TÉCNICO
LISBOA

HOMOPHILIC SELF ORGANIZING FEATURE MAPS: FINDING TOPICS ON SOCIALY CONNECTED DATA, USING SOCIAL NETWORK RELATIONS

Bernardo Simões

Dissertação para obtenção do Grau de Mestre em

Engenharia de Redes de Comunicações

Júri

Presidente:	Prof. Paulo Jorge Pires Ferreira
Orientador:	Prof. Pável Pereira Calado
Vogal:	Prof. Alexandre P. Francisco

Outubro de 2014

Acknowledgments

First of all, I need to thank Professor Pável for introducing me to the world of Machine Learning, without his teachings during the course of web analysis and information extraction, and full support during the development of this thesis. Sofia Martins, for all the support when things got rugher and for proofreading this thesis more times than its humanly bearable. To my parents and family, whom dedicated the last 25 years into making me an educated person, and for always worring with my success. I need to thank all my friends and coleges from Instituto Superior Técnico, in special to Afonso Oliveira whom was more than a friend, but an actual mentor. Guilherme Vale, Mario Nzualo, Fábio Domingos, João Vasques, João Andrade, David Dias, Rui Costa, Artur Balanuta and probably many more, for all the working hours we have spent together, I couldn't have finished my course without you. Finally, to everybody that makes Taguspark Campus probably on the best places to work in the world. It was an awesome journey.

Abstract

With the evolution of social media platforms, the amount of unlabeled information has gone skyrocketing.The process of labeling this kind of information is 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. In this thesis, 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.

Keywords

topic detection, twitter, self-organizing maps, classification, clustering

Resumo

Palavras Chave

detecção de tópicos, twitter, mapas auto organizados, classificação, agrupamento

Contents

List of Figures

List of Tables

List of Acronyms

YAML Yet Another Markup Language

SOM Self-Organizing Maps

DC Document Clustering

NLP Natural Language Processing

U-Matrix Unified Distance Matrix

TF-IDF Term Frequency–Inverse Document Frequency

LDA Latent Dirichlet Allocation

TDT Topic Detection and Tracking

VSM Vector Space Model

IR Information Retrieval

ML Machine Learning

ANN Artificial Neural Network

MDS Multi Dimensional Scalling

PCA Principle Component Analysis

URL Uniform Resource Locator

JSON JavaScript Object Notation

CSV Comma Separated Values

1

Introduction

Contents

With the evolution of social network websites like Facebook and Twitter, the amount of pertinent content about a specif issue is increasing dramatically, which calls for new ways to make sense and catalog this data. The usage of social networks for branding quality and on-line marketing is specially compelling since 19% of all tweets [14] and 32% of blog posts [26] are about brands or products. Nevertheless, 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. As a consequence, the ML course at Standford is the one with more students enrolling in the year of 2014 [1] with more than 760 students enrolled.

Using unsupervised ML, Le et al. [20] 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. He used a 9-layered locally connected sparse auto-encoder with pooling and local contrast normalization on a large dataset of images (the model has 1 billion connections, the dataset has 10 million 200x200 pixel images downloaded from the Internet). This dataset was trained using model parallelism and asynchronous SGD on a cluster with 1,000 machines (16,000 cores) during three days. Even though the amount of computing power used in this project was of several order of magnitude, it is remarkable how an unsupervised algorithm could achieve such results.

Even though a lot of solutions have arisen in order to automate real time searches, topic categorization and many other data intensive tasks are still done manually. Twitter still uses humans to deliver ads to trending queries, states Edwin Chen's Data Scientist responsible for ads quality at Twitter. On his blog post [2], Edwin Chen describes the process of delivering real time adds to trending queries at Twitter. The main problems that arise in the Twitter platform in order to identify rising topics are:

- The queries people perform have never before been seen, so it is impossible to know beforehand what they mean.
- Since the spikes in search queries are short-lived, there's only a short window of opportunity to learn what they mean.

This means that when an event happens, people immediately come to Twitter in order to know what is happening in a determined place. Twitter solves this issue by monitoring which queries

[1] http://www.forbes.com/sites/anthonykosner/2013/12/29/why-is-machine-learning-cs-229-the-most-popular-course-at-stanford/

[2] http://blog.echen.me/2013/01/08/improving-twitter-search-with-real-time-human-computation/

are currently popular in real time, using a Storm topology [3]. After the queries are identified, they are sent to a Thrift API [4] that dispatches the query to Amazon's Mechanical Turk service [5] where real people will be asked a variety of questions about the query.

Social Media Analytics is another raising topic that draws from Social Network Analysis [16], ML, Data Mining [38], Information Retrieval (IR) [30], and Natural Language Processing (NLP). As stated Melville et al. [26], 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 Sumal[6] or ThoughtBuzz[7], but also solutions from the old players like IBM[8] and SAS[9]

It's also important to notice that in the last few years Data Science/Analysis has been a trending topic, mostly due to the fact that big dot-com companies have been having high revenues by exploiting user specific information in order to deliver ads and sell products. Not surprisingly that if you look that in the top ten ebooks sold by O'Reilly throughout 2013, four are about data science [10].

1.1 Motivation

Clustering analysis has been widely used throughout the times, from its first occurrence in England, where John Snow was able to map a wider amount of people infected with cholera to a well in the center of London. Nowadays the applications are endless, and fields where it is applied are quite vast.

Specifically with a greater amount of people describing events around them, and their lives on social media, it is increasingly more challenging to categorize this data, due to its sparsity and volume.

However, data generated on social networks, have more information than simply written text, on a tweet, or a photo published on Facebook. Data generated in social network is connected by entities, and these entities tend to be closer to other entities with the same kind of interests [25]. In this thesis we will explore how a clustering algorithm can be altered in order to add social relevance to its fed data. By focusing on using an unsupervised learning technique based on

[3] http://storm-project.net/
[4] http://thrift.apache.org/
[5] https://www.mturk.com/mturk/
[6] https://sumall.com/
[7] http://www.thoughtbuzz.net/
[8] http://www-01.ibm.com/software/analytics/solutions/customer-analytics/social-media-analytics/
[9] http://www.sas.com/software/customer-intelligence/social-media-analytics.html
[10] http://shop.oreilly.com/category/deals/bestoforeillydotd.do?code=DEAL&cmp=twnabooksvideosinfo-authornote_best_of_2013

neural networks named SOM [18] in order to detect topics in Twitter posts, by using the Social Network users as base neurons for clustering. After the network is trained, it will be possible to categorize tweets in real time.

1.2 Objectives

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 SOM, that will later classify each future tweet that arrives on the network without further delay.

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 fact that there is a relationship between authors of tweets.

1.3 Contributions

The main contributions of this work are as follow:

- **Homophilic SOM:** We proposed and analyzed the application of integrating social relations into the SOM clustering algorithm.
- **SOM framework:** A framework to easily extend the SOM algorithm.
- **Mini Twitter:** An highly customizable twitter crawler.

1.4 Dissertation outline

This document is organized as follows:

- **Chapter 2** presents the background information about SOM and Document Clustering (DC), being the two most important concepts needed to understand the following chapters.
- **Chapter 3** describes the latest work done with Topic Detection and Tracking (TDT) on Twitter, and the latest work done with SOM.
- **Chapter 4** describes the theoretical details which need to be taken into account, in order to efficiently modify the SOM algorithm.
- **Chapter 5** analyzes and benchmarks the work described on Chapter 4.
- **Chapter 6** summarizes the work developed and analysis of future work.

2

Background

Contents

In this section, we will start by generally describing what clustering is and how it works. We will then outline how SOM [18] perform, which is the document clustering algorithm used on this thesis.

2.1 Document Clustering

Document clustering is an optimal division of documents into categories without prior knowledge of the data that is being organized, based only on the similarity between them. Due to the fact that no prior knowledge of the data has to be known, document clustering is labeled as Unsupervised ML [11].

Liu et al. [22] asserted that document clustering can be used in a variety of Computer Science fields, such as:

- Natural Language Preprocessing.
- Automatic Summarization.
- User preference mining.
- Improving text classification results.

There are two main types of document clustering: hard clustering and soft clustering. In hard clustering, one document can only belong to one cluster, while in soft clustering one document can belong to multiple clusters.

The clustering process usually works as described in Figure 2.1. In the first step, a data set must be provided with the documents to be clustered. The second step is where non relevant words are removed from the documents, to improve clustering quality [15].

The third step is characterized by converting the keywords of each document into vectors. The most common model used for this task is VSM. In VSM, each vector dimension represents one detected keyword and each document is represented by the vector of keywords in the feature space. This process is illustrated in Figure 2.2 and works in the following way:

- **First step:** string tokenization, and token selection. In this case, stop words and repeated words will be removed.
- **Second step:** string to VSM conversion. Each different word will correspond to a position in the array, and its value will correspond to the number of occurrences.

There are many clustering algorithms. In the following section we will describe the particullar case of the SOM algorithm, the solution used in our work.

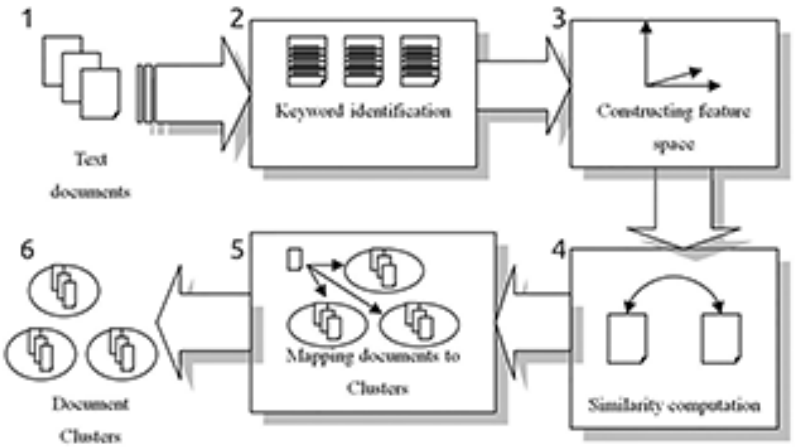


Figure 2.1: Text clustering main framework [9]

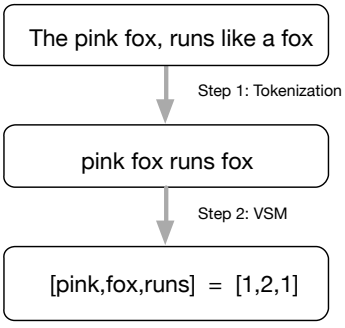


Figure 2.2: Text tokenization and transformation to Vector Space Model.

