A research report based on the paper: Model-based Clustering of Short Text Streams

Bo, Jason, Kamlakant*
Department of Computer Science
Old Dominion University
Norfolk, VA 23529
bzhan001@odu.edu
jason.orender@outlook.com
ktripath@cs.odu.edu

December 14, 2018

Abstract

Due to the rapid growth of short text documents on popular social media platforms such as Twitter, solving the problem of short text clustering by topic has become a valuable line of inquiry for data scientists looking for real-time insight regarding public opinion in order to advise on strategies for everything from political campaigns to shoe sales.

The paper selected for our project proposes a model-based short-text continuous stream clustering algorithm (MStream) that is demonstrated to better provide solutions for problems related to topical drift and text sparsity. The proposed MStream algorithm can achieve state-of-the-art performance with only one pass of the stream, and can show even better performance when multiple iterations are allowed over a well defined batch. Moreover, the authors also provide an aditional feature with the so-called "MStreamF" version of the algorithm, which is designed to delete clusters of an outdated batch.

For our project, we used the same data set as the original authors and constructed new Python code based on the algorithms detailed in the paper. As the figures will show, the clustering observed was very similar to the results shown from the original paper.

*Bo, Jason, and Kamlakant are graduate students in Department of Computer Science at Old Dominion University, Virginia, US.

1 Introduction

The proliferation of short-text documents available from a wide range of social media sources has generated a large amount of potentially valuable data and a devising an efficient way to accurately extract meaning from this data trove has become a worthwhile pursuit. Clustering the data into relevant groups is one of the ways that meaning can be extracted in an unsupervised fashion, and there are several models currently in use.

The family of algorithms known as DTMs (Dynamic Topic Models) are generative models that can be used to analyze the evolution of topics of a collection of documents over time [4]. The DCT-L (Dynamic Clustering Topic model with long term dependency) enables tracking of time-varying distributions of topics over documents in addition to words over topics [6]. Lastly, Sumblr proposes an online tweet streem clustering algorithm which is able to efficiently cluster the tweets and maintain compact cluster statistics with only one pass of the stream [5].

The introduction of MStream and MStreamF is an attempt to capture the best elements of the current state of the art and improve the clustering characteristics as well as the speed at which the clustering occurs.

2 Proposed models

The defining contribution of the algorithms proposed in this paper is the use of the Dirichlet Process Multinomial Mixture (DPMM) model used for calculating the probabilities of either being added to a current cluster or creating a new cluster. In the equations below, α and β are hyperparameters in the suggested range of 0.01 to 0.05. We selected midrange values of 0.03 for both (in line with what the original authors chose).

Taken from the original paper, the probability of being added to a current cluster is expressed as:

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

And the probability of being added to a new cluster is:

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

The nomenclature in the equations above follows. The variables used are: ”p” for probability, ”z” for the cluster number, ”d” for the set of documents in a cluster, ”m” for the number of documents, ”n” for the number of words in reference to a cluster, ”D” for the number of recorded documents (total), ”N” for the number of words in reference to a document, ”K” for the total number of clusters, and ”V” for the size of the vocabulary. The subscripts used are: ”d” when referring to a document, and ”z” when referring to a cluster. There is only one superscript used, which is ”w” and denotes the portion of the data that only applies to a specific word.

As an example, an ”N” with a subscript ”d” and a superscript ”w” would indicate the total number of a specific word in a particular document. In another example, an ”m” with a subscript ”z” in addition to a subscript ”not d”, would indicate the number of documents in a particular cluster except for the one specific document.

3 Proposed algorithms

3.1 MStream

MStream is the baseline configuration of the algorithm described in [1]. It assumes that the streaming documents come in one by one and that we can only process each document one at a time in the single-pass implementation. Importantly, iterating is not allowed and the documents themselves are not retained in this single-pass version. MStream can be further utilized in a batch scheme, and this is the manner in which the algorithm was applied in our implementation. Using a batch scheme will allow us to iterate over a small group of documents before adding their data to the clusters, improving the fidelity of the results. The documents are then discarded, and the program will move on to the next batch.

3.1.1 MStreamF

MStreamF adds an extra loop around the MStream batch implementation in which certain meta-data is retained to identify when a document is outdated so that its data can be deleted from the cluster in which it resides. This deletion process occurs after each batch is processed. The MStreamF function is simply a wrapper around a call to the MStream function that contains this extra function-

ality; all of the document processing actually occurs exclusively in the MStream function.

