Homophilic Self Organizing Feature Maps: Topic Detection on Socially Connected Data

Bernardo Carvalho Simões
Instituto Superior Técnico
bernardo.simoes@ist.utl.pt

***Abstract*—With the evolution of social media platforms, the amount of unlabeled information has gone skyrocketing, the process of labeling this kind of information evermore complex. Typical approaches used on the WEB for Topic Detection and Tracking cannot be directly applied due to the small amount of text produced per tweet, orthographic errors, abbreviations and so on.**

We propose and analyze a new form of topic detection and tracking on social networks. By leveraging the social relations between authors of the gathered content, and apply them to the clustering process.

In order to achieve this, we proposed some modifications to the artificial neural network and clustering algorithm — Self Organizing Maps.

***Index Terms*—topic detection, twitter, self-organizing maps, classification, clustering**

I. Introduction

FINDING topic sensitive information on social networks is extremely complicated due to the fact that documents have very little content, slang vocabulary, orthographic mistakes and abbreviations. Asur and Huberman [2] successfully predicted box-office revenues by monitoring the rate of creation of new topics based on debuting movies. Their work outperformed some traditional market-based predictors.

Thus, academic and enterprise worlds started looking at Machine Learning (ML) for new ways to achieve revenue or simply explore and discover patterns in data.

Using unsupervised ML, Le et al. [13] was able to achieved 81.7% accuracy in detecting human faces, 76.7% accuracy when identifying human body parts and 74.8% accuracy when identifying cats.

Social Media Analytics is another raising topic that draws from Social Network Analysis [10], ML, Data Mining [26], Information Retrieval (IR) [19], and Natural Language Processing (NLP). As stated Melville et al. [15], 32% of the 200 million active bloggers write about opinions on products and brands, while 71% of 625 million Internet users read blogs and 78% of respondents put their trust in the opinion of other consumers. In comparison, traditional advertising is only trusted by 57% of consumers. This kind of data drives companies to Social Media Analytics as a way to know what people are saying on the web about their companies and products. This new worry has brought to life a lot of new startups like Sumalor ThoughtBuzz, but also solutions from the old players like IBM and SAS.

The main objective of this project is to find topics on tweets by contextualizing the social network involving the person that authored the tweet in the clustering process.

We start by building a dataset, in order to train the Self-Organizing Maps (SOM), that will later classify each future tweet that arrives on the network.

After creating the dataset, we will try to find clusters of topics using the default SOM approach, converting each tweet to Vector Space Model (VSM). After analyzing the results from the default SOM approach, the algorithm will be changed in order to give relevance to the social relationship between authors of tweets.

II. Related Work

A. Clustering with Self-Organizing Maps

1) The Self-Organizing Map: SOM are a two layer recurrent Artificial Neural Network (ANN) that has the desired property of topology preservation, thus mimicking the way cortex of highly developed animal brains work. SOM allow cluster visualization of multi-dimensional data, similar to methods such as Multi Dimensional Scalling (MDS) [12] and Principle Component Analysis (PCA) [9] .

Bação et al. [3] described the basic idea behind SOM as a mapping between input data patterns into a n-dimensional grid of neurons, or units. That grid is also know as the output space, as opposed to the initial space — input space — where the input patterns reside. An illustration of both spaces can be seen in Figure 1.

SOMs work in a similar way as is thought the human brain works. Analogously to the human brain, SOMs also have a set of neurons that, through learning experience, specialize in the identification of certain types of patterns. These neurons are responsible for categorizing the input patterns for which they are responsible to identify. Nearby neurons will be organized by similarity, which will cause similar patterns to activate similar areas of the SOM. With this topology preserving mapping, the SOM organizes information spatially, where similar concepts are mapped to adjacent areas. The topology is preserved in a sense that, as far as possible, neighborhoods are preserved throughout the mapping process. Output neurons are displayed in an n-dimensional grid, generally rectangular, but other structures are possible, such as hexagonal or octagonal. The grid of neurons, in the output space, can be divided in neighborhoods — where neurons responsible for the same kind of input reside. In SOM, neurons will have the same amount

of coefficients as the input patterns and can be represented as vectors.

Before describing the algorithm, it is important to define two key aspects of the SOM: the learning rate and the quantization error. The learning rate is a function that will be decreased to converge to zero. It will be applied to winning neurons and their neighbors in order for them to move toward the corresponding input pattern in progressively smaller steps. Quantization error is the distance between a given input pattern and the associated winning neuron. It describes how well neurons represent the input pattern. The radius of the neighborhood around the winning neuron is also particularly relevant to the topology of the SOM, deeply affecting the unfolding of the output space as stated by Bação et al. [3].

In order to know how well a neuron maps to all the input patterns it represents, the average of the quantization error can be used(Eq. 1). On the equation, $d_{i,n}$ is an input pattern that is represented by the neuron w. Each neuron represents an arbitrary number — n — of input patterns. That group of input patterns is represented as $D_{i,j}$.

$$\varepsilon(w) = \frac{\sum_{i=0}^{n} \|w - d_i\|}{n}, d_i \in D, \forall n \tag{1}$$

Algorithm 1: Self-Organizing Map [11]

Data: Input patterns $X = \{\overrightarrow{x_1},\ldots,\overrightarrow{x_N}\}$, number of iterations t_{max}, neighborhood function $\sigma(t)$, learning rate $\epsilon(t)$

Result: Trainned map and clustered input patterns

Randomly initialize neurons, $w_i \in \mathbb{R}^D, \forall i$

for $t = 1$ *to* t_{max} **do**

Randomly draw an input pattern, $\overrightarrow{x_d}$

1 $p = \arg min_i\{\|\overrightarrow{x_d} - \overrightarrow{w_i}\|\}$

2 $\overrightarrow{w_i} = \overrightarrow{w_i} + \epsilon(t) \cdot h_{ip}(t) \cdot (\overrightarrow{x_d} - \overrightarrow{w_i}), \forall i$

3 $\sigma(t) = \sigma_0(\sigma_f/\sigma_0)^{t/t_{max}}$

4 $\epsilon(t) = \epsilon_0(\epsilon_f/\epsilon_0)^{t/t_{max}}$

5 $t \leftarrow t + 1$

The learning phase is characterized by the Algorithm 1, which works the following way:

- **On line 1:** The neuron closer to the input pattern is selected. The Euclidean distance (Eq. 2) is generally used.

$$Dist = \sqrt{\sum_{i=0}^{i=n}(V_i - W_i)^2} \tag{2}$$

- **On line 2:** the winning neuron (p) previously selected on line 1 is updated, in order to better represent the input pattern. Also, all other neurons inside a specific radius will also be updated — this process is described in Figure 1. Each neuron is updated with a different rate of influence determined by how far away it is from the winning neuron, which is defined by the neighborhood influence function $h_ip(t)$. The Gaussian (Eq. 3) is often used.

$$h_{ip}(t) = \exp -\frac{|\overrightarrow{a_i} - \overrightarrow{a_p}|^2}{\sigma^2(t)} \tag{3}$$

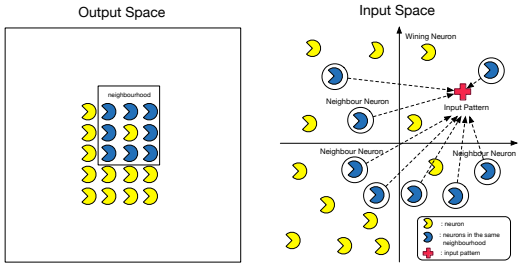

Fig. 1. On the left the output space neighborhood, on the right the neighbors of the winning neuron converging in the direction of the input pattern

- **On line 3:** the size of the radius will be updated.
- **On line 4:** the learning rate is updated.
- **On line 5:** the number of iterations is incremented.

In order for the algorithm to converge, the learning rate and the radius of the neighborhood need to decrease at a given rate. This process can be seen on line 3 and 4, respectively .

The prediction phase can start after the model is learned. On the prediction phase, new input patterns can be quickly assigned to the SOM, without need to apply the learning rate to the winning neuron and his neighbors. In other words, only line 1 will run. Due to the fact that the input pattern will be assigned to the cluster that is mapped by the nearest neuron. Thus, it is very easy and fast to classify new data now. As stated by Liu et al. [14], the advantages of using SOM are: data noise immunity, easy to visualize data, and parallel processing.

In order to visually interpret the result of the SOM, Unified Distance Matrix (U-Matrix) method may be used [3]. The U-Matrix is a representation of the SOM, in which the distance between neurons is represented in a gray-scale where the darkest colors represent the farthest distance and the lightest colors the closer neurons.

2) The Geo-SOM: The Geo-SOM, by Bação et al. [3], applies the first law of geography "Everything is related to everything else, but near things are more related than distant things." [21] to the SOM algorithm. In this case, the winning neuron is chosen in a radius defined by the geographic-coordinates of the data, forcing units that are close geographically to be close in the output space.

The algorithm works by defining a variable k which is used as a "geographical tolerance" that forces the winning neuron to be geographically near the input pattern. When $k = 0$, the winning neuron is forced to be the unit geographically closest to the input data, whilst k increases, the tolerance for data with further geographic coordinates, increases as well. k is a geographic radius applied in the output space. When the radius exceeds the size of the output space, every unit is eligible to be the winning neuron, and therefor, we have a regular SOM.

The selection of the winning neuron is done in two steps. First, geographic neurons inside the tolerance k with the input data as a center are selected. Only after that, comparisons are made with the rest of data present in the input data. The representation of the Geo-SOM can be seen in Figure 2, where the units considered for the best match are defined by a sort

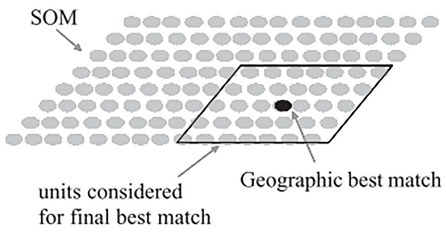

Fig. 2. Geo-SOM structure, where the units considered to be the winning neuron are constrained by the geographic coordinates of the data, from Bação et al. [3]

of geographic radius defined by k, whilst in the original SOM, the winning neuron could have been any of the units presented on the figure.

The Geo-SOM approach to the alteration of the default SOM algorithm is specially interesting due to the fact that this thesis objective is also to give relevance to data patterns that are not located in the same space as the trained data. In a way, what we are trying to achieve is similar to the work by Bação et al. [3] but changing the geographic relevance in data by a social relevance.

3) WEBSOM: Honkela et al. [8] developed a new approach to automatically order arbitrary, free from textual, document collections, using two different SOMs. The first SOM is called word category map and it's used to find words that have similar meaning, while the second SOM, called document map, is the one actually used to cluster the documents.

The WEBSOM was not based on keywords and boolean expressions, instead, words with the same meaning are encoded in a word category map, where placement and frequency in documents is taken into account. This way it is possible to remove words with similar meaning — greatly reducing the VSM size making it possible to train the document map in a scalable way.

B. Twitter Data Mining and TTD

In this subsection, we will focus on work done on the Twitter social network in order to leverage insights on how the public data available from the website can be explored.

1) Topic and Trending Detection: Allan [1] defined Topic Detection and Tracking (TDT) as "a constantly arriving stream of text from newswire and from automatic speech-to-text systems that are monitoring selected television, radio, Web broadcast news shows. Roughly speaking, the goal of TDT is to break the text down into individual news stories, to monitor the stories for the events that have not been seen before, and to gather stories into groups that each discuss a single news topic".

Nowadays, due to the rapid adaptation of people to always be on-line, through the usage of cellphones on the move, desktops at work and even TV at home, the increase of user generated content has increased tremendously in latest years. In 2006, 35% of on-line adults and 57% of teenagers created content on the Internet [1], which in "Internet Years" was ages ago.

The challenge of TDT is evermore focused on online generated documents, and in new forms to be able to track and categorize all the information that is continuously being generated. Many TDT techniques have been proposed, a significant amount of them rely on the Term Frequency–Inverse Document Frequency (TF-IDF) [4]. Because tweets are very small, often with typos or slang words, and because the same tweet might be written in multiple languages, TF-IDF is not particularly adequate for topic detection on twitter. In this subsection, we will take a look at multiple methods of topic detection in general, and also specifically on the Twitter social network.