2.2 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) [19] and Principle Component Analysis (PCA) [13] .

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 2.4.

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].

SOM training is always subject to some variability due to multiple causes, like the sensitivity of initial conditions, convergence to local minima and sampling variability [6].

No general formula exists to minimize quantization error [6] . In order to achieve a minimal value, the number of neurons, value of neurons and order of the input data is randomly changed. Multiple SOMs are trained and the one with the lowest mean quantization error is chosen.

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. 2.1). 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 $D_{i,j}$.

$$\varepsilon(w) = \frac{\sum_{i=0}^{n} \|w - d_i\|}{n}, d_i \in D, \forall n \tag{2.1}$$

Algorithm 1: Self-Organizing Map [18]

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. 2.2) is generally used.

$$Dist = \sqrt{\sum_{i=0}^{i=n}(V_i - W_i)^2} \tag{2.2}$$

- **On line 2:** the winning neuron (p) previously selected on line 1 is updated, in order to better represent the input pattern — this process is represented on Figure 2.3. Also, all other neurons inside a specific radius will also be updated — this process is described in Figure 2.4. 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. 2.3) is often used.

$$h_{ip}(t) = \exp -\frac{|\overrightarrow{a_i} - \overrightarrow{a_p}|^2}{\sigma^2(t)} \tag{2.3}$$

- **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 .

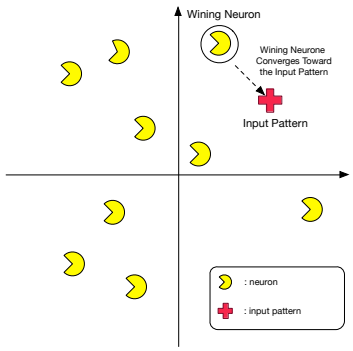


Figure 2.3: Winning neuron converging at learning rate

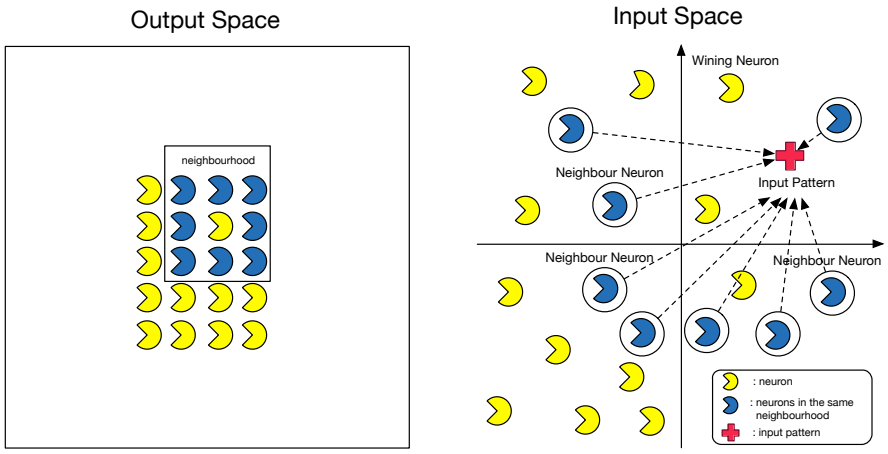


Figure 2.4: On the left the output space neighborhood, on the right the neighbors of the winning neuron converging in the direction of the input pattern

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. [22], 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. One way to compute the U-Matrix is described in Algorithm 2.

The U-Matrix algorithm (Alg. 2) is composed by four main cycles, which work in the following way:

- **Cycle 1:** On line 1 neurons are added to the empty matrix, in order to be able to calculate the distance between them.
- **Cycle 2:** From line 2 to 4, we are calculating the distance between adjacent neurons, and adding the distance to the empty position between them. On line 2 and line 3 we are looking for neurons horizontally and vertically, respectively. On line 4 we are looking for neurons on the diagonal. The diagonal calculus is more complex due to the fact that we need to calculate the lower diagonal to the current neuron, and average it with the upper diagonal of the neuron immediately billow it.
- **Cycle 3:** At this, stage the matrix is full of the distances between neurons, theoretically nothing else would be needed to be calculated. But it still has neurons residing on the matrix which need to be removed. On line 5 we substitute all neurons with an average of the distances surrounding them.
- **Cycle 4:** Finally, all the values on the matrix are substituted by colors on a black to white scale. Bigger distances between neurons are represented with darker colors while smaller distances are represented with lighter colors. This conversion is done on line 6.

An example of an U-Matrix can be seen in Figure 2.5(b).

Algorithm 2: U-Matrix

Data: $W = \{\overrightarrow{w_{0,0}}, \ldots, \overrightarrow{w_{n,n}}\}$ are the trained neurons

$D_{i,j}$ be the input patterns represented with neuron $w_{i,j}$

U is an empty matrix with size $2n - 1.2n - 1$

Result: U-Matrix

`/* Initialize` U `by adding the trained neurons */`

for $w_{ij} = \overrightarrow{w_{00}}$ to $\overrightarrow{w_{n,n}}$ **do**

1 $U_{i*2,j*2} \leftarrow w_{i,j}$

`/* Calculate the distance between every adjacent neurons, and apply it to the square between them */`

for $i = 0$ to U_{max} **do**

  for $j = 0$ to U_{max} **do**

    if $l + 1 < m || j + 1 < m$ **then**

2 $U_{i+1,j} = \|u_{i,j} - u_{i+2,j}\|$

3 $U_{i,j+1} = \|u_{i,j} - u_{i,j+2}\|$

4 $U_{i+1,j+1} = \frac{\|u_{i,j} - u_{i+2,j+2}\| + \|u_{i+2,j} - u_{i,j+2}\|}{2}$

    $j \leftarrow j + 1$

  $i \leftarrow i + 1$

`/* Substitute the neurons for an average of surrounding distances */`

for $i = 0$ to U_{max} **do**

  for $j = 0$ to U_{max} **do**

5 $u_{ij} = avg(Adj[u_{ij}])$

    $j \leftarrow j + 1$

  $i \leftarrow i + 1$

`/* convert the distances to color */`

$WHITE = 255$

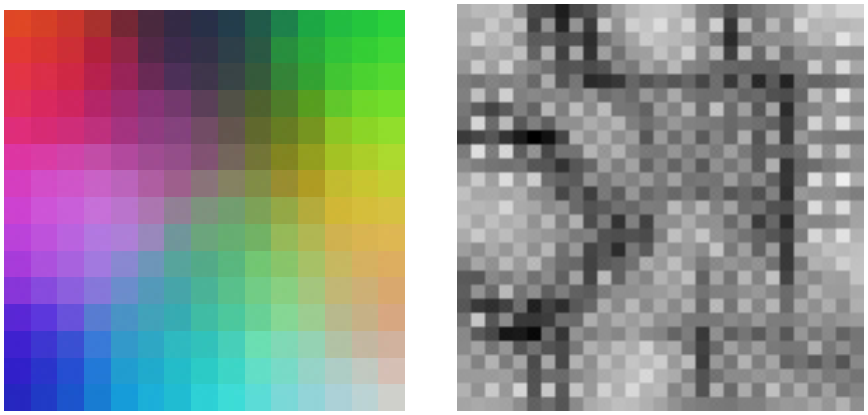
$BLACK = 0$

$u_{max} \leftarrow max(U)$

$u_{min} \leftarrow min(U)$

for $u_{ij} = u_{00}$ to $u_{n,n}$ **do**

6 $U_{i,j} \leftarrow (1 - \frac{u_{i,j} - u_{min}}{u_{max} - u_{min}}) * WHITE$



(a) SOM ouputspace trained with random RGB vectors

(b) U-Matrix representing how well each neuron maps to input patterns it represents

Figure 2.5: U-Matrix and SOM output space, computed after 300 epochs of training, with 1500 random input patterns representing an RGB color.

3

State of the art

Contents

This section provides insight of work done in several research areas that are related to the project. In section 3.1, work done using SOM maps will be described. Section **??** and 3.2 are dedicated to data clustering and to data mining specifically on Twitter [1].

3.1 Self-Organizing Maps

SOM are used in a wide area of applications, from authentication systems [9] through network intrusion detection [31] and speech recognition and analysis [17]. In this section we highlight some of their applications.

3.1.1 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." [34] 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 3.1, where the units considered for the best match are defined by a sort 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.

3.1.2 Detecting Hidden Patterns on Twitter Usage

Cheon and Lee [8] analyzed hidden patterns in the usage of twitter. In their study, they started by collecting data from the twitter API of different kinds of topics like "2009 Iran Election" and "iPhone 3.0 OS launch". They made multi level signal extraction, not only from information directly

[1] http://www.twitter.com

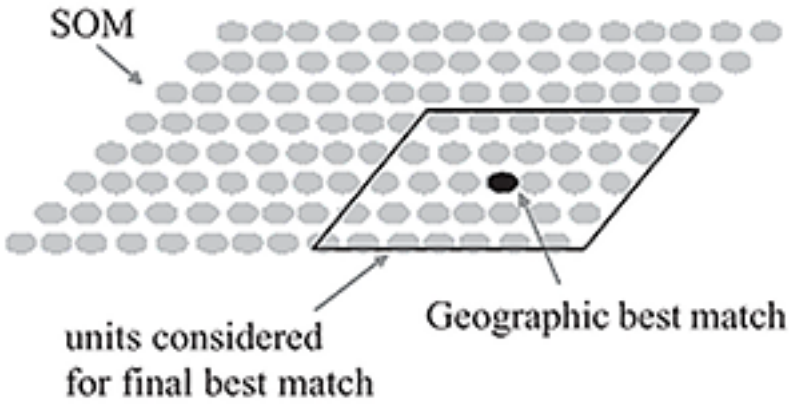


Figure 3.1: Geo-SOM structure, where the units considered to be the winning neuron are constrained by the geographic coordinates of the data, from Bação et al. [3]

present on the tweet, but also by cross referencing with other social websites and with the twitter user profile information. The signals retrieved from the social network can be seen in Table 3.1.

