Experimenting with Gibbs Sampling Dirichlet Multinomial Mixture Model for Short-Text Topic Modeling

Noon Pokaratsiri Goldstein

March 22, 2020

Abstract

The project implements the Gibbs Sampling Dirichlet Multinomial Mixture Model for short-text clustering and topic modeling. Resulting clusters are evaluated in terms of homogeneity and completeness. Various Dirichlet β values were explored to observe how the hyper-parameter affects overall performance.

1 Introduction

In an era where vital information is obtained from tweets and internet forum queries, the ability to efficiently group this type of data into sensible topics may provide certain benefits to various downstream tasks. While Latent Dirichlet Allocation (LDA) (Blei, 2003) is a robust and effective topic modeling approach, it is not without limitations. Among these are the fact that LDA requires a known fixed number of topics as part of the algorithm (Blei, 2003) and that the resultant topics become less coherent when applied to short texts (Mazarura and de Waal, 2016).

One of the approaches to deal with the data sparsity problem in short-text clustering and topic modeling is the Gibbs Sampling algorithm for Dirichlet Multinomial Mixture Model (GSDMM) proposed by Yin and Wang (2014). GSDMM not only infers the number of topics automatically from the data, but the algorithm can also obtain the representative words of each resultant cluster (Yin and Wang, 2014). In addition, there is a clear relationship between its hyper-parameters (α, β) and the clusters' homogeneity and completeness scores (Yin and Wang, 2014).

This project compares the results from my implementation of GSDMM on a short-text data set to those achieved by Yin and Wang (2014) in order to observe the extent to which the performance variability follows the characteristics described in the paper.

2 Approach

The GSDMM algorithm was run on the data set described in Section 2.1 5 times with the following β values: *0.02, 0.05, 0.1, 0.2, 0.3*. The α hyper-parameter was kept constant at

0.1. According to the experiment on different α values on the *TweetSet* in Yin and Wang (2014), there is not a signifiant variability in the evaluation results when α values are greater than 0 and less than 1. The initial number of topics, K, was set at 100 for all runs. As long as the initial number of topics exceeds the true number of topics in the data, this number can be any arbitrary number (Yin and Wang, 2014). Although Yin and Wang (2014) set $K = 500$ as their data sets have over 100 true clusters, since the data set in this experiment only has 20 true topics, setting $K = 100$ provides a sufficient margin above the true number of topics. As with the experiments in Yin and Wang (2014), the number of iterations was fixed at 10 for all runs.

2.1 Data Set

Ideally, I had wanted to replicate the results obtained in Yin and Wang (2014) by applying GSDMM on the same data sets. Since Yin and Wang (2014) labeled data sets were not obtainable nor duplicatable, I evaluated my implementation on the *Stack Overflow Titles* data set developed for short-text clustering tasks by Xu et al. (2015) instead; the data set was made available by *Kaggle*[1].

The *Stack Overflow Titles* data set has ground-truth labels, which, albeit not necessary for the clustering and topic modeling tasks, were necessary for the evaluation metrics discussed in Section 2.3. The data set consists of 20,000 Stack Overflow titles as short-text documents divided equally across 20 topic labels with each topic containing 1,000 documents (Xu et al., 2015). The 20 topic labels, i.e. the most frequent words from the clusters, are shown in Table 2.

2.2 Pre-Processing

The following pre-processing steps as described in Yin and Wang (2014) were performed on the data set:

1. converting all letters to lowercase
2. removing non-latin characters and stopwords
3. lemmatizing with *NLTK WordNet Lemmatizer*[2]
4. removing words comprising fewer than 2 or more than 15 characters
5. removing words whose per-document frequencies are less than 2

2.3 Evaluation Metrics

Results are evaluated with the ground-truth labels based on Homogeneity (H), Completeness (C), and Normalized Mutual Information (NMI) scores. These are the same metrics used in Yin and Wang (2014) to evaluate GSDMM performance against other clustering algorithms. Homogeneity measures the extent to which *only* members of a ground-truth group are assigned the same topic and completeness represents the degree to which *all* members of a

[1] http://www.kaggle.com
[2] http://www.nltk.org

ground-truth group are assigned the same topic; Normalized Mutual Information measures how successfully the criteria of homogeneity and completeness have been satisfied (Yin and Wang, 2014).

I utilized *sklearn*[3] implementation for all of the evaluation metrics in this project.

3 GSDMM

My implementation of the GSDMM algorithm was written in Python 3.7[4] and followed the pseudocode in Yin and Wang (2014), which is described in Algorithm 1.