Cataldi et al. [6] proposed a new technique for emerging topic detection that permits real-time retrieval of the most emergent topics expressed by a community on Twitter. Their work applies the PageRank algorithm [17] to the users follower/followee relationship, in order to find the most influential users on the network. Then, the most trending topics are calculated, by relating social influence, word co-occurrence and time frame. In the end, an interface was created where it would be possible to navigate, through hot topics in a given time frame. Topic labeling was not automatic and was implicit by the time frame of an event.

Weng et al. [24] also used the PageRank algorithm to find the most influential twitter users on a certain topic. However, using a different approach, they represent each twitter user as a bag of words comprising of all the tweets that they have posted, and applied Latent Dirichlet Allocation (LDA) [5] in order to find topics in which users are interested in. Finally, it was possible to prove that follower/followee relations on twitter are not just casual, but that people actually follow other people to whom they have some resemblance or common interest. This concept is called homophily and will be further explored on this thesis.

2) Tweeter Natural Language Processing: Using standart NLP tools on tweets has been extremely unreliable, due to the fact that microbloging text tends to be full of abbreviations, emojis and smiles . Recently, Owoputi et al. [16] published a NLP library, specific for twitter. ARK Tweet NLP can tag words that are only used in social networks. The tagger was built using maximum entropy Markov model, where a tag is assigned to a word based on the entire tweet text, and the tag assigned to the word to its left. Owoputi et al. [16] state that the tagger has a 93.2% accuracy. By using NLP tools, it is possible to reduce the dimension of VSM space by only choosing words that are relevant, like common nouns, hashtags and proper nouns. This will not only yield better results by removing tweets that have no content, and therefor, cannot be categorized, but will also increase performance during training due to the reduced dimensions caused by less use of words.

[1]Data source: http://www.pewinternet.org/Presentations/2006/UserGenerated-Content.aspx

III. Solution

A. Clustering Tweets

In order to use SOM to cluster tweets, first the tweets need to be converted into VSM. Given the fact that tweets are often misspelled, with slang words and are written in multiple languages, the VSM tends to become pretty large with relative ease.

In order to reduce the amount of different words that could have the same meaning, or no meaning at all, the following rules were applied:

- Only English tweets were used during clustering.
- Uniform Resource Locator (URL) are removed. Since most of them are minimized, little information can be taken from them without domain translations.
- Numbers are removed.
- All letters are down cased.
- Runs of a character are replaced by a single character.
- Words smaller than 3 chars are discarded.
- Stop words are removed.
- The tweet text is stemmed.

By applying these rules, the VSM is greatly reduced without destroying major relevant words. More information about VSM reduction can be found on Sub-chapter IV-B. A visual application of these methods can be seen in Figure 3.

Since tweets are very small and have an average of only 10 to 14 words, there is no need to store term frequency on the VSM, and therefor only a binary count is made.

NLP techniques such as the one described on Subsection II-B2 can be used in conduction of the method described above for even a more efficient VSM reduction. In order to accomplish this, first we need to run the NLP on the dataset and specify what kind of tags we want to use. Then, we run the string reduction techniques described above on the words tagged by the NLP, these words will then be used to create the VSM where each word represents on column. This process can be seen in Figure 4.

Converting tweets from text to VSM can be done in two different approaches. The first one is the cumulative approach, where the VSM is being built at the same time that the tweets are read, new terms are added as columns to the VSM as soon as they are found. The second way relies on scanning all words present in the dataset in order to first build the VSM, then iterate through all tweets and mark them as ones and zeros if they occur in the tweet text.

After the VSM is filled with tweets, it can be feed to the SOM and therefor training can start. It is important to notice though, since a destructive process was done to minimize the size of the VSM some extra mechanisms must be implemented in order for the tweets to be humanly readable after training.

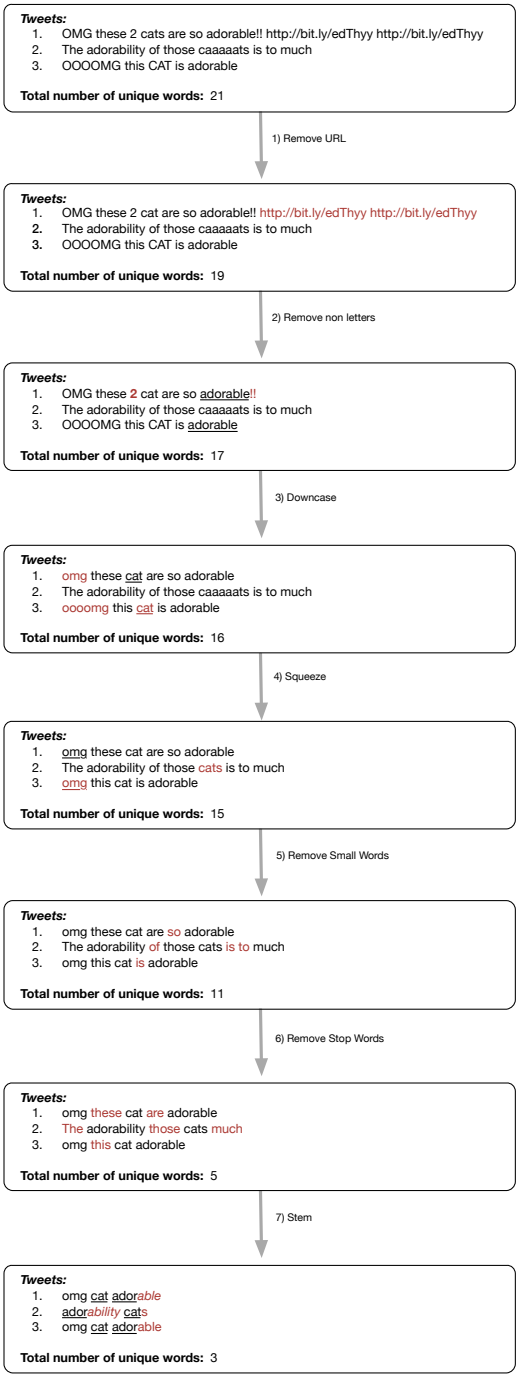

Fig. 3. Reducing the number of unique words on three tweets about cats. Text in red represents letters removed. Underlined text represents words that due to text transformation became equal.