Table 3.1: Twitter Signals

Twitt Corpus
Tweet Size
Replies
Re-tweets
Hashtags
Presence of URIs and Type of linked content
Type of Device
Tweet Location
Twitter Profile
Account Age
Gender
Country
frequency of posts
Friends to followers ratio
Number of customizations
External Sources
Other Social Network Accounts
Type of website

By applying a SOM, they could find four demographical clusters during the Iran 2009 Election. The first cluster was characterized by young web-based Iranians, with twitter accounts not older than three months with a high frequency of replies. The second cluster was mainly compound of web users from Iran accounts older that three months. The third cluster had Iranian users with mobile clients with large texts clearly trying to raise awareness. The fourth and final cluster

represented the users around the world trying to raise awareness about the issue by sharing tweets with URIs. Looking at their analysis about the topic ”2009 Iranian Election”, it is clear to see that it was possible to describe the type of users represented in the social network and the way they interact with it.

On the iPhone 3.0 OS launch, it was possible to find three main clusters. The first cluster was characterized by male users, accounts older than 90 days, coming from countries where the iPhone is marketed, with high adoption of social media clearly representing the target market of the iPhone or its customers. The second cluster had new accounts with higher rate of followers to followees, high frequency of posts per day, presence of URI linking to technology blogs or websites, no country or gender specified meaning that this cluster was clearly composed by news aggregators and technological news websites. Inside the second cluster, there was a sub-cluster of Japanese users which represents the high rate of iPhone adoption in Japan. Finally, the third cluster was clearly spammer accounts that where eventually deleted after a couple of months, characterized by popular social connections, posting more than fifty tweets a day with external URIs and the accounts were not older than a day or so.

In conclusion, it was possible to detect Twitter usage patterns, and specifically, detect spammers before they were banned from the social network.

3.1.3 WEBSOM

Honkela et al. [12] 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.

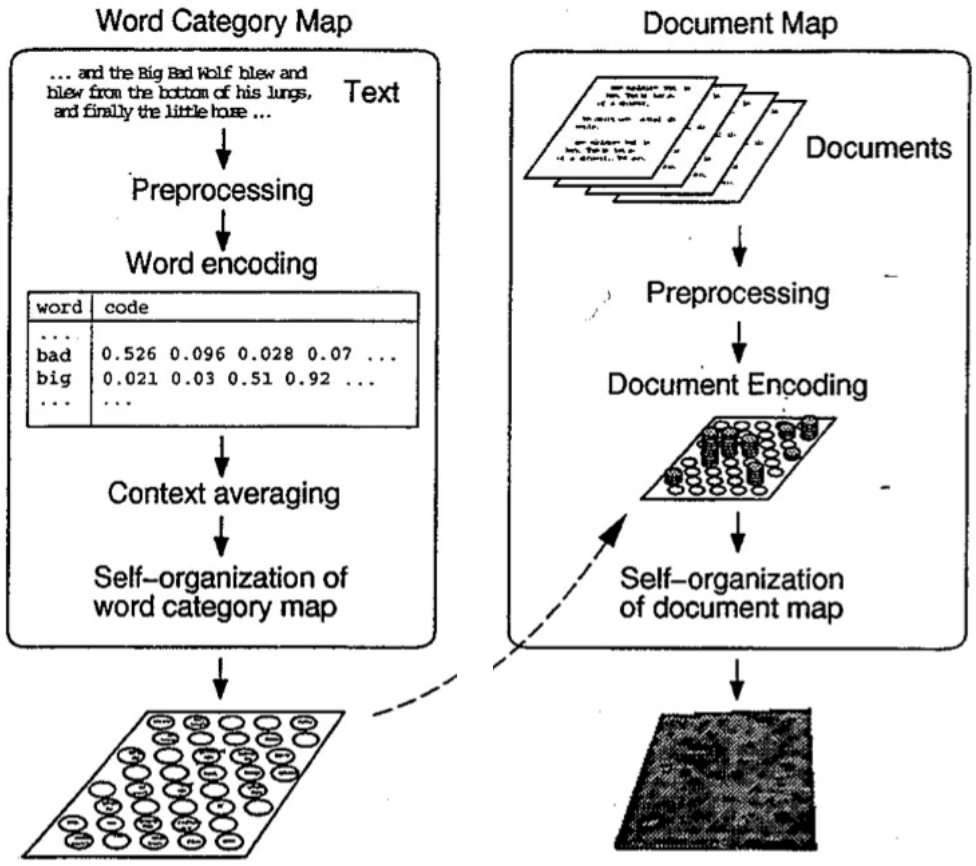
The WEBSOM was not based on keywords and boolean expressions, instead, words with the same meaning are encoded in a word category map (Fig 3.2(a)), where placement and frequency in documents is taken into account. This way it is possible to remove words with similar meaning — greatly reducing the VSM size making it possible to train the document map in a scalable way.

3.2 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.

3.2.1 Topic and Trending Detection

Allan [1] defined 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,



(a) The word cathegory map, a SOM where word frequency and placement is used for encoding text

(b) The document map, organized based on documents enconded with the word cathegory map.

Figure 3.2: Basic architecture of the WEBSOM method, from [12]

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 [2], 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. [7] 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 [28] 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. [36] 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.

3.2.2 Tweets Implicit Data

Tweet retrieval and analysis is a double edged problem. On one side, the tweet is really small, which makes it almost impossible to retrieve any actual sense from it. On the other hand, the amount of tweets generated per day is around 140 million[3], which means that it is very hard to do a deep analysis of the semantics and content of individual tweet, and that only the more appropriate signals should be evaluated. For this reason, Tao et al. [33] evaluated how the multiple signals that could be retrieved, directly or indirectly, from the tweet corpus could mean that a tweet

[2] Data source: http://www.pewinternet.org/Presentations/2006/UserGenerated-Content.aspx
[3] https://blog.twitter.com/2011/numbers

Table 3.2: Tao et al. [33] tweet characteristics hypothesis versus influence

Hypotheses	Influence of Features
Syntatical	
Tweets that contain hashtags are more likely to be relevant than tweets that don't	Not Important
Tweets that contain an URI are more relevant that tweets that don't	Important
Tweets that are replies to other tweets are less relevant	Important
The longer the tweet is the more relevant it is	Not Important
Semantic	
The more the number of entities the more relevant a tweet is	Important
Different types of entities are of can have different amount of interest to a give topic	Important
The greater the diversity of concepts mentions in a tweet the more likely for it to be relevant	Important
The relevance of a tweet is determined by its polarity	Important
Contextual	
The lower the temporal distance between a query and the creation of a tweet the more relevant the tweet is	Not Important
The more the number of tweets created by a user the more relevant one of his tweets will be	Not Important

is relevant for a determined topic. In their work, they present premises that seem intuitively true and proves they actually are relevant through a comparison of multiple precision and recall values. Their results on feature comparison are summarized in Table 3.2. The first column consists of all the made hypothesis categorized by type, and the second column tells if the data used actually influenced in precision and recall results. Tao et al. [33] also compared results, of topic characteristics, concluding that distinction between local and global events as well as temporal persistence proved to not be relevant on relevance prediction.

McCreadie and Macdonald [24] also approached the issue of having very little content on tweets in order to categorize them, and tried to solve the problem by applying the content of linked URIs into the tweet body in order to improve precision and recall. The best fitting approach was using Field-Based weighting, where for each tweet a new document is created, which contains two fields: the terms in the tweet and the terms in the linked document. Afterwards a learning to rank algorithm called PL2F [23] is used against the dataset from "Trec Microblog2011" in order to find the best weighting. With this model they were able to improve precision in an order of 0.9, over only analyzing the text contained in the tweets.

3.2.3 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. [27] published a NLP library, specific for twitter. As shown in Figure 3.3, 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. [27] 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.

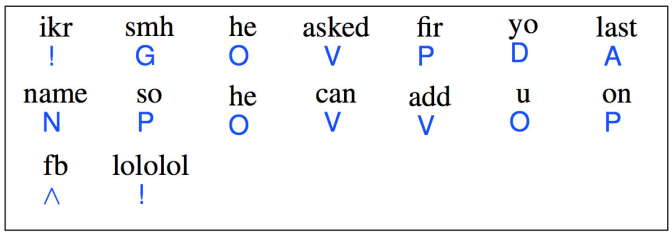


Figure 3.3: Tweet automatically tagged with ARK Tweet NLP. ! stands for interjection, while V stands for verbs and D for determiner. The full table of tags can be found in [27].

3.2.4 Rapidly Changing Trends

Due to the real time nature of Twitter, using typical retrieval model, that relies on term frequency models like Okapi BM25 or language modeling cannot be applied, as stated by Lin and Mishne [21]. The study of topic endurance on the social network proved that topics are presented in bursts of queries and mentions. In addition the typical usage of twitter for search is not the same of Google. When users are searching on twitter, they want to find out what is happening in that moment, meaning that classification techniques based on past events cannot respond this kind of problem. As stated by Lin and Mishne [21], this problem has not yet been solved at Twitter (or anywhere else at the time of writing this report), and issues a new kind of data analysis approach that was not taken into consideration in the past.

This effect of rapidly changing topics and queries based on real time events was named ”Churn”, and can be clearly seen in Figure 3.4.

By including social features into clustering algorithms, it might be possible to discover interesting rising topics to a specific specific user, by categorizing them through a trained SOM.

3.3 Summary

Ending section summarizing the chapter is typically a good idea.

Ensure that the next chapter starts in a odd page

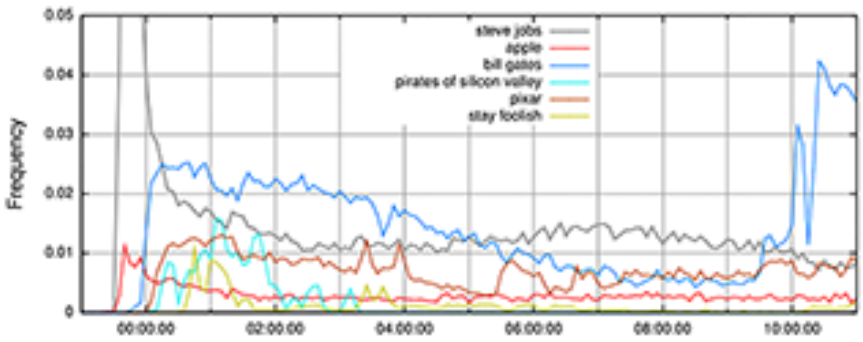


Figure 3.4: The Churn effect: Frequencies of queries related to Steve Jobs death over a 12 hour period in 5-minute intervals, normalized to the total number of queries in the interval. At its peak, the query “steve jobs” reaches 0.15 (15% of the query stream); Graph taken from [21]

4

Clustering Tweets with Self Organizing Maps

Contents

4.1 Twitter Data

Twitter is a social network website and mobile app, where users are able to share what's on their mind with less than 140 characters. Due to its limitations, twitter users started to adopt their own kind of language on the social network, sharing shortened Uniform Resource Locator (URL), and tagging topics with a word preceded with an hashtag – # – became so popular, that it was eventually implemented into twitter itself. Nowadays it is possible to monitor events through hashtags selections and links shared on the mobile app are automatically shortened.