In LDA, each document is viewed as a composition of words that are sampled from a mixture of topics (Blei, 2003). In GSDMM, since each document is a short text, it is assumed that instead of a mixture of topics, each document only has 1 topic and the words are sampled from only the document topic. Each topic is then assumed to follow a multinomial distribution over words and the prior for this multinomial distribution follows a Dirichlet distribution; the weight of each topic cluster is also sampled from a multinomial distribution whose prior is also a Dirichlet distribution (Yin and Wang, 2014).

Given the above assumptions, the conditional probability of a document d belonging to a topic cluster (as specified in the the re-sampling step described in Algorithm 1 in Section A, can be represented by the equation[5]:

$$p(z_d = z|\vec{z}_{\neg d}, \vec{d}) \propto \frac{m_{z,\neg d} + \alpha}{D - 1 + K\alpha} \frac{\prod_{w \in d} \prod_{j=1}^{N_d^w} (N_z^w{}_{,\neg d} + \beta + j - 1)}{\prod_{i=1}^{N_d} (n_{d,\neg d} + |Vocab|\beta + i - 1)} \tag{1}$$

More simply, Yin and Wang (2014) explains the steps in GSDMM through using the analogy of students in a Movie Group Process as follows:

> The input are D students (documents) and each student (document) is represented as a short list of movies (words) that they like. The goal is to cluster the students (documents) into several groups, so that students (documents) in the same group are similar in their representative movies (words) and students (documents) in different groups are dissimilar. The grouping process follows 2 rules:
>
> - **Rule 1 -** Choose a group with more students (documents).
> - **Rule 2 -** Choose a group whose students (documents) share similar movie interests (i.e. similar words).

In this way, topics with more documents will continue to have more documents assigned to them. On the contrary, topics with fewer documents will have increasingly fewer documents with each subsequent iteration. Eventually, many clusters will end up having 0 documents. After a number of iterations, the number of non-empty clusters approaches the number of topics inherent in the data. GSDMM can converge to a stable number of clusters in as few as 5 iterations (Yin and Wang, 2014) with the words in each resultant

[3] https://scikit-learn.org
[4] https://www.python.org/
[5] Consult the notation in Section A and see Yin and Wang (2014) for details regarding the derivation.

non-empty cluster being representative of the same topic. The extent to which GSDMM accomplishes this with respect to the chosen data set can be examined in the program output file, `gsdmm_clusters_and_representative_words_3.out`, which lists the 5 most frequent or representative words in each cluster after 10 iterations.

4 Results and Discussion

4.1 Homogeneity, Completeness, Normalized Mutual Information and β

Far from the high performance obtained in Yin and Wang (2014) of H, C, and NMI scores approximately between 0.8-0.9, my GSDMM implementation on the *Stack Overflow Titles* data set only achieved the H, C, and NMI scores between 0.4-0.5.

Despite inferior performance, the behaviors described in Yin and Wang (2014) can still be observed in the results from this data set. For instance, as Figure 1 shows, the algorithm converges to a stable number of clusters in as few as 5 iterations. Figure 1 and Table1 also demonstrate that as β value increases, the algorithm converges to a smaller number of clusters. Yin and Wang (2014) attributes this behavior to the relationship between enlarged β values and increased completeness as observable in Figure 2. Interestingly, this trend reverses to the contrary with a decline in completeness score as β value increases from 0.2 to 0.3. Unfortunately, Yin and Wang (2014) did not include experiments with β values beyond 0.2.

Furthermore, Figure 2 also demonstrates that my results follow the homogeneity score and β value relationship observed in Yin and Wang (2014): as the number of clusters decreases and the β value increases, homogeneity score decreases. With a decrease in cluster number, more documents that otherwise should have been assigned to different groups would be forced into the same topic cluster.

As for NMI, similar to what was observed in Yin and Wang (2014), Figure 2 shows that NMI peaked at $\beta = 0.1 - 0.2$ before dropping for $\beta > 0.2$. It is apparent that 0.1 seems to be the optimally compromised β value that balances homogeneity, completeness, and NMI scores. As Yin and Wang (2014) observed, adjusting the β term can affect GSDMM performance in terms of homogeneity and completeness.

4.2 Representative Words

Consistent with the relationship described in Section 4.1 between β and homogeneity, the most frequent representative words in Table 3 show that there are more words in common between the true and predicted clusters in the model with $\beta = 0.1$ than in the model with $\beta = 0.2$. Observing the ballpark ratio between the overlapping and non-overlapping words of around 1/2, it seems that the H, C, and NMI scores of around 0.4-0.5 are to be expected.

