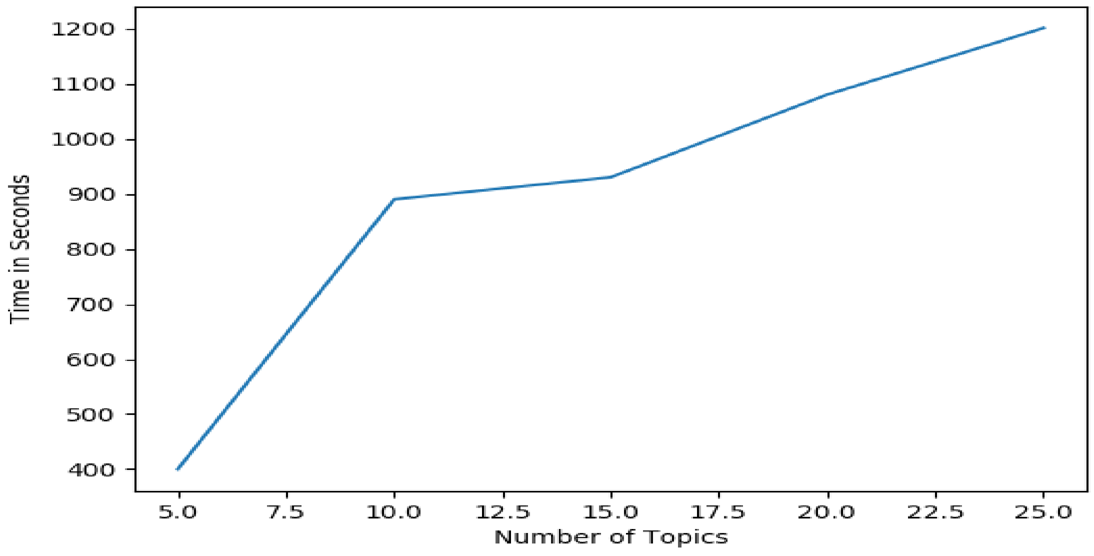

Figure 5

7 Conclusion

The experiment results have Shawn that LDA is a very effective tool for sentiment feature extraction, which have almost the same accuracy of Word to Vector representation. Moreover, the experiments proved how LDA can be used as a dimensionality reduction technique: just by using the distribution over 20 topics to represents every document, LDA was able give comparable accuracy to tf-idf, although tf-idf represents every document by a feature vector of length equals to the number of the words in the corpus. However, LDA have some short comings, like long training time, and LDA parameters are approximated which means no guarantee for optimality. Moreover, the report showed that LDA is not the best choice for classification feature extraction if different class in the dataset are highly semantically related.

8 Future work

For sure we covered most of the aspects related to LDA as a classification tool, however, there is some points that we would like to investigate more in the future. For example, no doubt that Word to Vector is performing better on its own, whoever some variations have been proposed like weighted Word to Vector, which is a method that combines both Word to Vector and tf-idf. We would like to investigate if such techniques can be used with LDA, and wither or not it will enhance the performance.

9 References:

[1] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation, 2003.

[2] https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation.

[3] https://en.wikipedia.org/wiki/Maximum_likelihood_estimation.

[4] Xiao, Han, and Thomas Stibor. "Efficient collapsed gibbs sampling for latent dirichlet allocation." Proceedings of 2nd Asian Conference on Machine Learning. 2010.

[5] Chen, Qiuxing, Lixiu Yao, and Jie Yang. "Short text classification based on LDA topic model." Audio, Language and Image Processing (ICALIP), 2016 International Conference on. IEEE, 2016.

[6] Li, Kunlun, et al. "Multi-class text categorization based on LDA and SVM." Procedia Engineering 15 (2011): 1963-1967.

[7] McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naive bayes text classification." AAAI-98 workshop on learning for text categorization. Vol. 752. 1998.

[8] Joachims, Thorsten. "Transductive inference for text classification using support vector machines." ICML. Vol. 99. 1999.

[9] Tong, Simon, and Daphne Koller. "Support vector machine active learning with applications to text classification." Journal of machine learning research 2.Nov (2001): 45-66.

[10] Forman, George. "An extensive empirical study of feature selection metrics for text classification." Journal of machine learning research 3.Mar (2003): 1289-1305.

[11] Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. Technical report, Ithaca, NY, USA, 1987.

[12] http://csmining.org/index.php/r52-and-r8-of-reuters-21578.html

[13] http://qwone.com/~jason/20Newsgroups/

[14] https://nlp.stanford.edu/projects/glove/