4 Implementation

The MStream and MStreamF algorithms were implemented in a total of eight functions. Only two are actually named MStream and MStreamF; the remaining functions consist of additional helper functions and administrative functions (like reading in the input file), which round out the remaining six. All were programmed in Python, and the json, platform, math, copy, time, and random import libraries were utilized.

4.1 ReadData

The ReadData() function reads the short text from the input file, then generates a list of data. Each entry of the list is an array of size 4 containing the original cluster No. (data[i][0]), the cluster No. after classification (data[i][1]), the total number of words (data[i][2]), and a dictionary containing the frequency of each word (eg. data[i][3]["apple"]).

4.2 ComputeProbability

The ComputeProbability() function computes the probabilities of a document belonging to each of the existing clusters as well as a new cluster. These probabilities are computed using the specific statistics for this particular document, the data for all clusters, the size of the vocabulary for all currently recorded documents at the time the probability is calculated, the number of current recorded documents, and the two hyper-parameters α and β, by using the formulas listed in the original paper.

4.3 SelectCluster

The SelectCluster() function selects the cluster for a document based on the probabilities computed. It will first normalize the probabilities, then generate a random number within the range [0,1]; depending on which range this random number is in, the corresponding cluster will be chosen.

4.4 MStream

The MStream() function is structured in the same way that the MStream algorithm is described in the original paper. It decides the cluster of each short-text document in the input batch of data, and updates the information for the cluster to which the document is added. The clusters are maintained in a Python list and each entry denotes a cluster; each entry of the list is an array of size 3 containing the number of documents in the cluster (cluster[k][0]), the number

of words in the cluster (cluster[k][1]), and a dictionary containing the frequency of each word (eg. cluster[k][2]["apple"]).

This function requires seven parameters: a batch of input data (documents), the cluster information, the vocabulary set of currently recorded documents, the number of currently recorded documents, the total number of iterations for each batch, and the two hyper-parameters α and β.

This function calls the ComputeProbability() helper function in both the "One Pass" Clustering Process and the "Update" Clustering Process. In the One Pass Clustering Process, our implementation will update the vocabulary set and total documents data with the document to be processed directly prior to computing the probabilities.

4.5 MStreamF

The MStreamF() function is also structured in the same way as the MStreamF algorithm described in the original paper. It decides the cluster of each short-text document in the input data streams, and updates the cluster information.

This function requires six parameters: a list of batches of input data (the dimensionality of this data is one higher than the dimensionality of the input to the MStream function), the cluster information, the number of stored batches, the number of iterations for each batch, and the two hyper-parameters α and β.

This function keeps track of the vocabulary set and the number of currently recorded documents. When the cache for stored batches is full, our implementation will regenerate the vocabulary set with the remaining recorded documents and substract the number of deleted documents from the total documents number. There is no specific guidance on this in the original paper.

4.6 RunMStream

The RunMStream() function uses the MStream function to cluster the whole dataset. It requires seven parameters: the whole dataset, an empty list for cluster information, a list of indexes of the first document for each batch, a list of indexes of the end for each batch, the number of iterations of each batch, and the two hyper-parameters α and β.

4.7 RunMStreamF

The RunMStreamF() function uses the MStreamF function to cluster the whole dataset. It requires eight parameters: the whole dataset, an empty list for cluster information, a list of indexes of the first document for each batch, a list of indexes of the end for each batch, the number of stored batches, the number of iterations of each batch, and the two hyper-parameters α and β.

4.8 ComputeNMI

The ComputeNMI() function computes Normalized Mutual Information (NMI) metric for the data. This metric is widely used to evaluate the quality of clustering results [2]. It requires two parameters: the data to be evaluated and the clusters generated.