Despite the disappointing performance, it is apparent that the words listed in Table 3, even when they are different from those in the true clusters, still follow the thematic subjects of the labels to which they are assigned. If one examines the true labels in Table

β	No. of Clusters
0.02	100
0.05	64
0.1	26
0.2	20
0.3	13

Table 1: Number of Clusters at different β values

No. Clusters	True Topic Labels
20	*wordpress, oracle, svn, apache, excel, matlab, visual-studio, cocoa, osx, bash, spring, hibernate, scala, sharepoint, ajax, qt, drupal, linq, haskell, magento*

Table 2: True cluster labels (Xu et al., 2015)

2 and the predicted labels in Table 4, it is evident that all the predicted labels can be considered related to the true labels. It is possible that the different topics inherent in the *Stack Overflow Titles* corpus are too closely related for the model to perform effectively. A potential avenue for future explorations could be to experiment with data sets whose topics are more thematically different from one another to see whether the model's H, C, and NMI scores are closer to the what Yin and Wang (2014) obtained.

4.3 Underflow Imprecision

While the evaluation results follow the expected trends and the qualitative analysis of the representative words suggest that my GSDMM implementation works as expected, the low H, C, and NMI scores may also have been caused by underflow imprecision. In an attempt to avoid this problem, I implemented all calculations in the natural log space. This approach still resulted in *NaN* values for many entries until I forced a 128-bit float data type and converted the normalized probabilities back to 64-bit float for *sklearn* function compatibility. It is possible that the solution I implemented was not sufficient to address underflow errors and further explorations for a more precise handling of the calculations may potentially result in a better model performance.

5 Conclusion

Overall, my GSDMM implementation performed as expected in finding the number of clusters inherent in the data as well as modeling corresponding topics. In terms of H, C, and NMI scores, the implementation was not able to duplicate the performance reported in Yin and Wang (2014) on the *Stack Overflow Titles* data set. Possible ideas to explore in the future include finding more effective ways to handle underflow imprecision and testing the model on other short-text corpora.

[6]Repeated labels or labels with two few document members are omitted; see program output file for complete listing

topic labels:	wordpress	svn	excel
true clusters:	***post***, ***page***, *category*, *plugin*	***subversion***, ***file***, ***repository***, ***directory***	***vba***, ***cell***, ***file***, *data*
$\beta = 0.1$	*drupal*, ***page***, ***post***, *custom*	***file***, ***subversion***, ***repository***, ***repository***	***file***, *matlab*, ***cell***, ***vba***
$\beta = 0.2$	*drupal*, *sharepoint*, ***page***, *magento*	*apache*, ***file***, ***subversion***, *server*	*oracle*, *sql*, ***file***, ***data***

Table 3: Representative words in select clusters; ***bold*** typeface represents matches between words in the true and predicted clusters

β	No. Clusters	Predicted Topic Labels[6]
0.02	100	*magento, oracle, visual, mac, svn, drupal, scala, linq, spring, sharepoint, bash, ajax, apache, wordpress, excel, hibernate, file, type, cocoa, matlab, haskell, qt, array, package, studio, parser, key, recursion, branch, foundation, sometimes*
0.05	64	*sharepoint, linq, visual, excel, drupal, magento, bash, file, web, scala, spring, ajax, svn, spring, apache, oracle, mac, matlab, cocoa, hibernate, haskell, window, menu, row, unicode, qt, bracket, combinator, wordpress*
0.1	26	*ajax, wordpress, scala, excel, linq, svn, visual, hibernate, magento, bash, sharepoint, qt, spring, oracle, mac, matlab, apache, haskell, drupal, language, stringstream, bracket, ccitt*
0.2	20	*wordpress, haskell, visual, svn, excel, linq, spring, qt, ajax, magento, matlab, hibernate, callcc, viewdidload, cube, preserve, transform*
0.3	13	*visual, magento, ling, spring, svn, hibernate, item, nsopenglview, encoding, plotting, explanation, difference, formula*

Table 4: Number of clusters at different β values and sample topics; topic labels represent the most frequent words assigned to the resulting clusters

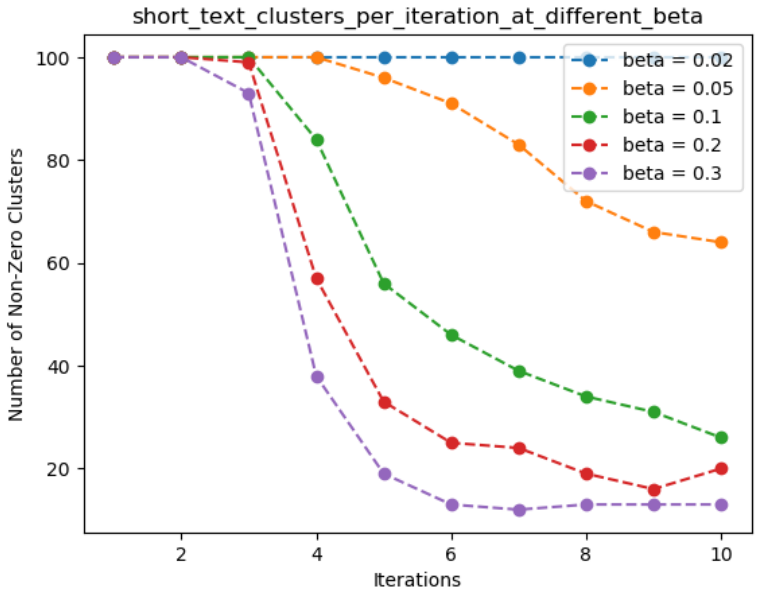

Figure 1: Number of non-zero document clusters after different iterations across experiments of different β values