With the rise of smartphones and decent prices for mobile Internet access, and people becoming always online, GPS coordinates were also added to tweets. In fact, a tweet nowadays has a massive amount of information, as can be seen in Figure 4.2.

There are two main ways to gather data from twitter: Crawl twitter HTML pages and scrap the intended information, access through the twitter API.

Crawling web pages is done through the analysis of HTML documents generated by twitter servers. Due to specific semantic rules, it is possible to gather almost all information that is possible to have access through the twitter API. Even though writing an HTML crawler is not particularly complex, specially through the usage of open source tools like nokogiri [1] or beautiful soup [2], Twitter specifically asks to not be crawled in some parts of their URL, as can be seen in Figure 4.1.

Basically, the restrictions that twitter asks on his robots.txt, only allows for search results, and hashtag searches to be monitored, which is pretty limiting.

```
# Every bot that might possibly read and respect this file.
User-agent: *
Allow: /?lang=
Allow: /hashtag/*?src=
Allow: /search?q=%23
Disallow: /search/realtime
Disallow: /search/users
Disallow: /search/*/grid

Disallow: /*?
Disallow: /*/followers
Disallow: /*/following

Disallow: /account/not_my_account

Disallow: /oauth
Disallow: /1/oauth
```

Figure 4.1: Twitter robots.txt piece where it is possible to see what should and shouldn't be crawled.

Since version 1.1, authentication is required in order to access twitter through their API. The authentication mechanism, is used to limit the amount of information users can gather from twitter.

[1] http://www.nokogiri.org/
[2] http://www.crummy.com/software/BeautifulSoup/

The API itself is divided in the streaming – used for subscribing directly to twitter public, user or site streams – and REST – used for programmatic access to read and write to the twitter API.

The streaming API is extremely useful for building datasets based on keywords, searches and entities. A huge amount of tweets can be collected without hitting rate limits, since their API default level lets an endpoint track up to 400 words, and 5000 user ids. As long as the amount of tweets streamed to the endpoint doesn't surpass the 1% of the total amount of tweets Twitter is currently streaming. Although, if some wrong terms are monitored, the amount of spam crawled can be huge, specially when monitoring public entities or trending hashtags.

The REST API can be used to get all information that is available on twitter by the time the request is sent. REST API limits are much greater than the ones applied at the streaming API. The basic rules are 15 minutes windows per endpoint where 15 requests can be made. There are some exceptions to these rules and those can be found on twitter documentation [3].

```
{
  "_id" : { "$oid" : "4fa14bc97e5617025fb14787" },
  "text" : "RT @FastCoDesign: A Paintbrush That Works On The iPad
     http://t.co/eWjEZAga (@sensubrushman)",
  "id_str" : "197701817864421376",
  "coordinates" : null,
  "in_reply_to_screen_name" : null,
  "in_reply_to_user_id" : null,
  "possibly_sensitive" : false,
  "favorited" : false,
  "in_reply_to_status_id" : null,
  "source" : "<a href=\"http://www.flipboard.com\" rel=\"nofollow
     \">Flipboard</a>",
  "possibly_sensitive_editable" : true,
  "contributors" : null,
  "retweet_count" : 0,
  "truncated" : false,
  "in_reply_to_status_id_str" : null,
  "geo" : null,
  "in_reply_to_user_id_str" : null,
  "entities" : { Enteties Object },
  "user" : { User object },
  "retweeted" : false,
  "id" : 197701817864421376,
  "place" : null,
  "created_at" : "Wed May 02 14:59:21 +0000 2012" }
```

Figure 4.2: JSON representation of a Tweet.

Due to the amount of specificity allowed by the REST API, it is better suited to create datasets that mimic the way twitter data is interconnected, like getting users and their followers, as well as their tweets.

[3] https://dev.twitter.com/rest/public

While researching ways to use SOM as a way to find clusters of topics on twitter, we presented some enhancements to the algorithm, which integrates the social network behind the authors of the tweets in the clustering process. In order to analyze socially linked data, a socially linked dataset is needed, and therefor a crawler that stores social connections between users must be implemented.

Given the fact that social relations were required, we opted for making a crawler based on the REST API. Due to the fact that twitter API rate limits would be achieved with some ease, the crawler should be prepared to achieve this maximum amount of requests per 15 minute window and wait until it can crawl again.

Also, at a given time, the crawler should be able to serialize it state in order to able to resume crawling in case it has to stop at any given worldly circumstances.

4.2 SOM

4.2.1 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.
- 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 5.1.2. A visual application of these methods can be seen in Figure 4.3.

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.

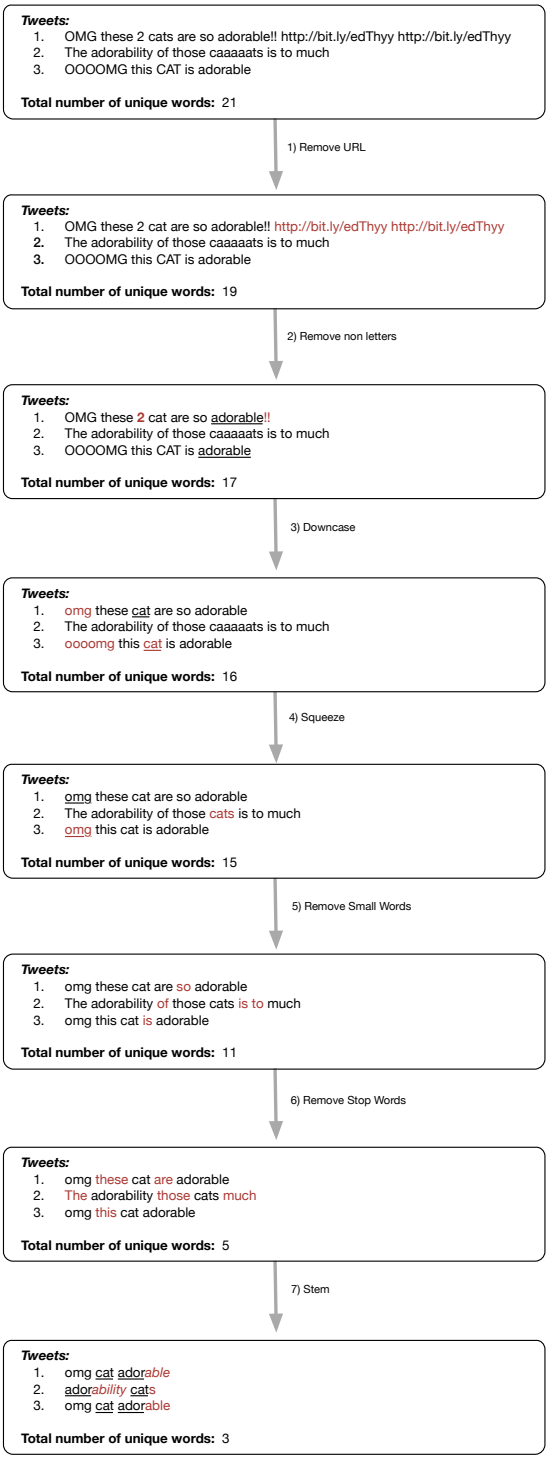


Figure 4.3: Reducing the number of unique words on three tweets about cats. Text in red represents letters removed. Underlined text represents words that due to text transformation became equal.

NLP techniques such as the one described on Subsection 3.2.3 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 4.4.

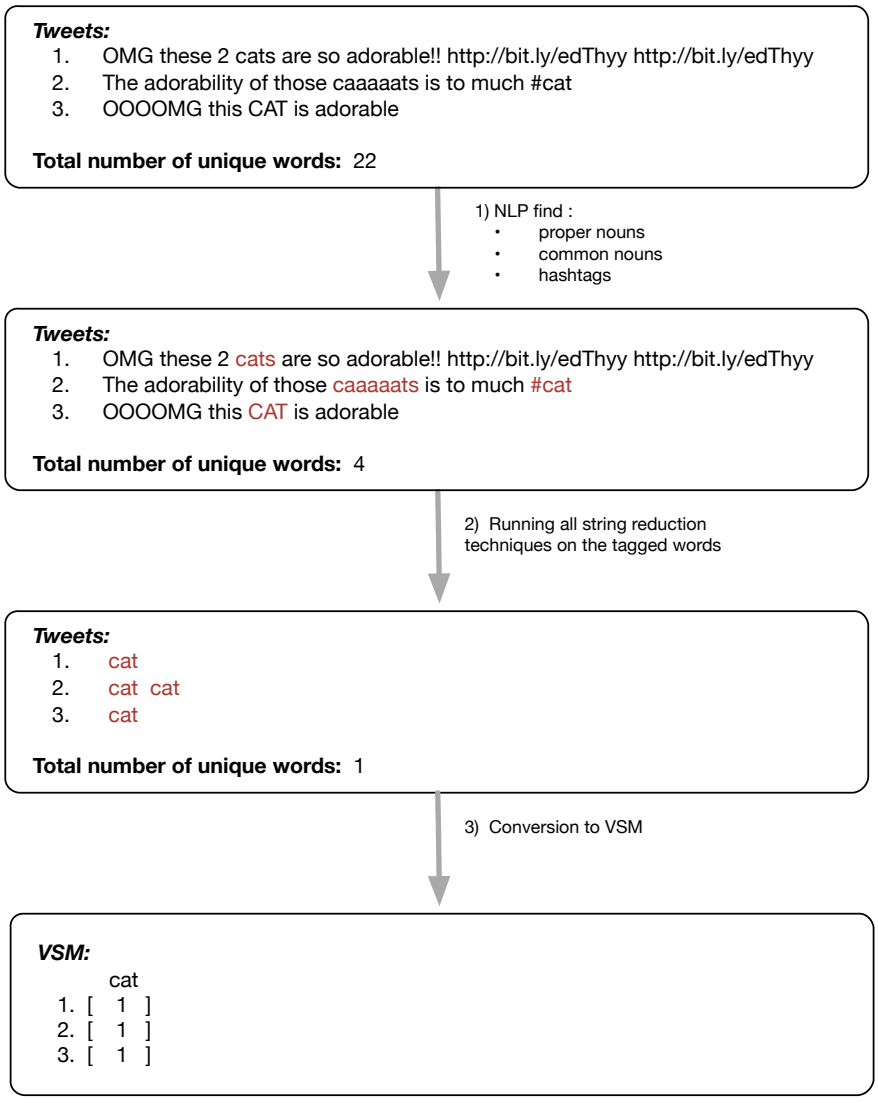


Figure 4.4: Using NLP with string reduction techniques to reduce the VSM size

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.

