Clustering tweets about sports using self-organizing maps

Elsa Scola Martín and Julen Miner Goñi

Abstract

In this document we report our proposal for the application of self-organizing maps to cluster tweets of a specific topic by different sub-topics and the sentiment associated to them. The main topic chosen for this project is ”sports” and the sub-topics are basketball, tennis and football. The performed tasks were: collect the tweets from UK that contain the word ”ball” (which is a common element on the three sports); extract four features, using the FastText library to extract the probability of a tweet corresponding to each of the three sports and the Pattern library to extract the sentiment associated to it; after the feature extraction the data is visualized and the high-dimensional data is clustered using the Somoclu package which is a highly efficient implementation of self organizing maps. Finally, our results show that there is a correlation between sport’s probabilities, therefore, they are valuable for the self organizing map to create very defined clusters, whereas the sentiment is only valuable for clustering when is discretized in two classes. We determine from our results that this technique is an optimal way to discover patterns in new unknown data.

Contents

1 Description of the problem

Internet can be used as one important source of information for machine learning algorithms. In particular, twitter has become a valuable tool of information for companies and social agents. Clustering algorithms have been extensively applied to the analysis of tweets. They can serve to identify tweets with a common topic, select exemplar tweets, or study the similarities between groups of tweets.

In this project we use the self-organizing maps algorithm to cluster tweets about sports, we search tweets that contain the word "ball" as it is a generic sport word and we then want to prove that Natural Language Processing libraries are capable of extracting information from generic texts as human would. As a predefined criterion of similarity between tweets we have chosen the probability of each tweet to talk about basketball, football and tennis, as well as the sentiment associated to it. To do this, we have to extract the features; to extract the probabilities of each tweet to talk about basketball, football and tennis we use FastText which is a Natural Language Processing library that Eneko Agirre mentioned in the special lecture he gave us, whereas for extracting the sentiment from each tweet we use the Pattern library. After the feature extraction, we use the Somoclu package to visualize the clusters, in this way we can interpret the emerging correlations in the data, as well as understand if the sentiment is relevant for clustering the data.

2 Description of our approach

I organized the implementation of the project according to the tasks:

1. Preprocessing of the data set
   (a) Collecting the data
   (b) Feature extraction
2. Clustering and visualization
   (a) Visualize 3D plot
   (b) Visualization of clusters using SOM (continuous sentiment values)
   (c) Visualization of clusters using SOM (discrete sentiment values)
   (d) Visualization of clusters using SOM initializing values with PCA
3. Analyze and contrast the results
4. Questions & Answers

3 Collecting the data

3.1 Creating a Twitter App

There are a number of Python libraries that connect to the Twitter API. Twitter provides a *full list of API libraries* [1] for both Python and other languages, we will be using *Tweepy* [3] for our analysis as it has been successfully used in several studies like in the article about *Application of location-based sentiment analysis using Twitter for identifying trends towards Indian general elections 2014* [2] . In practice, Tweepy is a simple wrapper for the Twitter API. It simplifies the interaction between Python and Twitter's API by handling complications such as rate limiting behind the scenes.

In order to extract tweets for a posterior analysis, we need to access to our Twitter account and create an app. The website to do this is https://apps.twitter.com/. From this app that we're creating we will save the following information in a script called credentials.py:

- Consumer Key (API Key)
- Consumer Secret (API Secret)
- Access Token
- Access Token Secret

The reason of creating this extra file is that we want to export only the value of this variables, but being unseen in our main code (our notebook). We are now able to consume Twitter's API. In order to do this, we will create a function to allow us our keys authentication.

3.2 Collecting tweets from UK

We want to ensure that at least the majority of the tweets we collect are written in English, therefore, once we have the API object instantiated we can use Tweepys geo_search method to query the Geo Search API. One English-speaker country is UK. In this way, we can obtain the UK search ID and add it at the beginning of the query to obtain only tweets from UK. In addition, we use as a filter for the query the word "ball" which is a common word in several sports and it will return non-specific sport tweets; we do this because we are interested in evaluating how Natural Language Processing libraries are capable of identifying the tweets that talk about certain sports.

