AN APPROACH FOR CLUSTERING SOCIAL MEDIA TEXT MESSAGES, RETRIEVED FROM CONTINUOUS DATA STREAMS

Assist. Prof. Bankov B.
University of Economics – Varna, Bulgaria
boris.bankov@ue-varna.bg

Abstract: *Using k-means clustering algorithm, a new approach to handle evolving topics and discussions in social media environment is proposed. Different segmentation techniques and applications to handle large volumes of data are explored. Relevant works that consider using fading functions and half-life weight measurements as a tool to remove inactive clusters are discussed. A set of rules and a controlling variable called time to recover are introduced as a simple means of managing cluster lifecycles. Short case study is conducted with Twitter data retrieved between the 19th and 22nd of January 2018.*

Keywords: SOCIAL MEDIA MINING, ONLINE STREAM CLUSTERIZATION, FADING FUNCTION, CONTINUOUS TEXT CLUSTERIZATION

1. Introduction

Text mining has become one of the major focuses of research in recent years. Due to the speed and volume of data that originates on the web, text processing techniques are evolving. Statistical algorithms are pivotal in dealing with variety of unstructured data management problems. To answer these problems clustering has been applied to continuous streams as means to analyze patterns in real time. Online clustering is being used in telecommunications, network surveillance, weather conditions monitoring, website traffic analysis and so on **[1]**. Processing unstructured textual data, retrieved from social media platforms however, is somewhat more difficult than traditional clustering tasks. When monitoring network traffic there is always a general idea of what processes or actions are within the regular norm and anything else can be treated as a threat. However social media text messages are often short, unpredictable in their format and can be semantically similar, while morphologically very different. Another issue is the fast emerging topics of discussion and their eventual fading out. To combat the challenges of streaming clustering several algorithms have been developed. In this paper we propose a new function to address the need evolving topics in text messages create, in terms of active and inactive discussions.

2. Overview of continuous text stream clustering algorithms

Clustering is a process aimed to organize and categorize data based on some traits. To cluster text one needs to transform documents, paragraphs or sentences into vectors. The way to represent text as an object in the Vector Space Model is to build a matrix of all words that are mentioned in the dataset and compare it to the number of times individual words appear in individual documents. Often to engineer such matrix tf-idf (Term Frequency – Inverse Document Frequency) is used.

A common algorithm for clustering is k-means and his variations – k-means++, OSKM (Online Spherical K-Means), Mini Batch k-means, k-medians, k-medoid, etc. The basic steps are as follows:

1) determine k for the number of clusters;
2) build k clusters and approximate their centers;
3) assign objects in high dimensional space to clusters, based on the distance between the objects and the closest cluster center;
4) repeat step 3 until it is impossible to move objects from one cluster to another.

According to different research **[2][3]** the definitive advantages of k-means are the low-cost implementation and high performance capabilities of the algorithm. K-means is very efficient in software environment because it only requires memory to maintain vector`s coordinates and calculate distance between them. When looking at continuous data streams it is important to handle incoming points and assign them to either existing clusters or create new ones. Aggarwal et al. **[4]** present one of the first ideas to split the incoming data stream into chunks. Microclusters are stored systematically at particular moments in time called snapshots. These segments are organized in a structure, resembling a pyramid, where each row contains snapshots of the active microclusters. On the top levels of this structure a small number of older snapshots remain, while lower levels represent recent microclusters. Row l contains segments, which appeared in a timeframe of 2^l of the stream. Any segment that is divisible by 2^{l+1} is removed from the row and does not generate a microcluster (see Fig.1).

row l	*time snapshots*
5	32
4	16 ~~32~~ 48
3	~~16~~ 24 ~~32~~ 40 ~~48~~
2	36 ~~40~~ 44 ~~48~~ 52
1	46 ~~48~~ 50 ~~52~~ 54
0	51 ~~52~~ 53 ~~54~~ 55

Fig. 1 *Structure of microcluster snapshot pyramid*

Ackermann et al. **[5]** proposes splitting the stream into small segments in order to reduce the number of dimensions. Initially *n* number of segments are present, that contain *m* number of data points. All segments following the first one contain between 0 and *m* points. The first segment is used as a safety net for incoming vectors. When a partition reaches its maximum capacity all points are moved to the next segment. If that one is full as well both partitions are paired into a new segment and cleared of all points, so that they can accept fresh data (see Fig. 2).