4.2.2 Extensible SOM Library

When researching ways to extend the SOM algorithm, in order to add social features to the learning process. We found that the number of SOM libraries was not very expense. 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 3.1.1 Geo-SOM, the Growing Hierarchical SOM [29], the time adaptive SOM [32], the Ontological SOM [10], 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, **Q-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 4.3 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.

4.3 Homophilic SOM Definition

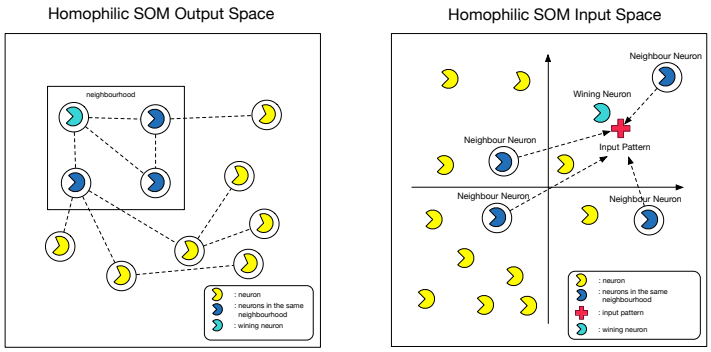
The default SOM algorithm has no idea whatsoever of the social connections between the tweets, it simply looks at the binary vectors that represent sentences and assigns it to the most similar neuron.

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.

4.3.1 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:



(a) The neighbourhood is defined by the relations of followers/followees between the winning neuron and the other neurons

(b) Homophilic input space works in the same way as a normal input space

Figure 4.5: Homophilic SOM output and input space during the learning phase.

- 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.

4.3.2 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. 2.3) 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.

4.3.3 Visualizing Neuron Representation Quality

The homophilic SOM has an output space that doesn't relate to any kind of geometric figure, due to this fact, it is not possible to visualize the clusters formed using U-Matrix. In order to solve this problem, we propose an alternative way to visualize SOM that, instead of focusing on the distance between the neurons on the output space — like the U-Matrix —, it focuses on the mean quantization error applied to each neuron and the input patterns it is representing. We named this method Q-Matrix (Alg. 3) which stands for unified mean quantization error matrix.

Algorithm 3: Q-Matrix

Data: Input patterns $X = \{\overrightarrow{x_{0,0}}, \ldots, \overrightarrow{x_{n,n}}\}$,
Trainned neurons $W = \{\overrightarrow{w_{0,0}}, \ldots, \overrightarrow{w_{n,n}}\}$
let $D_{i,j}$ be the input patterns represented by neuron $w_{i,j}$
Result: Q-Matrix
let Q be an empty matrix of size $n * n$
for $i = 0$ to W_{max} **do**
  for $j = 0$ to W_{max} **do**
    $q_{i,j} \leftarrow avg\_quant\_error(w_{i,j}, D_{i,j})$

```
/* convert the distances to color                                    */
```

$WHITE = 255$
$BLACK = 0$
$q_{max} \leftarrow max(Q)$
$q_{min} \leftarrow min(Q)$
for $q_{ij} = q_{00}$ to $q_{n,n}$ **do**
  $Q_{i,j} \leftarrow (1 - \frac{q_{i,j} - q_{min}}{q_{max} - q_{min}}) * WHITE$

The algorithm works in the following way: for each neuron in the grid, we find the quantization error between it and all the input patterns it represents. Afterwards, we calculate the average between all the quantization errors associated to the neuron, and add it to the Q-Matrix. After this process is applied to all neurons, all quantizations errors are converted to a color that is directly proportional to the amount of the mean quantization error.

Due to the fact that each neuron is a cluster on the homophilic SOM, the Q-Matrix algorithm lets you easily see which clusters have the lower quantization error, and therefor better match their input data. On the default SOM algorithm, the Q-Matrix can be applied in order to easily visualize which neurons don't represent well their input patterns, being an extra tool to analyze training quality. An example of a Q-Matrix can be seen in Figure 4.6.

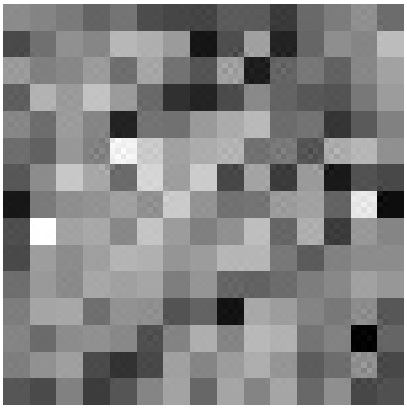


Figure 4.6: Q-Matrix calculated after 300 epochs of training a default SOM algorithm to identify colors. By looking at the matrix it is very easy to see which neurons are not representing their associated input patterns well – in black – and in white the ones that have very little quantization error and therefor are good at representing their input patterns

5

Evaluation Metrics

Contents

5.1 Clustering Tweets with Self-Organizing Maps

5.1.1 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. Due to simplicity of the approach, an overwhelming amount of different words, some, even without any clear meaning. Due to the huge amount VSM size, 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 **??** where used. The SOM training was performed using the R kohonen package [35]. We have added some information about this train on Appendix A, three kinds of clusters where found. Clusters where no topic could be made sense of, clusters with a ton of tweets which had the same text and clusters that had more than one topic/no topics at all. An example of tweets presents in these clusters can be seen in Figures 5.1(a), 5.1(b) and 5.1(c).

5.1.2 Reducing SOM vector size

In Subsection 4.2.1 we introduced string reducer methods which enabled great VSM reduction. On Figure 5.2 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 where able to reduce the VSM size in about 75%.

String reduction techniques work directly with text and has no notion whatsoever of linguistic semantics. I Subsection 3.2.3 we presented a twitter NLP libray which can be used effectively to understand the linguistic semantics used on twitter.

By feeding the same dataset as used above to the library we where able to identify multiple types of words that can afterwards be considered relevant for TDT. On Figure 5.2, the red bars show the amount of words unique words found under a specific semantic tag, whilst in red 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. iKitCloudKicker,guess its a twitter day at work
2. mgdotorg,rick hunolt back in action for more than a song or two just saw this posted on exodus san fran
3. ChoclatelceCian,such a rough and shit sleep lastnight has totally ruined my twitter buzz today l and tea isnt cheering me up o
4. oooocherryoooo,received a postcrossing postcard from finland
5. Carnage4Life,wil google is making a huge and annoying mistake
6. earcario,this is a great invention simply via joleneo
7. __missvalentine,guess my hair is going in a pony stupid rain
8. iMacNoBook,it s crazy how twitter can make or break a relationship
9. MattMullenUK,missed this yesterday google bigquery now out of trial mode brings bigdata for all at a price
10. AtlantaStreet,career work people are freaking out over a conservative whose twitter account keeps getting shut please retweet

(a) Cluster without topics

1. amanzana_com,la aplicaci n en la ipad o el iphone que un m dico debe chest journal diario m dico de la univerdad de stanford
2. Avi_Fogel,nyangels ipad based look of day outfit sales for women bar rafaeli investor and model designers multiple stylist
3. GaryPHayes,multi platform fantasy experience graphic novel ipad app augmented reality app film anomaly this oct pet
4. iOSsociety,via instagram neon museum las vegas ios iphone ipad
5. iznix,what s better than a protective case for iphone or ipad a case that can charge your device too solar cases are
6. itine_utine,edwin andriann anak mu udiiin gak maen puzzle ntr tapi zaman ipad d pake maen tembing
7. Visit_Newquay,we re selling our ipad new so if your in town its free delivery
8. DigitalLyncher,if only spotify premium wasn t so pricy lol rt spotify finally arrives on ipad thanks to new universal update
9. crunchy_f,i would love to get an ipad that s why i entered to win one
10. landmarkpx,younger abstention doctrine kindle ipad landmarkpublications
11. canadanielle,country music hey who s meowing who has an ipad
12. dawsotron,anyone want to buy an ipad gb g locked to network my new ipad rd generation is being delivered tomorrow inbox me if interested
13. CelinaAtFCM,linkedin launches a killer new ipad app business insider
14. miramahira,arinasolehah hahaha pakcik die pun dah takleh wat pape kalu die dah conquer ipad tu
15. GuitarChordz,spotify launches new ipad app with beautiful graphics gapless playback and more please retweet
16. bahrainme,blackberry porsche design p apple iphone s ipad brand new original unlocked blackberry porsche desi
17. ErikBernskiold,linkedin goes ipad native ios
18. 54111145,ipad z