3.3 Tweet extraction

We first create a function that given a tweet returns True if it is a Retweet, we do this so there are no duplicated tweets on the dataset. After this, we proceed to the data extraction sending our query to the Search API through Tweepys Cursor method and storing the tweets in an array. From this array we create a DataFrame using Pandas in order to manipulate the information easily.

4 Feature extraction

One of the most challenging parts of this project, was extracting the features from the tweets so the self-organizing map could find similarities between tweets. Five features have been extracted: three using FastText and two using Pattern.

4.1 FastText

The FastText Natural Language Processing library [4] has been used to extract three of the features. This library was selected seeing as Eneko Agirre mentioned it in the special lecture he gave us and as it has been used in other successful experiments made by Facebook's research team, as written in the article Bag of Tricks for Efficient Text Classification [5]. Using the text classification tools provided by the library, three supervised classifiers were created, one for each topic: basketball, football and tennis. To train the classifiers, it was necessary to create three files with words corresponding to each topic, its label and examples that show what words do not correspond to the topic. Therefore, each classifier is capable of classifying a text in a given topic.

The text classifiers that FastText provide are probabilistic text classifiers. A probabilistic classifier is a classifier that is able to predict, given an observation of an input, a probability distribution over a set of classes, rather than only outputting the most likely class that the observation should belong to. Taking advantage of the probabilistic text classifiers, all three of the features were created with the probability given for each tweet. Consequently, the value of the features go from 0 to 1.

4.2 Pattern

Pattern is a web mining module [6] built for Python programming language. This module provides some useful tools and the Natural Language Processing tool has been used in this project. This tool provides sentiment analysis functions, used to extract the other features. This module was used as it was suggested in the problem description. There is also an article, Pattern for Python [7], about the usage of the module.

The sentiment analysis function returns a tuple with two values, the polarity and the subjectivity. The polarity value is the important value to know if a tweet has a positive or a negative sentiment and its value goes from -1 to 1. From this value two features were extracted: the polarity as a continuous value (no modification) and the sentiment as discrete value from the polarity. As it is suggested in the official documentation of Pattern [8], the positivity or negativity of the tweet can be extracted comparing the value of the polarity with a threshold. If the polarity is greater or equal than the threshold, the tweet can be taken as positive. The threshold that has been used in this project is the one suggested in the official documentation [8], 0.1. If the tweet was positive, the assigned value to the feature was 1; if the tweet was negative, the assigned value was 0.

5 Self-organizing maps to cluster and visualize the data

5.1 Self-organizing maps

Self-organizing map is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map, and is therefore a method to do dimensionality reduction.