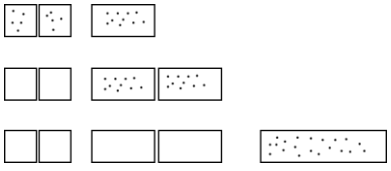

Fig. 2 *Structure of segmented data stream*

Zhong **[6]** presents an algorithm under the name of Online Spherical K-Means (OSKM) that deals with large volumes of data by taking into account both time and significance of data. The technique is aimed at splitting the incoming data into segments or chunks. Zhong`s algorithm loops through all the streaming data without any interruptions or pauses and thus results are kept in the machine memory. While segmented into chunks the dataset is easier to cluster because cluster centroid information can be read in runtime. In order to prioritize new data points Zhong proposes a

variable that measures the period of time cluster information is kept in memory. In the context of social media mining, information about recent messages is more valuable than older topics of discussion which tend to diminish rather quickly. The variable Zhong introduces is called a **decay factor** and is used to determine the historical weight of existing clusters. To calculate the weight, a decay factor between 0 and 1 is accepted and then multiplied by the number of vectors in the cluster. The decay factor is reciprocal to the period of time the cluster has existed and eventually becomes infinitely small.

Similar idea can be found in the works of Aggarwal and Yu **[7]**, who introduce a **fading function** that determines cluster weight while taking into account the last moment in time a data point was assigned to a specific cluster. That way clusters that are not accepting new vectors are labeled as inactive. To manage active and inactive clusters Aggarwal and Yu also rely on a variable that measures the time needed for a data point to lose half of its weight or significance – **half-life** variable. Half-life is also used when a new vector is introduced in the high dimensional space and it is labeled as an outlier or an anomaly. If within one half-life period another point close to the first one is delivered via the stream, then the anomaly is accepted as a new cluster. If no such vector is encountered the anomaly is treated as such and no new cluster is formed. Following this approach any clusters that are inactive for a period of time greater than their half-life are nullified or broken down.

Modern algorithms consider other metrics as well, but the main basis for developing fading functions are the variables time and significance or weight of clusters. We consider Zhong and Aggarwal to be a starting point for creating a modest rule set that can be applied in continuous stream text clustering. Our goal is to have an automated process that manages active and inactive clusters in the context of developing and diminishing topics in social media platforms.

3. Period of recovery for inactive clusters

To efficiently cluster text data, retrieved from social media platforms in real time, it is necessary to understand user behavior and how discussions happen. Let us assume there are two general reasons for users to post anything online – the first is a natural inclination to share information that comes from within and the second is provoked by events or people that happen without our direct involvement. In general the majority of people participate in more than one online chat group or comment on more than one subject. Politics, natural disasters, sports outcome or entertainment news can all spark discussions that last days, weeks or months. Some less important occurrences might go unnoticed if we only look at the big picture and decide to cluster data only in predetermined topics. Furthermore it is almost an impossible task to predict all the possible categories of human interest on social media which means creating a static distribution of vectors into cluster group is not an effective solution. A possible solution is to label clusters as active, inactive and nullified, based their and how they develop in time.

We represent time with three variables:

1) t_a – time, as a sum of all periods of activity for a specific cluster;
2) t_s – time, as a sum of all periods of inactivity for a specific cluster
3) T – time, that expresses the "life" of a cluster, as a sum of t_a and t_s;

Active is a cluster that continues to accept data points in his vicinity. It also has a sum of periods of inactivity lesser than the sum of periods of activity. Inactive is a cluster that in two successive moments in time does not get any vectors assigned to it. If none of the clusters receive any new points, then their t_s does not increase as to account for stream inconsistencies. Apart from that another reason could be an outlier or an anomaly that is actually a new topic. A cluster is nullified when after being inactive is terminated and no longer exists in vector space. So for a cluster *K* if (1) is true we consider it active and we keep it intact.