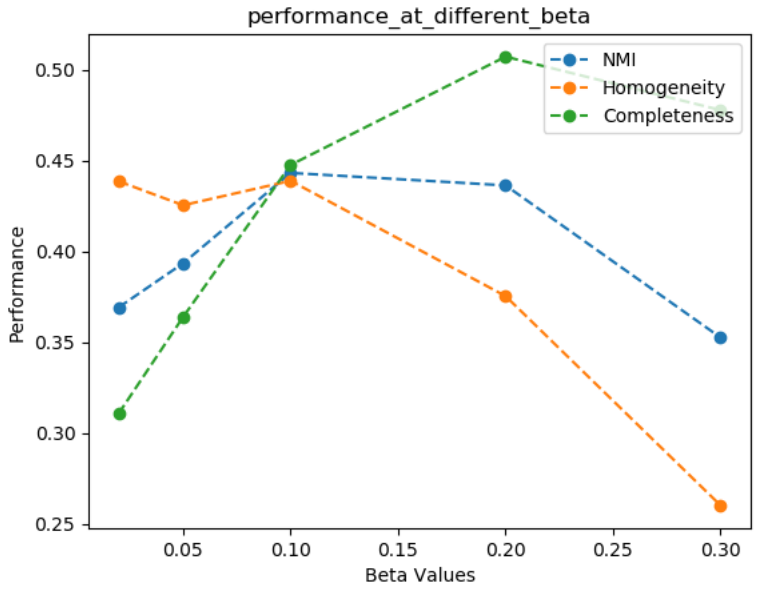

Figure 2: Homogeneity, Completeness, and NMI scores at different β values

A Appendix

Algorithm 1: GSDMM as proposed by Yin and Wang (2014)

Data: Documents in the input, $\vec{d}$
Result: Cluster labels of each document, $\vec{z}$
initialize number of documents (m_z), number of words (n_z), and occurrences of word w in cluster z (N_z^w) as zero;
for *each document* d $\in [1, No.documents, D]$ **do**
  sample a cluster for d;
  assign a topic z to a document d, drawing z from a *Multinomial* distribution with $p = 1$ / number of clusters (K);
  increment number of documents in cluster z (m_z), by 1;
  increment number of words in cluster z (N_z), by the number of words in document d (N_d);
  for *each word* $w \in document, d$ **do**
    add the number of occurrences of word w in document d (N_d^w) to the number of occurrences of word w in cluster z (N_z^w);
  end
end
for $i \in [1, No.iterations]$ **do**
  for *each document* $d \in [1, No.documents, D]$ **do**
    record the current cluster of d;
    decrement by 1 the number of documents in cluster z, (m_z);
    decrement the number of words in cluster z (N_z^w) by the number of occurrences of word w in the document d (N_d^w);
    for *each word* $w \in d$ **do**
      subtract the number of occurrences of word w in document d, (N_d^w), from the number of occurrences of word w in cluster z, (N_z^w);
    end
    resample a cluster for d;
    draw a topic z from the *Multinomial* distribution with p = conditional probability of document d belonging to a topic cluster z, when the topic label of document d is removed from $\vec{z}$: $p(z_d = z|\vec{z}_{\neg d}, \vec{d})$;
    increment number of documents in cluster z (m_z) by 1;
    increment number of words in cluster z (N_z) by the number of words in document d (N_d);
    for *each word* $w \in documentd$ **do**
      add the number of occurrences of word w in document d (N_d^w) to the number of occurrences of word w in cluster z (N_z^w);
    end
  end
end

References

Blei, D. M. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Mazarura, J. and de Waal, A. (2016). A comparison of the performance of latent Dirichlet allocation and the Dirichlet multinomial mixture model on short text. pages 1–6.

Xu, J., Wang, P., Tian, G., Xu, B., Zhao, J., Wang, F., and Hao, H. (2015). Short Text Clustering via Convolutional Neural Networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 62–69, Denver, Colorado. Association for Computational Linguistics.

Yin, J. and 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 - KDD '14*, pages 233–242, New York, New York, USA. ACM Press.