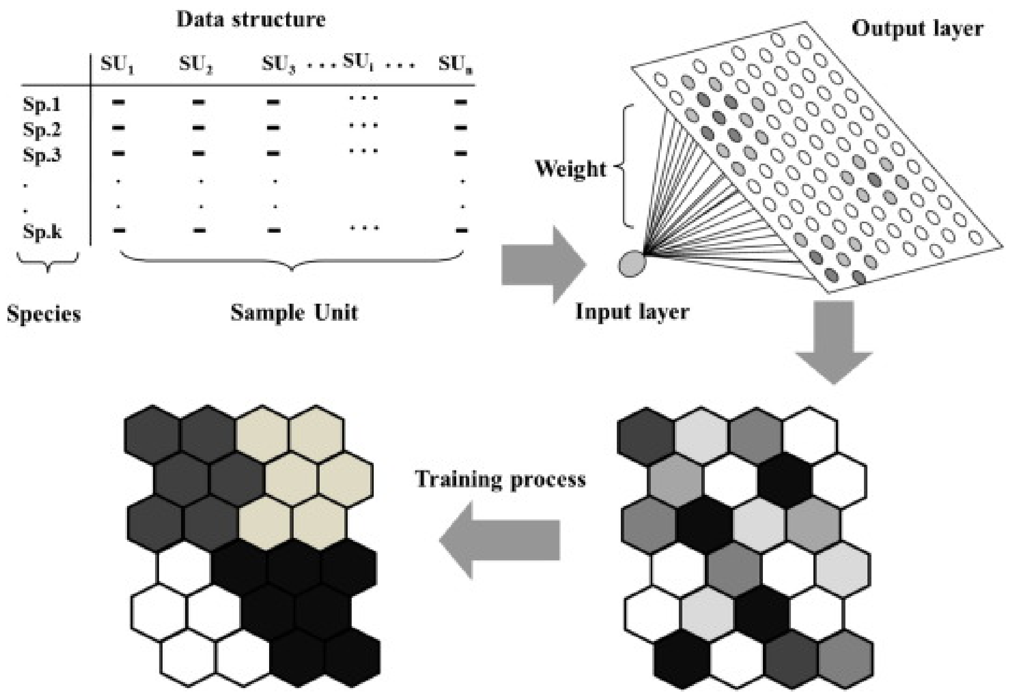

Figure 1: Structure of a Self-Organizing Map

In a SOM algorithm, the neurons are ordered in two layers: the input layer and the competition layer. The input layer is composed of N neurons, one for each input variable. The competition layer is composed of a topological low-dimensional grid of neurons (usually 2-dimensional) geometrically ordered. The number of neurons in the output layer depends on the problem. It may fluctuate from a few to several thousands depending on the complexity of the problem. Finally, a learning algorithm is used in order to calculate the associated weights for each competition layer neuron and a weight update is performed so that the winning neuron is approached to the input observation.

Each input layer unit is connected with every neuron of the competition layer and next, an N-dimensional weight vector is assigned to each competition layer unit.

The algorithm calculates the models that best describe the domain of observations. The models are arranged in the two-dimensional grid so that similar models are closer each other than the different ones. The SOM keeps a neighborhood relation between the original distribution of N-dimensional data and the topological low-dimensional grid.

There are three main ways in which a Self-Organizing Map is different from a standard Artificial Neural Network:

- A SOM is not a series of layers, but typically a 2D grid of neurons (see Figure 1)

- They dont learn by error-correcting, they implement competitive learning
- They deal with unsupervised machine learning problems
- Access Token Secret

Competitive learning in the case of a SOM refers to the fact that when an input is presented to the network, only one of the neurons in the grid will be activated. In a way the neurons on the grid compete for each input.

The **unsupervised** aspect of a SOM refers to the idea that you present your inputs to it without associating them with an output. Instead, a SOM is used to find structure in your data. The SOM can be used for finding structures in the data, dimensionality reduction and data visualization.

5.2 Somoclu for self-organizing maps

Self-organizing maps are computationally intensive to train, especially if the original space is high-dimensional or the map is large. For this reason, we have decided to use Somoclu to fast train the self-organizing maps.

”Somoclu is a massively parallel tool for training self-organizing maps on large data sets written in C++. It builds on OpenMP for multicore execution, and on MPI for distributing the workload across the nodes in a cluster.Apart from fast execution, memory use is highly optimized, enabling training large emergent maps even on a single computer.” *Somoclu: An Efficient Parallel Library for Self-Organizing Maps* [9].

5.2.1 Planar maps

Once that the self-organizing map has learned from the data, we can visualize the planar maps or component planes that had been generated, one with each feature.

By component plane representation we can visualize the relative component distributions of the input data. Component plane representation can be thought as a sliced version of the self-organizing map. Each component plane has the relative distribution of one data vector component. In this representation, dark values represent relatively small values while lighter values represent relatively large values. By comparing component planes we can see if two components correlate. If the outlook is similar, the components strongly correlate.

In the following images the component plane’s slices are shown, the neural network learns progressively for each slice (each slice represents the learning process for each feature).