5 Dataset

Table 1: Statistics of the text datasets

Dataset	D	K	V	Avg Length
Tweets and Tweets-T	30,322	269	12,301	7.97
News and News-T	11,109	152	8,110	6.23

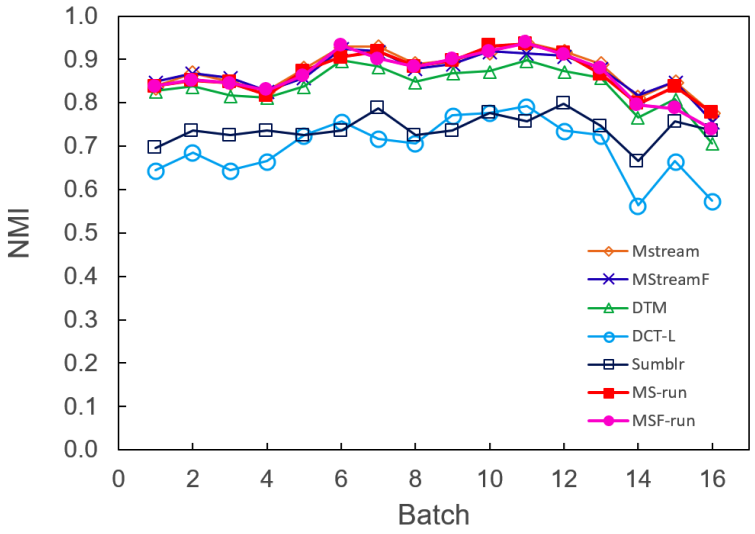
The same dataset has been used as the original research for our experiment which consists short-text document datasets (there are two variants: Tweets and News). The Tweets data has 30,322 tweets that are highly relevant to 269 queries in the TREC 2011-2015 microblog track. The News data consists of 11,109 news titles belonging to 152 clusters. Here, the "-T" appellation indicates that the data set is sorted and consolidated by Topic. Additional statistics are shown in table 1, where "D" is the number of documents, "K" is the number of ground-truth clusters (generated by researchers manually), and "V" is the size of the vocabulary.

5.1 Preprocessing

A certain amount of preprocessing has been performed on the raw Tweets and News data in order to ensure that only meaningful words are categorized, and multiple forms of the same words are categorized together. To this end, all characters have been converted to lowercase, stop words have been removed (e.g. a, an, the, etc. - words with little semantic weight), and stemming has been performed (words with similar roots collected together).

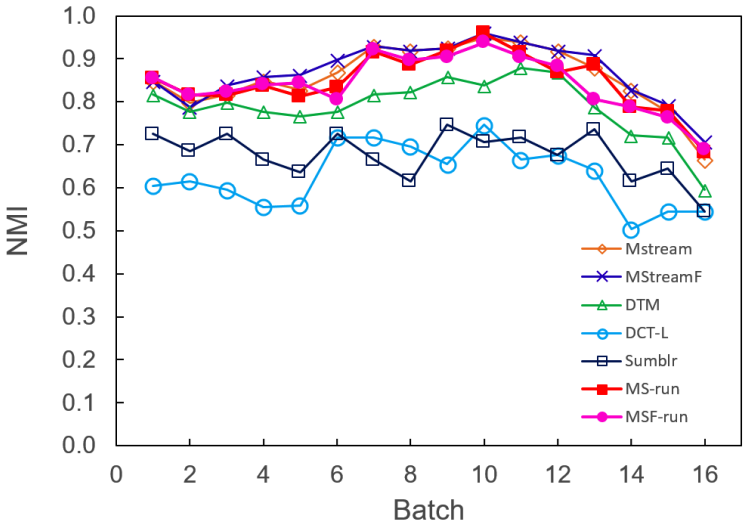
6 Results

Our results, shown in figure 1 through figure 4 and table 2 are labeled as "MS-Run" and "MSF-Run" to show the results from implementing the MStream algorithm and the MStreamF algorithm, respectively.