B. Extensible SOM Library

When researching ways to extend the SOM algorithm, by adding social features to the learning process, we found that there weren't many SOM software libraries. Even though, programing languages often used in ML and Data Mining, such as Python or C++, have their own implementation of the SOM algorithm. We've found that most of these libraries are made in such a way to be extremely fast, in order to take as much advantage from the hardware they are running on as possible. They often lack the modularity needed to adapt the SOM algorithm to specific problems.

The SOM algorithm has been changed many times in order to better categorize data with specific features. For example, the previously described in Subsection II-A2 Geo-SOM, the Growing Hierarchical SOM [18], the time adaptive SOM [20],

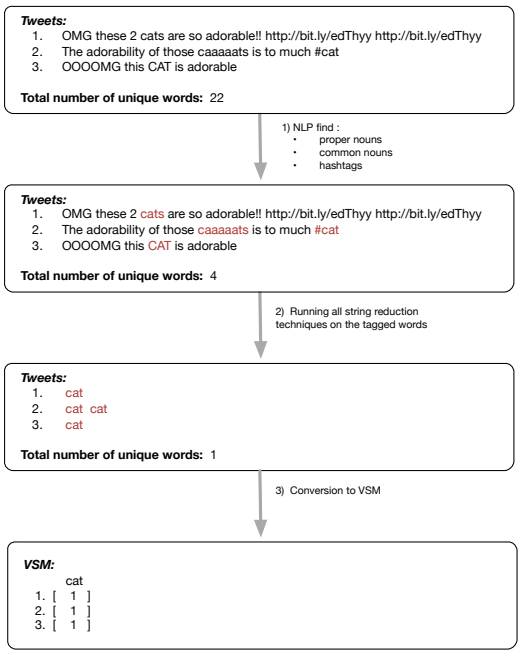

Fig. 4. Using NLP with string reduction techniques to reduce the VSM size

the Ontological SOM [7], and the list goes on...

The SOM framework is an open source ruby library for creating custom SOM implementations. The SOM framework, implements the basic SOM algorithm with a squared output space, is readably available. Any kind of data which implements enumerable — can be treated as arrays — can be used as input patterns.

It is possible to print U-Matrix, Unified Mean Quantization Error Matrix (Q-Matrix) at any given time of the training, as well as inspect the topological error. In case the output space is represented as colored vectors, it is also possible to print the current color of the output space at each iteration.

In order to create the homophilic SOM, described in Section III-C we first created a SOM framework that is easy to extend due to be fully object oriented, scripted — even though it can be compiled to run on the JVM — and without C extensions.

C. Homophilic SOM Definition

The default SOM algorithm has no idea whatsoever of the social connections between the tweets, it simply looks at the binary vectors that represent sentences and assigns it to the most similar neuron.

In order to better categorize socially connected data, we propose some alterations to the SOM algorithm in order to make it aware of the social connections between the tweets, and therefor, better represent the homophilic behavior present on social networks.

1) Output Space: The output space is the zone on the SOM algorithm where the neurons reside. It works like a cortex where neurons are scattered in a geometric fashion, generally a square. The output space is generally initialized with random values, with a relatively high learning rate, and also a relatively high number of epochs. The algorithm is made this way in order to be able to identify any type of data that can be represented as vectors.

First, we will try to change the output space to better resemblance the social network. In order to do this, the squared grid that defines the output space was changed by the social network connections, and the neurons, are represented by a social network user. This changes are applied in the following way:

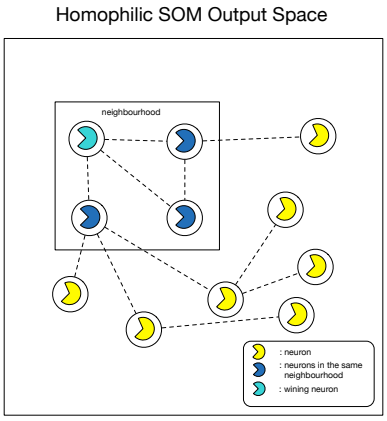

Fig. 5. The neighborhood is defined by the relations of followers/followees between the winning neuron and the other neurons

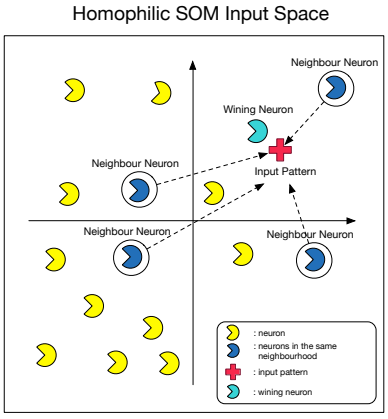

Fig. 6. Homophilic input space works in the same way as a normal input space

- Each neuron is comprised of the text from all the tweets that he authored.
- Each neuron has a unique id, and stores the ids of his followers and followees that are present in the output space.
- During the learning phase, the radius will be defined as the maximum number of hops separating the winning neuron and followers/followees of followers/followees.

2) Learning Phase: Like in the default SOM, the learning phase is where the output space is trained in order to organize the input data into clusters. Since this algorithm is specific to categorize tweets using social network features, the learning

rate, radius and number of epochs used can be greatly reduced in order for the algorithm to converge. The learning phase operates in the following way:

- The distance between the input pattern and all the neurons is calculated. The neuron closest to the input pattern is considered the winning neuron.
- When the winning neuron is selected, it and its social neighbors within k hops, update their representations in the input space, and move closer to the input pattern. The Gaussian function (Func. 3) is also used here. As a way for the neighbors that are closer to the winning neuron, be significantly more influenced by the input pattern, while the neurons further away are less influenced.
- This process is repeated for a predefined number of epochs. In order for the algorithm to converge, whilst the number of epochs increases, the learning rate and number of hops that defines the neighborhood decreases.

Just like the default SOM algorithm, after the map is trained, input patterns can be fast assign to the nearest neuron since the neuron positions in the output space are no longer updated.

IV. Result Assessment

A. SOM training

Our first approach to cluster tweets with SOM started by dynamically creating VSM for each new word that was encountered while scanning each tweet on the dataset. Due to simplicity of the approach, an overwhelming amount of different words, some even without any clear meaning, took relevance on the VSM. This created a VSM with a huge size and the trainings at hand took an eternity to process. In order to prevent this from happening, we took a sample of 50MB of tweets, all in English, from the dataset and started to train the SOM with it. String manipulation for VSM reduction described on Subsection III-A were used. The SOM training was performed using the R kohonen package [22]. Three kinds of clusters were found: clusters where no topic could be made sense of, clusters with one or more topics and clusters with a ton of tweets which had the same text .