In Figure 2, on the left, can be seen the component planes of the SOM with continuous values for the sentiment, whereas in the Figure 2, on the right, the sentiment is represented with discrete values. Further analysis of the results can be found in the ”Results” section.

5.2.2 Initialization with principal component analysis

Principal Component Analysis (PCA)

The main goal of a PCA analysis is to identify patterns in data; PCA aims to detect the correlation between variables. If a strong correlation between variables exists, the attempt to reduce the dimensionality only makes sense. In a nutshell, this is what PCA is all about: Finding the directions of maximum variance in high-dimensional data and project it onto a smaller dimensional subspace while retaining most of the information.

PCA when initializing the SOM

Even if the data is high-dimensional (as there are four features), we can see that there is little impact on our results when applying PCA before training the neural network (see Figure 3) as this technique is more orientated to apply it on data sets with more dimensions, for this reason it is more convenient to use the Principal Component Analysis procedure to reduce very high-dimensional data without losing coherence.

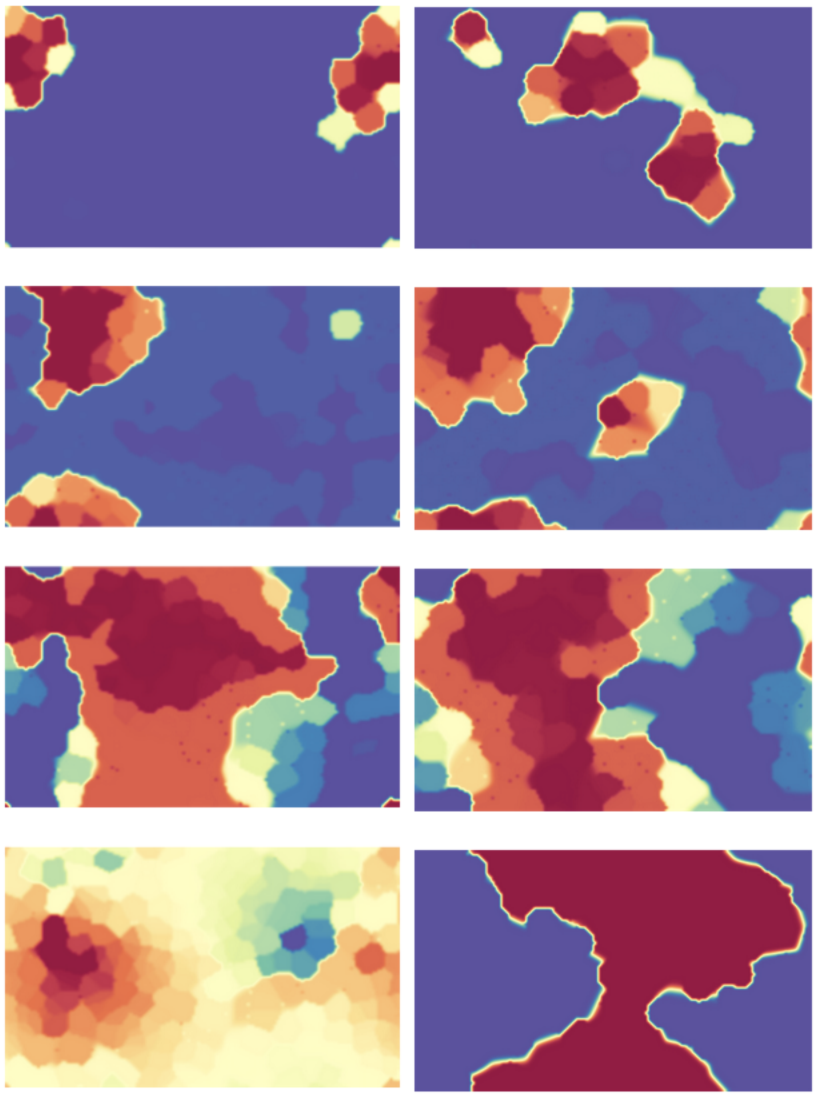

Figure 2: Component planes of the SOM: continuous values for sentiment on the left, discrete values for sentiment on the right