(b) Cluster with the iPad topic

1. studio44salon,i posted a new photo to facebook
2. mynor1,i posted a new photo to facebook
3. Sheikh_Basit,i posted a new photo to facebook
4. Antalyarentacar,i posted a new photo to facebook
5. sagitario1509_,i posted a new photo to facebook
6. nishantgourav,i posted a new photo to facebook
7. DISTINCT89,i posted a new photo to facebook
8. markleeks,i posted a new photo to facebook
9. bjbaloncio,i posted a new photo to facebook
10. pastorbbg,i posted a new photo to facebook
11. Christochelsea9,i posted a new photo to facebook
12. ShaniChaudhary,i posted a new photo to facebook
13. MaderaBrookside,i posted a new photo to facebook
14. eveproducer,i posted a new photo to facebook
15. Imamean1,i posted a new photo to facebook
16. Peetje1960,i posted a new photo to facebook
17. BmacMotorsport,i posted a new photo to facebook
18. emilybr00ke,i posted a new photo to facebook
19. StratfordSchool,i posted a new photo to facebook
20. walkers4u,i posted a new photo to facebook
21. sairussairus,i posted a new photo to facebook
22. delifreshfood27,i posted a new photo to facebook
23. igaurav430,i posted a new photo to facebook
24. MyHolylandOrg,i posted a new photo to facebook
25. saurav_saus,i posted a new photo to facebook

(c) Cluster with the iPad topic

Figure 5.1: Three types of clusters

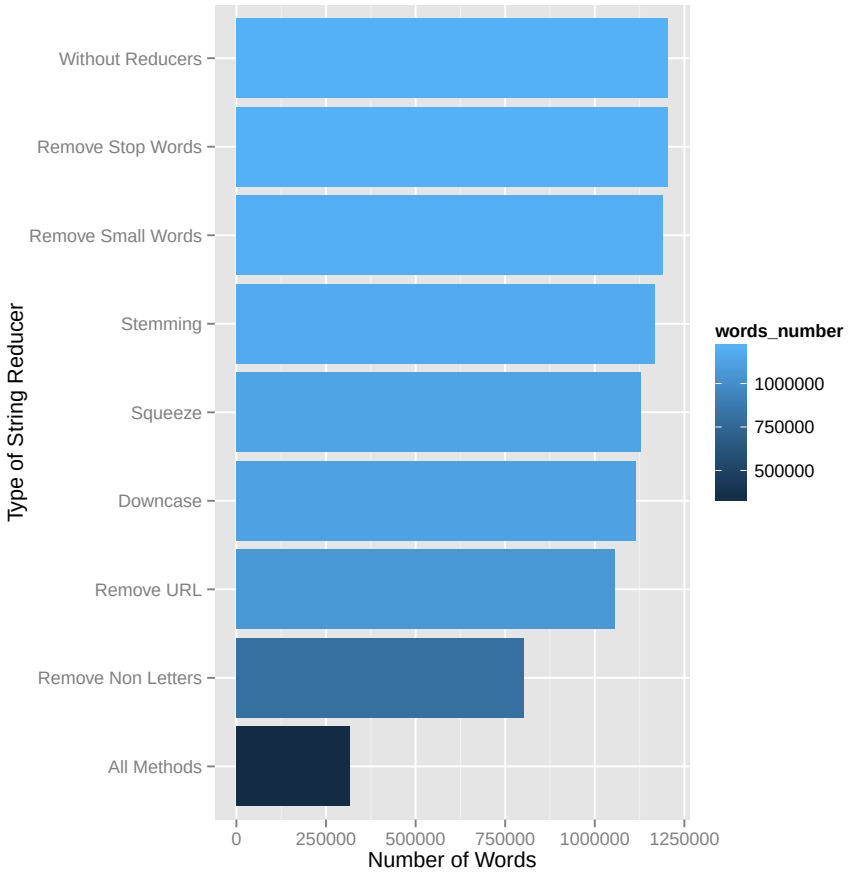


Figure 5.2: Amount of unique words present on dataset sample with 902802 tweets, based on the string word reduction technique applied.

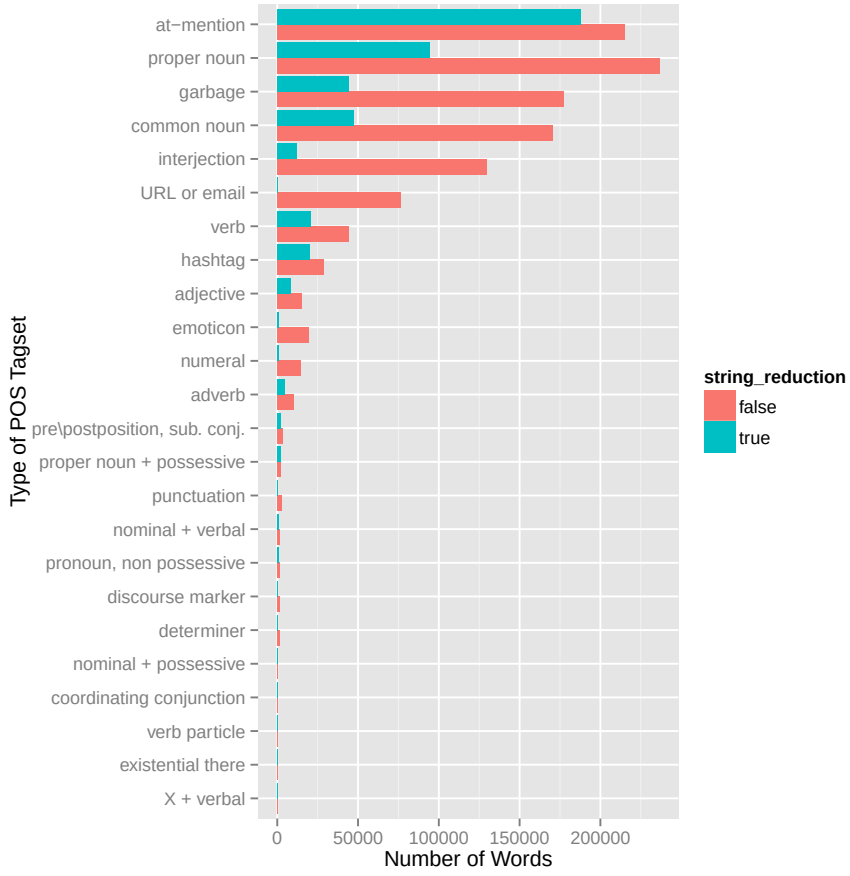


Figure 5.3: Number of words tagged with Ark Tweet NLP. In red we can see the number of words unique words tagged in each category, while in blue we can see the amount of unique words, after applying string reduction techniques.

5.1.2.A Identify Tweets language

Twitter is a social network with users from every corner of the world, and due to this fact, tweets tend to be in a lot of different languages which complicates the process of clustering due to increasing the number of sining ms, idiomatic expression and results interpretation due to lack of knowledge of a specific language. In order to find a tweets language, it is possible to see the language that a user has on his twitter profile through the twitter API, but these are not always the same thing. Some times, users use their profiles on different languages that the language in which they issue their tweets. Identifying tweets that where not in the English language was done through the usage of Ruby library called whatlanguage [1], which tries to identify one language through Bloom Filters. Inside the tweet there is a field which identifies the user language, we found that x is not acurate. Removing tweets that weren't in the english language reduced the amount of different words in x and therefor will reduce the dimensional size of the SOM. In order to prove that the using the whatlanguage library in conduction with the language which users have on twitter will improve the overall percentage of English written tweets, we took a random sample of 100 tweets from our own dataset, and categorized it by hand. Afterwards we compared the results of using only the language on the tweet, using the language detection library and both. The results are summarized on Table 5.3.

Table 5.1: Tweets language detection summary

Total amount of tweets gathered	87
Tweets with user language and tweet language in english	25
Tweets with user language and tweet language not in english	7
Tweets tagged in english and tweet language was in english	8
Tweets tagged in english and tweet language was not	0

Even though the sample is quite small, it was possible to understand that if both techniques where combined in order to detect the language of the tweet, the biggest problem which we could come across, would be to discard tweets that where not tagged as being in English, but in fact where. This is preferable, given that it easier to get more tweets in order to have English tweets, than constructing an VSM with foreign terms.

5.1.3 Conclusions

5.2 Twitter Crawler

After analyzing the work accomplished by clustering tweets with SOM we decided that some alterations to the algorithm should done in order for it to take into account the fact we are deal-

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

ing with socially connected data. In order to accomplish this we needed a dataset with social connections of behind the author of the tweet. There was two ways to accomplish this:

- **First aproach:** For each tweet we have on our dataset, fetch the user information including the users he is connected to.
- **Second aproach:** Create our own crawler, where the social connections, tweets and users are saved.

We followed the second approach in order to have more control over the data used from this moment forth, and to have a better integration of the tweets with the SOM framework, described in Section 5.3.

When designing the twitter crawler, we took into consideration that it had to be extremely resilient in order to be able to be left alone, crawling the twitter, until told to stop. Also if anything happened to the machine where the crawler was running it would be necessary to return to some previous crawling state, with minimum data loss. The crawler works in the following way:

- **Step 1:** Choose some seed users to start crawling or deserialize a serialized version of the crawler if available.
- **Step 2:** For each seed user get all of his followers, and add them to an array if they haven't yet been crawled.
- **Step 3:** Repeat step one with random users taken from the array on step 2, until API limit is reached.
- **Step 4:** When API limit is reached, print the state of the crawled network, serialize the current state, and wait 15 minutes until it is possible to resume crawling.

The crawler is able to get an average of 3 users with their social connections, and 150 tweets per 15 minutes window, it serializes itself to Yet Another Markup Language (YAML) which besides being directly compatible with Ruby, it is also human readable making it easy to parse with unix tools.

5.3 SOM Framework

The SOM framework was developed in the Ruby programing language [2] 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.

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

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.

By default, the base SOM algorithm is implemented as described by the Algorithm 1 in Section 2.2.

5.3.1 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 ate any given moment of the training to export the output space to 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.

The machine used for training had the hardware specifications outlined in table 5.3.