$$t_s < t_a \tag{1}$$

A simple rule like that can be easily calculated in runtime without a significant impact on performance. To account for new emerging clusters we also need to take note of the number of objects that enter the high dimensional space. In order to preserve the inequation (1) and address historical significance it is important to look at how time and cluster weight are related. Because both variables are not homogeneous first we take note of the number of data points that enter vector space while a cluster is alive. Furthermore we choose to look at the average rate at which high dimensional objects are sent via the data stream and are assigned to all present clusters. So if there are *N* total points in vector space, then on average $\frac{N}{T}$ points are assigned to clusters per time frame. Therefore the product

$$\frac{N}{T} * t_a, \qquad T \neq 0 \tag{2}$$

gives us the approximate maximum number of points that cluster *K* could have in his vicinity. On the other hand the product

$$\frac{N}{T} * t_s, \qquad T \neq 0 \tag{3}$$

can be used to measure the approximate maximum number of points that are outside of *K* and are assigned to other clusters. If we take a look back at (1), add (2) and (3) we keep the original inequation.

$$\frac{N}{T} * t_s < \frac{N}{T} * t_a, \qquad T \neq 0 \tag{4}$$

Now however both sides represent number of vectors rather than sum of periods of time. From a theoretical stand point such rule can be only applied if cluster *K* has a number of points equal or greater than all other clusters in the high dimensional space. Statistically it is more likely that the inequation (4) is untrue for the majority of the data stream. To summarize, it is not possible to determine if a cluster should be active without accounting for the significance on the objects assigned to it. In addition, important clusters need a **period of recovery t_r** that allows them to remain inactive before completely being nullified. As a general rule, clusters which are not historically important or which do not assimilate enough data points are subject to being nullified after long inactivity.

Under usual circumstances, when t_s becomes greater than t_a the recovery period will act as a balancing variable that factors in cluster significance. Cluster unit weights can be calculated with tf-idf, word2vec probabilistic distribution, chi squared and others **[8]**. Assuming ω is the weight of a cluster *K* and *n* is the number of data points that are assigned to it, $\frac{\omega}{n}$ results in an average weight per point. Because (2) gives us the maximum possible amount of objects cluster *K* can receive for a period of activity t_a, with formula (5) we can determine an approximate maximum significance ω'.

$$\omega' = \frac{N}{T} * t_a * \frac{\omega}{n}, T \neq 0, n \neq 0 \tag{5}$$

When calculating the recovery period t_r we will take into Aggarwal and Yu`s idea of a half-life measure (the period of time it takes for a unit to lose half of its significance). However we are choosing to measure the time it takes for the real significance ω to decrease to a value of $\frac{1}{2} * \omega'$. As a result the closer ω to ω' or the better a cluster performs in terms of receiving important and relevant data points, the bigger the period of recovery t_r would be.

On the other hand if there is a large difference between the values of ω to ω', that would indicate that cluster *K* has been performing poorly and its significance is lower than expected. To illustrate this better we propose the following example.

In a coordinate system, where the abscissa represents cluster weight and the ordinate represents time, we create points for $\omega = 10$ (see Fig. 3) and $\omega' = 15$ (see Fig. 4) against a period of activity $t_a = 10$. We choose to gradually increase ω from point A to point B, in such a way that resembles an identity function with the sole purpose of facilitating an easier comprehension (e.g. a value of 1 weight per 1 moment of time). As soon as point B is reached, the period of inactivity t_s also starts increasing until it becomes equal to t_a. At this moment a period of recovery t_r should be measured and extended to cluster *K* to either become active again or be nullified.