B. Reducing SOM vector size

String reducer methods enable great VSM reduction. On Figure 7, we can see the amount of words removed by each method alone, and by all methods combined — column ”All Methods”. In order to build this graph, we applied each method independently to a sample of 902802 tweets.

It is interesting to see that each method by itself doesn’t remove a great amount of words. The method that removed more words by itself was ”remove non letters” — which removes every character that is not a letter —, at an order of 33%. On the other hand, the method ”remove stop words” by itself removed only 400 of words. This was expected due to the fact that the full list of MySQL stop words used by this method only has 543 words. All methods combined were able to reduce the VSM size in about 75%.

String reduction techniques work directly with text and has no notion whatsoever of linguistic semantics.

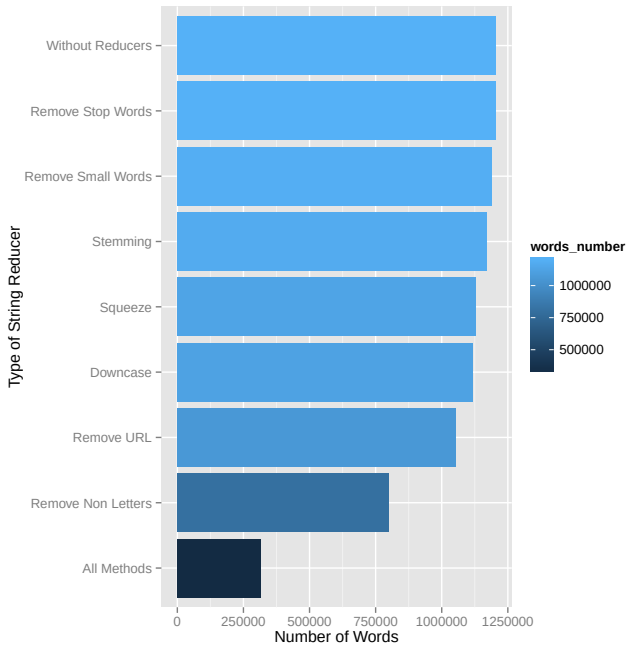

Fig. 7. Amount of unique words present on dataset sample with 902802 tweets, based on the string word reduction technique applied.

By feeding the same dataset as used above to the library, we were able to identify multiple types of words that can afterwards be considered relevant for TDT. On Figure 8, the red bars show the amount of unique words found under a specific semantic tag, whilst in blue we can see words tagged under the same category after applying string reduction techniques.

Due to the fact that we are trying to identify topics, most of the tagged words are of no use. We chose to use only common nouns, proper nouns and hashtags during the clustering process. By applying all these filters to the dataset sample, we have a VSM reduction of about 90%, from 1 204 743 different words to 132 861.

1) Identify Tweets language: Twitter is a social network with users from every corner of the world, thus tweets tend to be in a lot of different languages. Twitter internationalization greatly affects the clustering process, with multi language synonyms and idiomatic expressions. Another problem is associated with results interpretation, when a cluster is formed with tweets with languages foreign to the people who are analyzing the results. When this happens, identifying topics on the cluster is extremely hard, and most of the times the researcher will have to manually translate a cluster.

One way to infer a tweet language, is by looking at the language which a user has on his twitter profile, through the twitter API. Some times, users use their profiles on different languages than the language in which they issue their tweets, which makes the process of excluding foreign tweets harder.

An alternative approach to language identification, is by using an external library like whatlanguage [2], which tries to identify one language through the usage of Bloom Filters.

[2]https://github.com/peterc/whatlanguage

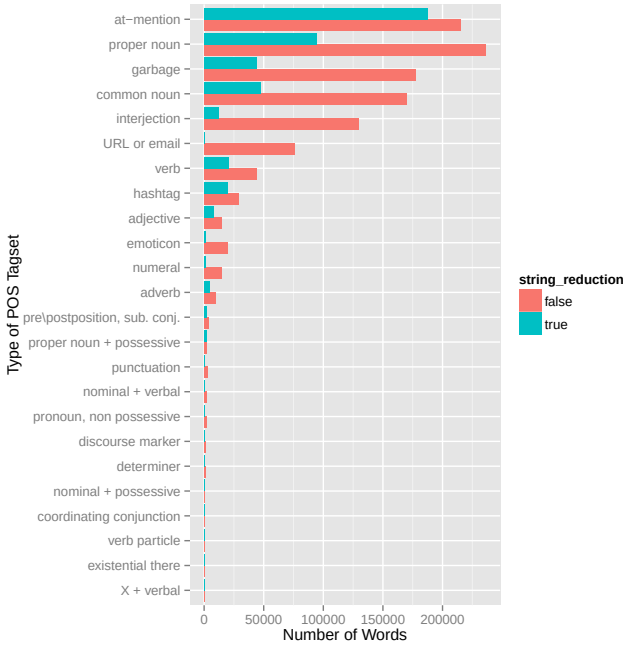

Fig. 8. Number of words tagged with Ark Tweet NLP. In red we can see the number of unique words tagged in each category, while in blue we can see the amount of unique words, after applying string reduction techniques.

In order to better understand the relationship between a tweet language and a user profile language, we analyzed a sample of 87 tweets, all from different users and categorized their language by hand. We found that 7 users were using an English profile, and tweeting in other language.

On the same sample, we used the language identification library and found that even though only 8 tweets were identified as being in English, it did not misidentified any foreign tweet as being in English.

Even though the sample is quite small, it was possible to understand that selecting only tweets with English profiles and afterwards selecting only the ones identified in English by the language identification library, could yield a bigger number of false negatives — tweets in English which were thought to be in other language — , but would help prevent selecting tweets that were not in English.

V. SOM Framework

The SOM framework was developed in the Ruby programing language [3] due to the desired characteristic of allowing great levels of introspection and being an almost pure object oriented programing language. Due to this characteristics, making modifications to core parts of the algorithm is fairly easy.