Table 5.2: Test machine one specs

Operative System	OSX 10.9.5
Memory	8 GB, 1067MHz DDR3
Processor	2,4GHz Intel Core 2 Duo
Hard Drive	128GB SSD

A summary of the training is specified in Table **??**

Table 5.3: SOM trainning resumed

Number of Neurons	225
Output Space Size	15x15
Number of Input Patterns	1500
VSM size of Input Patterns and Neurons	3
Number of Epochs	600
Training Duration	14 hours
Type of Train	print each epoch training
Initial learning rate	0.6
Initial Radius	8

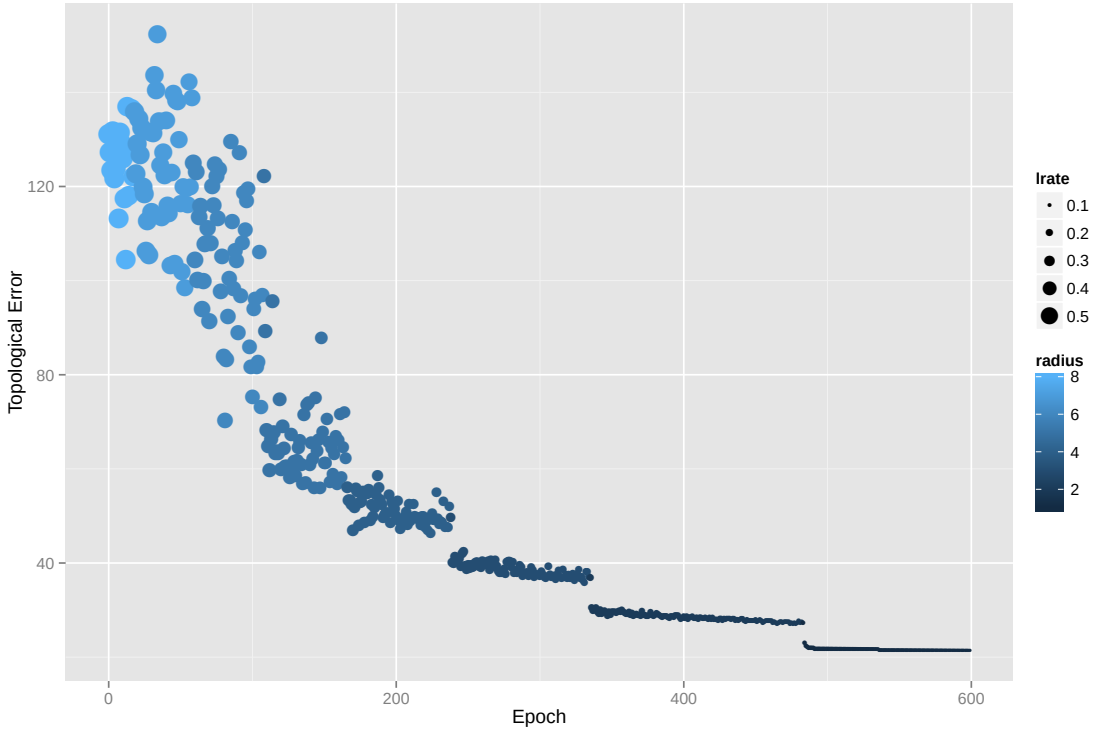


Figure 5.4: Changes in topological error throughout the SOM training, lrate stands for learning rate, and radius for radius applied to the winning neuron

On Figure 5.3.1 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 5.3.1 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

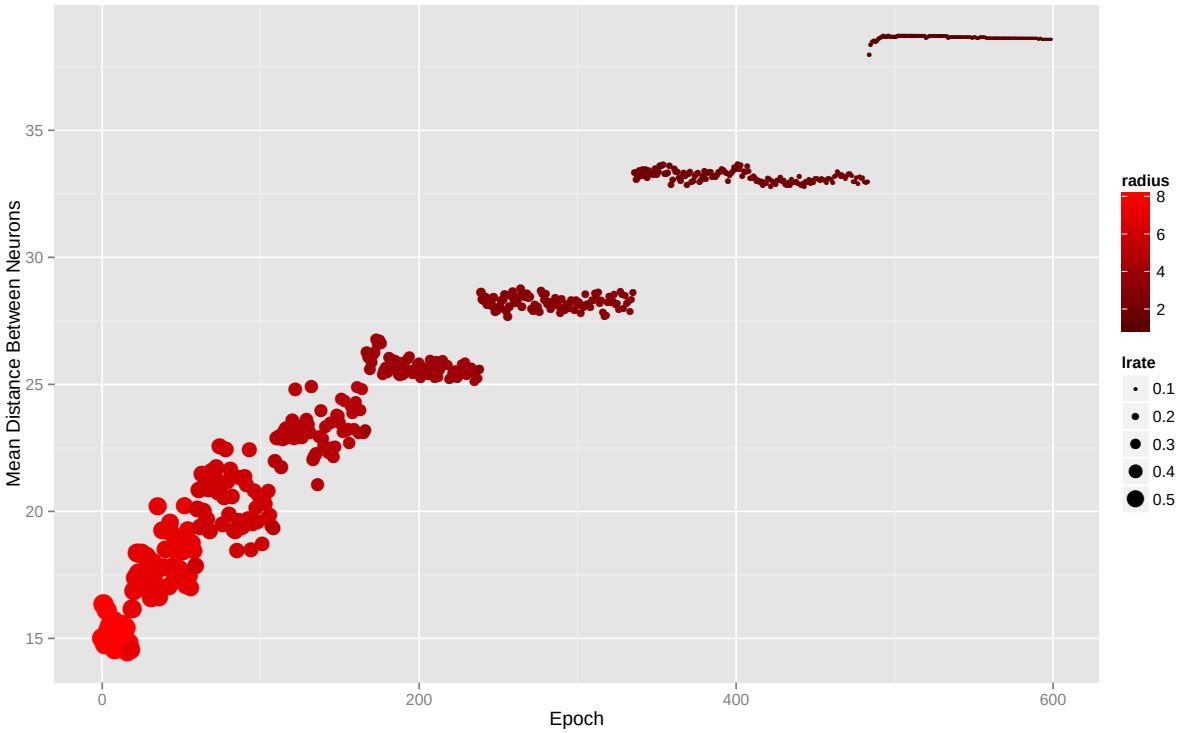
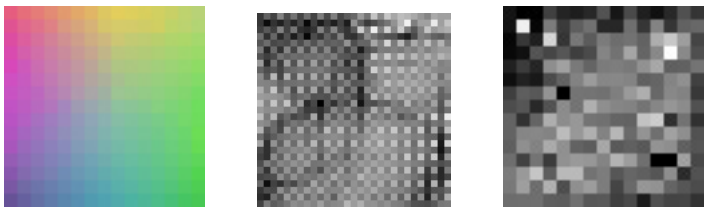


Figure 5.5: Changes in the average distance between neurons, throughout the SOM training

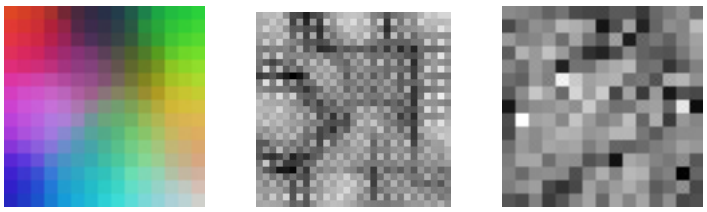
the begging, half of the train, and finally at the end of the training. When comparing the output space on Figure **??** at the staring of the training, it is possible to see a lot less colors that on Figure **??**, this is due to the fact that neurons are getting more specified through the training. The U-Matrix evolves in a way where clusters are almost unnoticeable, which is due to the fact that each neuron is a cluster by itself, and the distance between it and his neighbors was homogenized throughout the output space. The **Q-Matrix!** evolves, by becoming whiter which represents that the mean topographic error is becoming smaller. This was already seen before in Figure 5.3.1, but now we can also see which neurons are worst at representing the input patterns.



(a) Output Space **(b)** U-Matrix **(c)** Q-Matrix

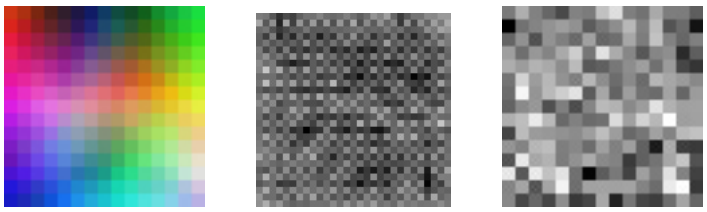
Figure 5.6: SOM state after first epoch of training. Its learning rate is at 0.598, and radius at 8.

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



(a) Output Space **(b)** U-Matrix **(c)** Q-Matrix

Figure 5.7: SOM state after second epoch of training. Its learning rate is at 0.22, and radius at 3.



(a) Output Space **(b)** U-Matrix **(c)** Q-Matrix

Figure 5.8: SOM state after third epoch of training. Its learning rate is at 0.081, and radius at 1.

its input patterns. Afterwards we printed the input patterns associated to that neuron. This process was graphically represented in Figure 5.9 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.

5.3.2 Benchmarking

The SOM was not created with the purpose of being extremely fast, for that there are already very goo implementations like Wittek [37] distributed library for SOM or Wehrens and Buydens [35] 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. For example the code necessary for training the colored vectors on the previous Subsection can be seen in Figure 5.10.

In order to better understand the amount of data SOM framework is able to handle, we benchmarked it on the same machine used in Table 5.3. 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 where summarized on Figure 5.11, where it is possible to see on the

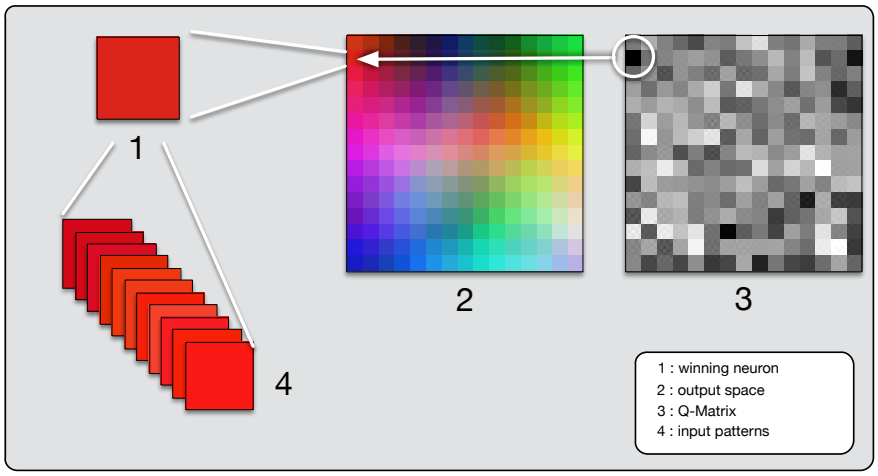


Figure 5.9: Input patterns associated with the neuron with maximum topological error –31. Even though the neuron has the biggest topological error of all neurons, it still has a good representation of the input patterns. The colors in this image are not figurative, and represent the entities at the end of training

```
## Create a new SOM object
som = SOM::SOM.new output_space_size: 15, epochs: 1500

## Generate 1500 random input patterns
som.input_patterns = 1500.times.inject([]){ |arr| arr << Array.new(3){ rand(0..255) }; arr  }

## Start training
som.exec!
```

Figure 5.10: Ruby code necessary for training a SOM with 1500 input patterns.

upper quadrant that if all parameters increase, SOM training will suffer as well.