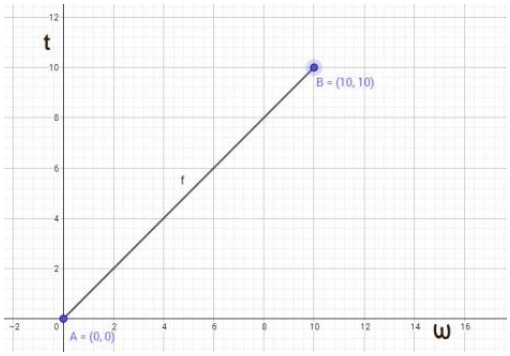

Fig. 3 *Real significance of cluster in B*

Let us assume that during the period of activity of cluster *K* the number of points that are introduced in vector space is equal to three times the number of vectors in that particular cluster ($N = 3 * n$). Referring back to (5) we calculate ω'

$$\omega' = \frac{N}{T} * t_a * \frac{\omega}{n} = \frac{3 * n}{t_a + t_s} * t_a * \frac{\omega}{n} = \frac{3 * n}{10 + 10} * 10 * \frac{10}{n} = \frac{3 * 10 * 10}{20} = 15 \quad (6)$$

Fig. 4 *Approximate maximum significance of a cluster in B*

The time it would take for ω to reach $\frac{1}{2} * \omega'$ we can measure either with a custom multiplier or as it is in our example with a decay factor $\gamma = 1$ of 1 weight point loss per 1 moment in time.

$$t_r = \left(\omega - \frac{1}{2} * \omega'\right) * \frac{1}{\gamma} = 2.5,\ \gamma \neq 0 \quad (7)$$

Segment DE on Fig. 5 represents the time to recover t_r needed for ω to reach $\frac{1}{2} * \omega'$. Surpassing a period of 2.5 without any new vectors entering the cluster`s vicinity would result in the nullification of said cluster.

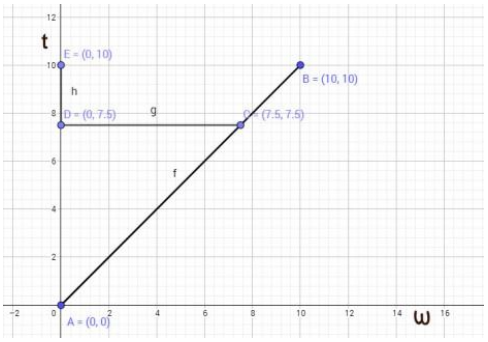

Fig. 5 *Period of recovery measured in segment DE*

We assumed that the number of data points introduced in vector space was 3 times bigger than the number of points assigned to cluster *K*. A question might arise as to why we did not choose $N = 1.5 * n$ or $N = 2 * n$ for example. As a matter of fact any number lower or equal to $2 * n$ would mean that cluster *K* either contains the majority of points or has an amount equal to the sum of all points that do not belong to it. If that is the case then we can use (4) to determine if it needs to be nullified.

A larger the amount of data points in vector space as well as a higher value of individual cluster`s significance will result in a longer period of recovery being issued. On the other hand a smaller number of points in the cluster alongside a higher life period will be followed by a shorter recovery period.

4. Results and discussion

To test our algorithm we choose to download data from Twitter using the platform`s Streaming API published in the span of 4 days between 19th and 22nd of January. Twitter supplies researchers with a rich and structured output that contains plenty of labeled data such as publishing date, location, user information and so on. We choose to retrieve text messages that are mainly in Bulgarian. As a result we received a little over 50000 messages before applying sanitization filters. Because clustering does not offer one single solution and needs to be run multiple times with different parameters such as batch size, maximum number of iterations, number of clusters, etc., we choose to look at 5 to 10 clusters and compare results. In our dataset there were 4 distinct and 1 free-for-all categories. Over the period the major topics that dominated chat messages were Boyko Borissov, Angela Merkel, the Istanbul convention and traffic accidents (see Table 1).

Table 1: *Number of data points in the clusters for the whole period*

Date\Topic	Merkel	Borissov	Istanbul	Accidents
2018.01.19	441	352	480	233
2018.01.20	1385	755	300	216
2018.01.21	693	603	240	210
2018.01.22	445	537	1080	370

However when looking at individual data distribution throughout the day two of the big clusters existed alongside smaller ones which were nullified and assimilated after they stayed inactive for too long. An underlying topic found within discussions about Angela Merkel was the Schengen Area agreement (see Fig. 6), which was more of a secondary subject in user messages and did not outperform the German Chancellor. The second theme that did not have a strong representation on its own was the Pirin mountain discussion that often was mentioned in tweets addressing Bulgarian Prime Minister (see Fig. 7).