The SOM Framework was developed in a test driven fashion, having 100% of its public methods tested and documented for expected behavior. These characteristics, associated with the fact that was published under an open source license, makes it available for other researchers to implement their own SOM variants.

[3]https://www.ruby-lang.org/en/

By default, the base SOM algorithm is implemented as described by the Algorithm 1.

A. Clustering Color Vectors

Out of the box, the SOM Framework implements a squared output space, where all residing neurons are manipulated as arrays. It is possible at any given moment of the training to export the output space to JavaScript Object Notation (JSON), Comma Separated Values (CSV) or to visualize its current U-Matrix. Also, during training, a progress bar is displayed in order to know how much time will be needed for the training to end.

Due to the features described above, it is possible to train a SOM to identify random colors — RGB vectors — while printing the results. In order to do this, we will start by:

- Initializing a SOM object with an output space size of 15 by 15 neurons, which will yield a total of 255 neurons — and directly maps to the maximum number of clusters — and 700 epochs.
- Create 1500 input patterns with size 3 and random values between 0 and 255.
- Tell the SOM to print its state at the end of each epoch.

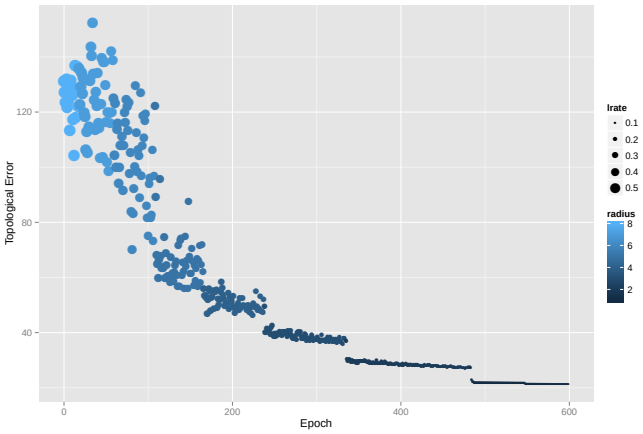

Fig. 9. Changes in topological error throughout the SOM training, lrate stands for learning rate, and radius for radius applied to the winning neuron

On Figure 9 we can see the evolution of the topological error and how it is converging throwout the training process, as the radius and learning rate are decreasing, and as well as the number of epochs is rising.

On Figure 10 we can see the average distance between neurons increasing. At first this might not look like a desired property, but in fact it is. When the distance between the neurons is increasing and the topological error is decreasing, it means that the neurons are scattering in the output space in order to better identify the input patterns they are responsible for.

During the training of this SOM, we analyzed the output space, U-Matrix and Q-Matrix during the begging, half of the train, and finally at the end of the training. The U-Matrix evolves in a way where clusters are almost unnoticeable, which is caused by the homogeneity and randomness of the input patterns. The Q-Matrix evolves, by becoming whiter

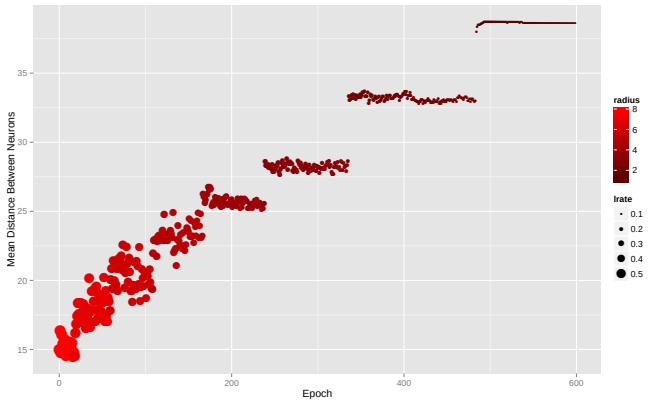

Fig. 10. Changes in the average distance between neurons, throughout the SOM training

which represents that the mean topographic error is becoming smaller. This was already seen before in Figure 9, but now we can also see which neurons are worst at representing the input patterns.

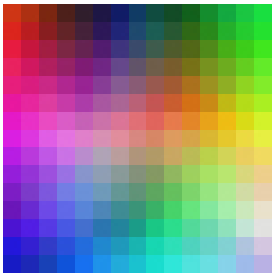

Fig. 11. Output Space, at the end of training.

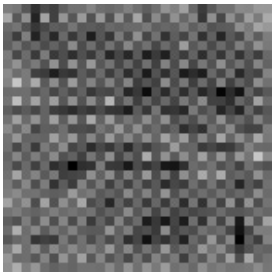

Fig. 12. U-Matrix, at the end of training.

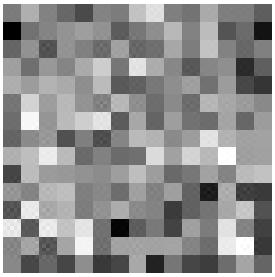

Fig. 13. Q-Matrix, at the end of training.

In order to see how well the neurons are representing the input patterns, we looked at the Q-Matrix and selected the darkest area in order to know which neuron is the worst at representing its input patterns. Afterwards, we printed the input patterns associated to that neuron. This process was graphically represented in Figure 14 where the colors which represent the input patterns are in fact RGB vector coordinates used during training. It is possible to see that even though this neuron ought to be the worst at representing it input data, he represents it quite well as they are all shades of red. It is important to know that all of this visualization can only be made due to the fact that, we are working with arrays with three dimension and values comprised between 0 and 255 — which makes it possible for them to be presented as RGB images.

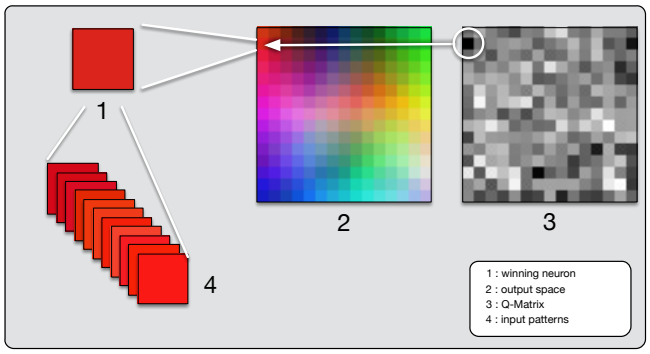

Fig. 14. Input patterns associated with the neuron with maximum topological error –31. Even though the neuron has the biggest topological error of all neurons, it still has a good representation of the input patterns. The colors in this image are not figurative, and represent the entities at the end of training.