5.4 Homophilic SOM

In order bring the concept of homophily — love of the same — to clustering socially connected data, on Section 4.3 we suggested some alterations to the default SOM algorithm. These alterations where mainly applied to the output space of the SOM, where in order for the neurons to actually represent users, and the connections between users of a social network.

These features where implemented into the SOM framework described in the previous chapter. The data used to train the homophilic SOM can be seen in Table 5.5. The training was performed on a machine with the characteristics show in Table 5.4.

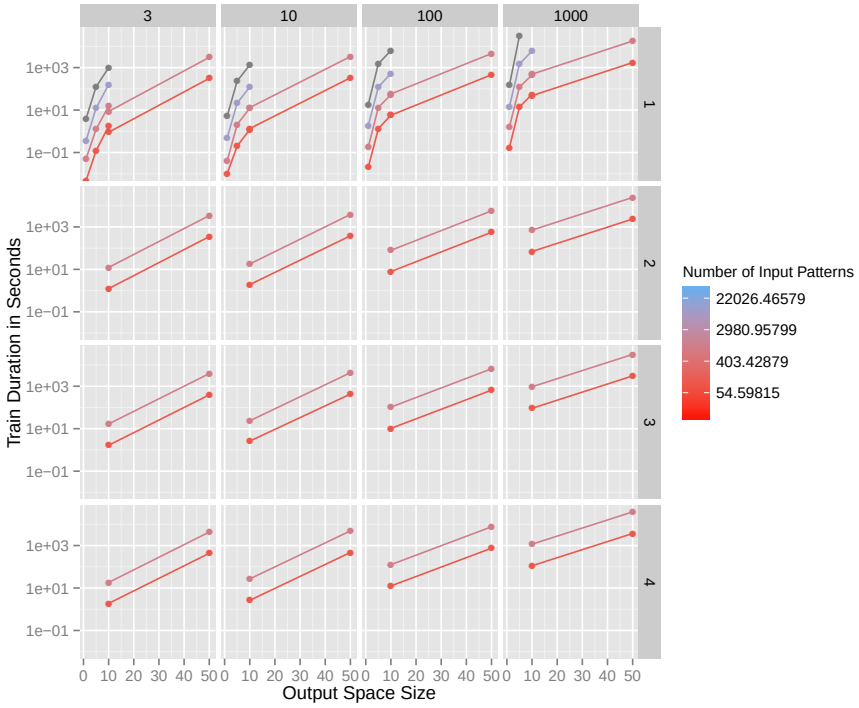


Figure 5.11: SOM framework train duration, influenced by output space size in the x axis, number of epochs in the left, size of input patterns on top, and number of input patterns on the right in color from red to blue.

Table 5.4: Second test machine specs

Operative System	Ubuntu 13.10
Memory	16GB DDR3 Synchronous 1600 MHz
Processor	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
Hard Drive	2TB HDD

Table 5.5: Homophilic SOM characteristics

Number of tweets	1575
Number of users	25
Output space size	100
Input space size	3342 words
Words used for clustering	Hashtags, comon and proper nouns
SVM reductors used	all
Training duration	74h
Initial number of hops	4

It is not possible to draw a U-Matrix due to the fact that the output space from 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 where neurons clearly responsible to identify music, like it can be seen on Figure 5.12(a). On Figure 5.12(b) we can see a cluster about tech and programing, the most surprising part about this cluster is that the tweet about the banana phone, is actually inserted into the tech topic — it was tech project presented at codebits —. On the other hand, some clusters of people saying that they have posted photos on facebook, or that they've liked youtube videos where also found. Even though they can be considered topics, their relevance is not very high.

5.5 Conclusions

1. I think I haven't had a segmentation fault in years http://t.co/COjaaFj6Ib
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 vía @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/HRpBwK3lSB
13. Amazing smart phone foldable computer concept. http://t.co/O8h9ilNqRQ
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

(a) Cluster about tech and programing

1. I think I haven't had a segmentation fault in years http://t.co/COjaaFj6Ib
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 vía @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/HRpBwK3lSB
13. Amazing smart phone foldable computer concept. http://t.co/O8h9ilNqRQ
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

(b) Cluster about music

Figure 5.12: Two clusters with different topics

6

Conclusions and Future Work

Draw your conclusions here and sell your work. Trasmit to the juri how hard it was to develop the presented work.

A future work section is usually here.

Bibliography

[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.

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

[6] Bodt, E. D., Cottrell, M., and Verleysen, M. (2005). Statistical tools to assess the reliability of self-organizing maps. Neural networks, 15(8-9):967–978.

[7] 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.

[8] Cheong, M. and Lee, V. (2010). A Study on Detecting Patterns in Twitter Intra-topic User and Message Clustering. 2010 20th International Conference on Pattern Recognition, pages 3125–3128.

[9] Dozono, H. (2012). Application of Self Organizing Maps to Multi Modal Adaptive Authentication System Using Behavior Biometrics. Applications of Self-Organizing Maps, pages 120–141.

[10] 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.

[11] Hinton, G. E. and Sejnowski, T. J. (1999). Unsupervised learning: foundations of neural computation. MIT press.

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

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

[14] Jansen, B. J., Zhang, M., Sobel, K., and Chowdury, A. (2009). Twitter power: Tweets as electronic word of mouth. J. Am. Soc. Inf. Sci. Technol., 60(11):2169–2188.

[15] Kang, S.-S. (2003). Keyword-based document clustering. Proceedings of the sixth international workshop on Information retrieval with Asian languages -, 11:132–137.

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

[17] Kohonen, T. (1988). The 'neural' phonetic typewriter. Computer, 21(3):11–22.

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

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

[20] 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.

[21] Lin, J. and Mishne, G. (2012). A Study of" Churn" in Tweets and Real-Time Search Queries (Extended Version). arXiv preprint arXiv:1205.6855.

[22] 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.

[23] Macdonald, C., Plachouras, V., He, B., Lioma, C., and Ounis, I. (2006). University of glasgow at webclef 2005: Experiments in per-field normalisation and language specific stemming. In Peters, C., Gey, F., Gonzalo, J., Müller, H., Jones, G., Kluck, M., Magnini, B., and de Rijke, M., editors, Accessing Multilingual Information Repositories, volume 4022 of Lecture Notes in Computer Science, pages 898–907. Springer Berlin Heidelberg.

[24] McCreadie, R. and Macdonald, C. (2013). Relevance in microblogs: Enhancing tweet retrieval using hyperlinked documents. In Proceedings of the 10th Conference on Open Research Areas in Information Retrieval, OAIR '13, pages 189–196, Paris, France, France. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

[25] McPherson, M., Smith-Lovin, L., and Cook, J. (2001). BIRDS OF A FEATHER : Homophily in Social Networks. Annual review of sociology.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

[31] Samarjeet Borah, A. C. (2013). Intrusion detection system using self organizing map (som): A review. SCIENCE PARK, 1(2).

[32] 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.

[33] Tao, K., Abel, F., Hauff, C., and Houben, G.-J. (2012). What makes a tweet relevant for a topic? In Rowe, M., Stankovic, M., and Dadzie, A.-S., editors, #MSM, volume 838 of CEUR Workshop Proceedings, pages 49–56. CEUR-WS.org.

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

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

[36] 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.

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

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

Appendix A

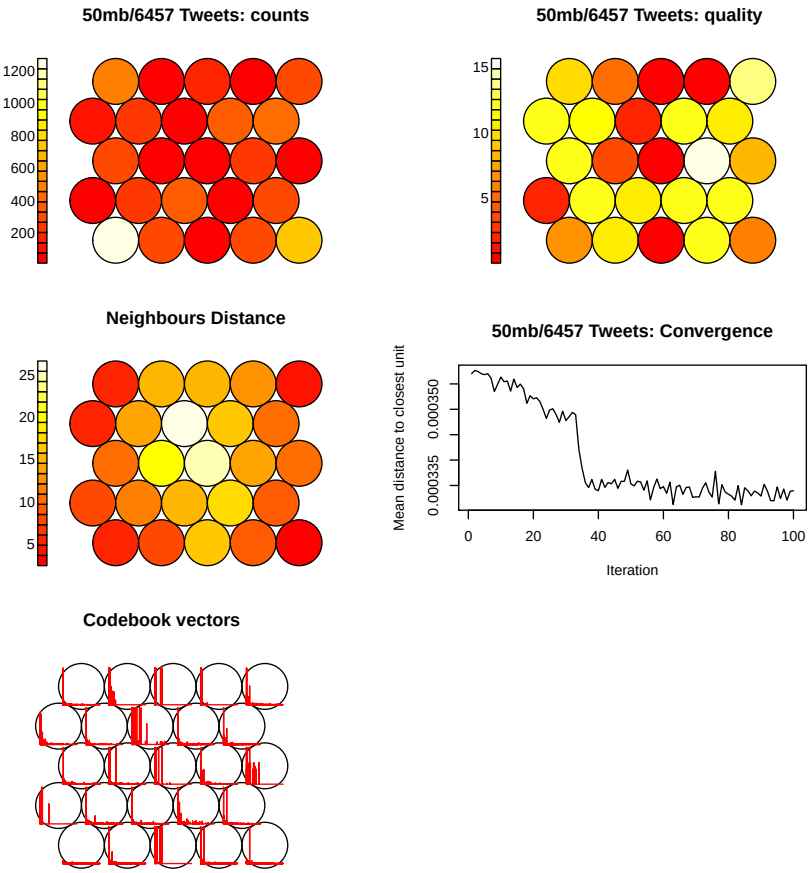


Figure A.1: Training data for 6457 tweets. The counts map, shows us how many tweets are mapped to each cluster. Quality shows the mean distance of objects mapped to a unit to the codebook vector, the smaller the distance the better the representation. The neighborhood distance show the U-Matrix and finally the tweets convergence shows the distance from each node's wheights to the samples represented by that node