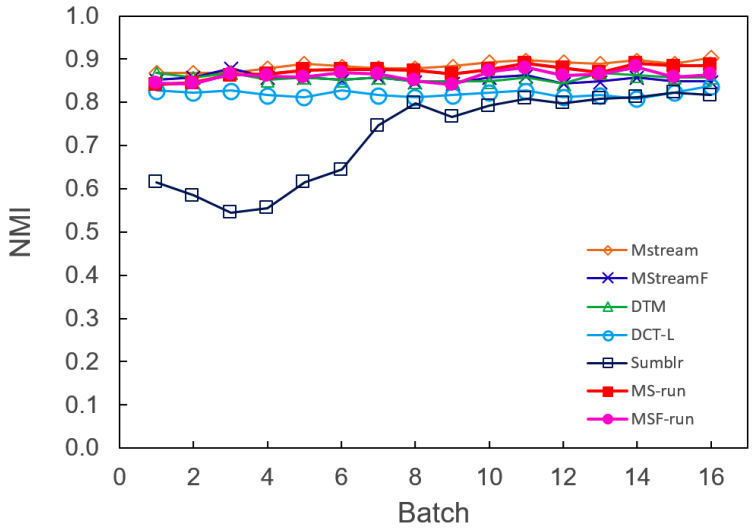
(a) Tweets

Figure 1: Tweets



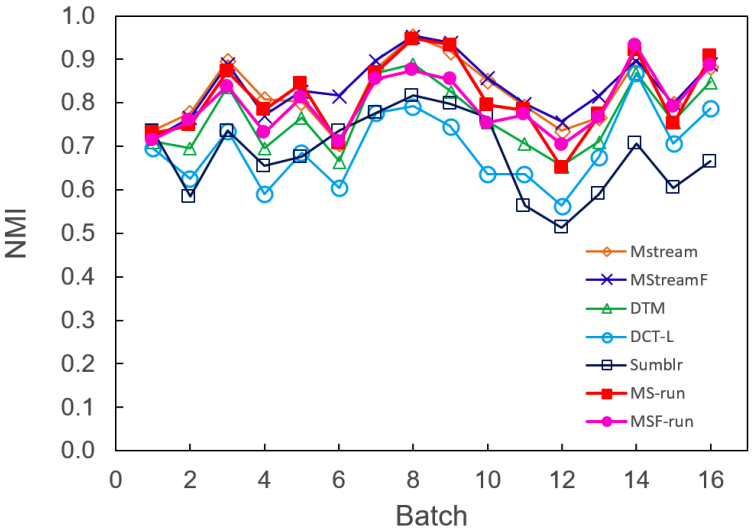
(b) Tweets-T

Figure 2: Tweets-T (topically sorted and grouped)



(c) News

Figure 3: News



(d) News-T

Figure 4: News-T (topically sorted and grouped)

There is some small variation between our results and the results from the original paper because there was no indication given regarding the particular psuedorandom number generation algorithm nor the random seed used. Our implementation uses the default settings from the "random" package, which generates a random float uniformly distributed in the semi-open range [0.0, 1.0) using an implementation of the Mersenne Twister as the core generator [3]. Ultimately, however, the results were highly comparable and in most cases resemble the results in the paper so closely that they obscure the original results on the plot.

Table 2: Average Results (our results are not shown with a range in this table because ours only represents a single run)

Algorithm	Tweets	Tweets-T	News	News-T
Mstream	0.844±0.002	0.882±0.004	0.834±0.004	0.850±0.004
MStreamF	0.823±0.005	0.923±0.003	0.797±0.003	0.873±0.003
DTM	0.801±0.002	0.802±0.001	0.793±0.002	0.806±0.002
DCT-L	0.697±0.002	0.669±0.005	0.733±0.002	0.744±0.004
Sumblr	0.689±0.001	0.695±0.003	0.575±0.005	0.720±0.002
MS-Run	0.868	0.848	0.872	0.814
MSF-Run	0.864	0.843	0.862	0.797

7 Conclusion

In this paper, we reviewed the short text stream clustering process based on the Dirichlet process multinomial mixture model (MStream). The proposed algorithm has the useful capablility that it can supposedly deal with the concept drift problem and use sparse text efficiently and naturally. The MStream algorithm did, in fact, show itself to be able to achieve state-of-the-art performance with only one pass of the stream, though it can show performance superior to that when multiple passes over well-defined batches are allowed. When the capability to "forget" (or delete) outdated clusters is added, the MStreamF algorithm can achieve even better performance when compared against the three baselines on real datasets. The claims by the original authors appear to be reproducible and accurately presented.

References

[1] Yin, J., & Wang, J. (2014, August). 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)**. ACM.

[2] Strehl, A., & Ghosh, J. (2002, July). Cluster ensembles-a knowledge reuse framework for combining partitionings. *In Aaai/iaai* **(pp. 93-99)**.

[3] 9.6. random — Generate pseudo-random numbers. (n.d.). Retrieved December 13, 2018, from https://docs.python.org/2/library/random.html

[4] Blei, D. M., & Lafferty, J. D. (2006, June). Dynamic topic models. *In Proceedings of the 23rd international conference on Machine learning* **(pp. 113-120)**. ACM.

[5] Shou, L., Wang, Z., Chen, K., & Chen, G. (2013, July). Sumblr: continuous summarization of evolving tweet streams. *In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval* **(pp. 533-542)**. ACM.

[6] Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine learning*, **39(2-3), 103-134**.