B. Benchmarking

The SOM framework was not created with the purpose of being extremely fast, for that there are already very good implementations like Wittek [25] distributed library for SOM or Wehrens and Buydens [22] R kohonen package which implements the training algorithm in C, and only exposes the interface in the high level language R. Being purely written in a higher level language, the SOM framework enables researchers and programmers to write training algorithms very fast.

The framework was tested against multiple sizes of output space and input patterns, multiple numbers of input patterns, and multiple numbers of epochs. The results were summarized on Figure 15, where it is possible to see on the upper quadrant that if all parameters increase, SOM training will suffer as well.

C. Homophilic SOM

In order bring the concept of homophily — which have proved to be present on social networks [23] — to clustering socially connected data, on Section III-C we suggested some alterations to the default SOM algorithm. These modifications were mainly applied to the output space, where each neuron started to represent a user, and his neighborhood is comprised of his social relations.

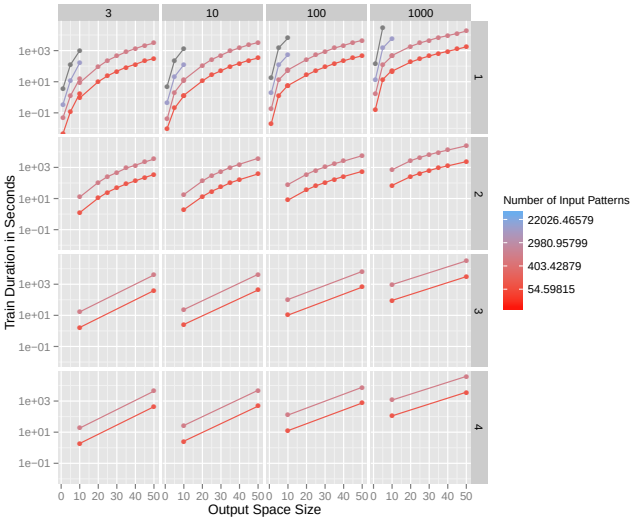

Fig. 15. SOM framework train duration, influenced by output space size in the XX axis, duration of the training on the YY axis, number of epochs in the right, size of input patterns on top, and number of input patterns on the right in color from red to blue.

These features were implemented into the SOM framework described in the previous chapter.

It is not possible to draw a U-Matrix due to the fact that the output space is no longer a rectangular matrix, but a graph. Also drawing, the Q-Matrix is possible, but the disposition of the neurons will not represent the actual disposition in the graph. This can still be useful to easily visualize which neurons are better at representing their input patterns.

Looking at the results, there are a lot of different topics of clusters. There were neurons clearly responsible to identify music, clusters about tech and programing which surprisingly could identify the tweet about the banana phone which was tech project presented at codebits. These clusters can be seen in Figure V-C and V-C. On the other hand, some clusters of people saying that they have posted photos on facebook, or that they've liked youtube videos were also found. Even though they can be considered topics, their relevance is not very high.

1. I think I haven't had a segmentation fault in years http://t.co/COjaaFj6lb
2. Just bought a banana phone at #bananamarket
3. Real Software Engineering by @glv http://t.co/kXeDmZSGi7 via @confreaks. @daviddias you're going to enjoy this (it is not about ruby)
4. R vrs SAS, interesting debate:
5. http://t.co/tx4xFED8zR
6. if that's what needs to be done, #atomselfie . Send it to bersimoes@gmail http://t.co/CvlXq8DmWd
7. I'm finding @duckduckgo to be pretty more reliable than google when searching for code. Gonna try it as my default se.
8. Centro de Ayuda de Twitter | Which Twitter Account is My Mobile Phone Associated With? https://t.co/Vz9HonbQ via @Ayuda
9. Vijf keer anders staken http://t.co/mjTB8vjNiX via @destandaard - hoe woede leidt tot creativiteit
10. "Learn from the data we have : In the US, 250,000 women track their pregnancy using a mobil phone app. @PregnancyTaboos #tedxbrussels"
11. 136 usuarios que sigo no me siguen de vuelta en Twitter. Entérate de quién no te sigue de vuelta http://t.co/u9m5D9zr9k
12. Revolutionary Foldable Smartphone Shows Shape-Shifting Future for Google Maps http://t.co/HRpBwK3ISB
13. Amazing smart phone foldable computer concept. http://t.co/O8h9iINqRQ
14. Superb mobile 3D scanning project | #3Dprinting #scanner #3Dmodel #photography #technologyintegration https://t.co/3fWjkKQnUx
15. Mind blowing results! Taking Commercial 3DP into the Nano Dimension - #3DPrinting | @scoopit http://t.co/zWE8Q6yipE
16. #iHive3D Rewards #3DPrinting Pros for Sharing Wisdom with Newbies #videocontest See more: http://t.co/KhgCsTgipA
17. #3Dprinting finally adopted by the masses. $299 3D Printer Hits Kickstarter Goal In 11 Min | TechCrunch | @scoopit http://t.co/3WMoxcug3D
18. The First $299 3D Printer Hits Its Kickstarter Goal In 11 Minutes | TechCrunch - See on Scoop.it - 3D... http://t.co/VGG2BzNgVO
19. #Education #opensource #recycling #fairtrade #3Dprinting Brilliant project @SavantUSA 3DforEDucation @scoopit http://t.co/Up51X8GgSc