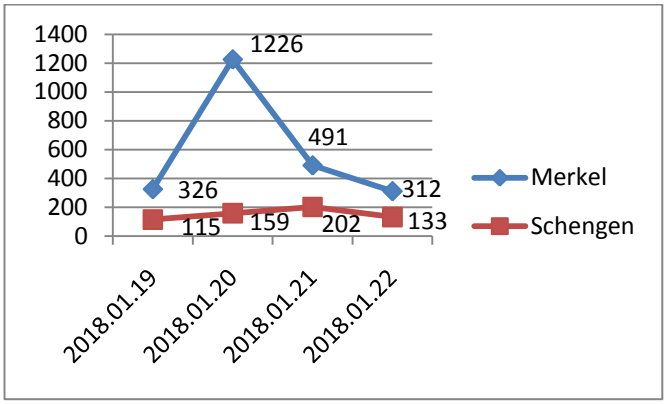

Fig. 6 *Number of points in individual clusters*

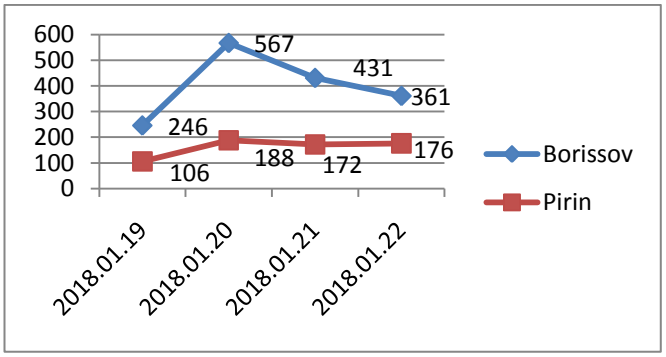

Fig. 7 *Number of points in individual clusters*

5. Conclusion

All clustering algorithms are aimed at summarizing vast amounts of data. When it comes to categorizing live streams techniques have been evolving in the past decades. Sometimes data can be hard to evaluate if we only look at the big picture. In this paper we presented a fast and easy to use set of rules that can be applied in continuous clustering. Determining the time to recover value does not significantly complicate the process. Although a fairly simple concept, our approach can be easily implemented as it does not require heavy computations.

6. References

[1] Mousavi, M., A. Bakar, M. Vakilian. Data stream clustering algorithms: A review. *International Journal of Advances in Soft Computing and its Applications*. vol. 7, no. 3, 2015, pp. 1-153.

[2] Kwale, F. An Overview of VSM-Based Text Clustering Approaches. *International Journal of Advanced Research in Computer Science*. vol. 5, no. 1, 2014, pp. 69-73.

[3] Jing, L., L. Zhou, M. Ng, J. Huang. Ontology-based distance measure for text clustering. *SIAM SDM workshop on text mining*. 2006.

[4] Aggarwal, C., J. Han, J. Wang, P. Yu. A framework for clustering evolving data streams. *Proceedings of the 29th international conference on Very large data bases*. vol. 29, 2003, pp. 81–92.

[5] Ackermann, M., M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, C. Sohler. StreamKM++: A clustering algorithm for data streams. *Journal of Experimental Algorithmics (JEA)*. vol. 17, 2012, pp. 2-4.

[6] Zhong, S. Efficient Streaming Text Clustering. *Neural Networks*, vol. 18, no. 5-6, 2005, pp. 790-798.

[7] Aggarwal, C., P. Yu. A Framework for Clustering Massive Text and Categorical Data Streams, *SIAM Conference on Data Mining*, 2006, pp. 479-483.

[8] Lan, M., S. Sung, H. Low, C. Tan. A comparative study on term weighting schemes for text categorization. *Neural Networks*. vol. 1, 2005, pp. 546-551.