Fig. 16. Cluster about tech and programing

1. I think I haven't had a segmentation fault in years http://t.co/COjaaFj6lb
2. Just bought a banana phone at #bananamarket
3. Real Software Engineering by @glv http://t.co/kXeDmZSGi7 via @confreaks. @daviddias you're going to enjoy this (it is not about ruby)
4. R vrs SAS, interesting debate:
5. http://t.co/tx4xFED8zR
6. if that's what needs to be done, #atomselfie . Send it to bersimoes@gmail http://t.co/CvlXq8DmWd
7. I'm finding @duckduckgo to be pretty more reliable than google when searching for code. Gonna try it as my default se.
8. Centro de Ayuda de Twitter | Which Twitter Account is My Mobile Phone Associated With? https://t.co/Vz9HonbQ via @Ayuda
9. Vijf keer anders staken http://t.co/mjTB8vjNiX via @destandaard - hoe woede leidt tot creativiteit
10. "Learn from the data we have : In the US, 250,000 women track their pregnancy using a mobil phone app. @PregnancyTaboos #tedxbrussels"
11. 136 usuarios que sigo no me siguen de vuelta en Twitter. Entérate de quién no te sigue de vuelta http://t.co/u9m5D9zr9k
12. Revolutionary Foldable Smartphone Shows Shape-Shifting Future for Google Maps http://t.co/HRpBwK3ISB
13. Amazing smart phone foldable computer concept. http://t.co/O8h9iINqRQ
14. Superb mobile 3D scanning project | #3Dprinting #scanner #3Dmodel #photography #technologyintegration https://t.co/3fWjkKQnUx
15. Mind blowing results! Taking Commercial 3DP into the Nano Dimension - #3DPrinting | @scoopit http://t.co/zWE8Q6yipE
16. #iHive3D Rewards #3DPrinting Pros for Sharing Wisdom with Newbies #videocontest See more: http://t.co/KhgCsTgipA
17. #3Dprinting finally adopted by the masses. $299 3D Printer Hits Kickstarter Goal In 11 Min | TechCrunch | @scoopit http://t.co/3WMoxcug3D
18. The First $299 3D Printer Hits Its Kickstarter Goal In 11 Minutes | TechCrunch - See on Scoop.it - 3D... http://t.co/VGG2BzNgVO
19. #Education #opensource #recycling #fairtrade #3Dprinting Brilliant project @SavantUSA 3DforEDucation @scoopit http://t.co/Up51X8GgSc

Fig. 17. Cluster about music

VI. Conclusions and Future Work

This work presents an innovative approach to topic detection on social networks. The clustering mechanism takes into consideration the concept of homophily, which have been proved to occur in social networks [23]. In order to achieve this, we presented a new way to reduce the VSM up to 90% with minimum relevant data loss for topic detection on twitter. Built a SOM library in ruby, which we edited afterwards to add social connections to the neurons during the training process.

Proposed a new visualization technique for SOM called Q-Matrix, where it is possible to see how well a neuron represents its associated input patterns.

As future work, we would like to improve performance of Homophilic SOM in order for it to be able to crunch a lot more data. It would also be awesome to have some pre categorized tweets, so we could compare results faster without having to be reading the content of the clusters.

Also designing U-Matrix and Q-Matrix for the homophilic SOM should be possible. This should be accomplished, by printing the output space as graph and afterwards, apply the same concepts used for the squared output space. One way to do this, would be by processing the SOM results into HTML documents, and use libraries like D3js [4] to build all the graphical parts.

References

[1] Allan, J. (2002). *Topic detection and tracking: event-based information organization*, volume 12. Springer.

[2] Asur, S. and Huberman, B. (2010). Predicting the future with social media. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 492–499.

[3] Bação, F., Lobo, V., and Painho, M. (2005). The self-organizing map, the Geo-SOM, and relevant variants for geosciences. *Computers & Geosciences*, 31(2):155–163.

[4] Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[4] http://d3js.org/

[5] Blei, D., Ng, A., and Jordan, M. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

[6] Cataldi, M., Di Caro, L., and Schifanella, C. (2010). Emerging topic detection on twitter based on temporal and social terms evaluation. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, MDMKDD '10, pages 4:1–4:10, New York, NY, USA. ACM.

[7] Havens, T., Keller, J., and Popescu, M. (2010). Computing with words with the ontological self-organizing map. *Fuzzy Systems, IEEE Transactions on*, 18(3):473–485.

[8] Honkela, T., Kaski, S., Lagus, K., and Kohonen, T. (1997). Websom - self-organizing maps of document collections. In *Neurocomputing*, pages 101–117.

[9] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components.

[10] Knoke, D. and Yang, S. (2008). *Social network analysis*, volume 154. Sage.

[11] Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*.

[12] Kruskal, J. and Wish, M. (1978). *Multidimensional Scaling*. Sage Publications.

[13] Le, Q. V., Ranzato, M., Monga, R., Devin, M., Corrado, G., Chen, K., Dean, J., and Ng, A. Y. (2012). Building high-level features using large scale unsupervised learning. In *ICML*. icml.cc / Omnipress.

[14] Liu, Y., Liu, M., and Wang, X. (2012). Application of Self-Organizing Maps in Text Clustering: A Review. *Applications of Self-Organizing Maps*, pages 205–219.

[15] Melville, P., Sindhwani, V., and Lawrence, R. (2009). Social media analytics: Channeling the power of the blogosphere for marketing insight. *Proc. of the WIN*, pages 2–6.

[16] Owoputi, O., Dyer, C., Gimpel, K., Schneider, N., and Smith, N. A. (2013). Improved part-of-speech tagging for online conversational text with word clusters. In *In Proceedings of NAACL*.

[17] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

[18] Rauber, A., Merkl, D., and Dittenbach, M. (2002). The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. *Neural Networks, IEEE Transactions on*, 13(6):1331–1341.

[19] Salton, G. and McGill, M. J. (1983). Introduction to modern information retrieval.

[20] Shah-Hosseini, H. and Safabakhsh, R. (2003). Tasom: a new time adaptive self-organizing map. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 33(2):271–282.

[21] Tobler, W. R. (1970). A computer movie simulating urban growth in the Detroit Region. *Economic Geography*, 46:234–240.

[22] Wehrens, R. and Buydens, L. (2007a). Self- and super-organising maps in r: the kohonen package. *J. Stat. Softw.*, 21(5).

[23] Wehrens, R. and Buydens, L. (2007b). Self-and super-organizing maps in R: the Kohonen package. *Journal of Statistical ...*, 21(5).

[24] Weng, J., Lim, E.-P., Jiang, J., and He, Q. (2010). Twitterrank: Finding topic-sensitive influential twitterers. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 261–270, New York, NY, USA. ACM.

[25] Wittek, P. (2013). Somoclu: An efficient distributed library for self-organizing maps. *CoRR*, abs/1305.1422.

[26] Witten, I. H. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.