5.3 Results

We can deduce that the self-organizing maps take related values when extracting relevant information. The probabilities extracted to the tweets are the features that more information gave to the self-organizing maps, as they are related. When visualizing the data in the 3D plot using the probabilities, we can see that there are huge groups of points in the corners of the plot, meaning that the important information of the cluster must be taken from this features. The sentiment of the tweets is used by the self-organizing maps to separate clusters about topics into smaller clusters.

In the first self-organizing map that we computed, we used the continuous polarity value as the fourth feature for the learning data. As we can see in the planar maps generated in the learning process, the three first planes are highly correlated, what leads us to conclude that these features are very correlated. In those planes, we can see that the groups are being formed as they grow or shrink. However, when we analyze the fourth plane, we can see that there is no correlation between this plane and the previous ones. This happens because there is no correlation between the polarity feature and the probability features. We can also observe that the colors of this plane look reasonably random. This is because the Euclidean distances between the points of the polarity are

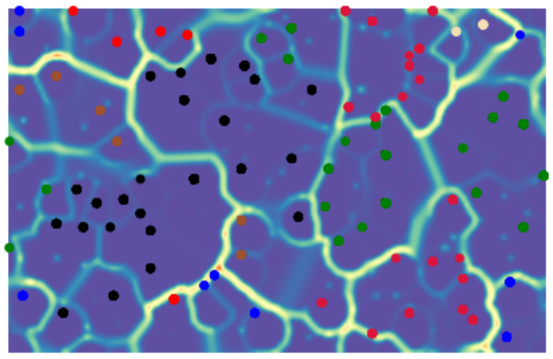
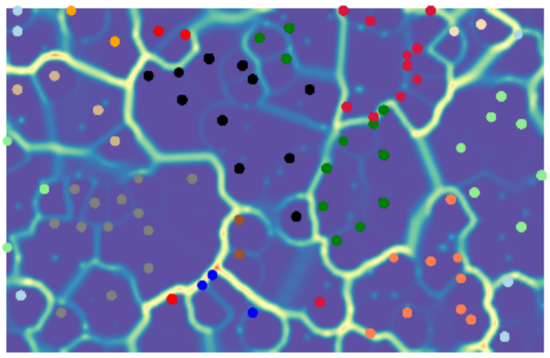
Figure 3: Results applying PCA initialization

very small. When showing the U-matrix of this self-organizing map, we can see that the clusters have been constructed based on the topics extracted from the tweets. When showing the colors of the sentiment, we can see that the differentiation is made inside the topic clusters, even though that the different sentiment points are sometimes mixed. This is because the values of the polarity were continuous.

In the second self-organizing map that we computed, we used the discrete sentiment value as the fourth feature for the learning data. As we can see in the planar maps generated in the learning process, the three first planes are highly correlated, as before. In those planes, we can see that the groups are being formed as they grow or shrink. However, when we analyze the fourth plane, we can see that there is no correlation between this plane and the previous ones. This happens because, again, there is no correlation between the sentiment feature and the probability features. But there is a huge difference between this self-organizing map and the previous one, more specifically in the fourth planes. In this case, the fourth feature shows a very sharp groups in the plane. This is because the Euclidean distances between the points of the polarity are very large, as the values of the feature are 0 or 1. When showing the U-matrix of this self-organizing map, we can see that the clusters have been constructed based on the topics extracted from the tweets. When showing the colors of the sentiment, we can see that the differentiation is made inside the topic clusters. Nevertheless, those sub-clusters are more differentiated than the previous ones.

In the third self-organizing map that we computed, we used PCA initialization. In this problem, we can see that the difference between the the U-matrix shown by the second self-organizing map and this PCA initialized self-organized map is not very relevant and does not add any value to the result of the given problem (Figure 4).

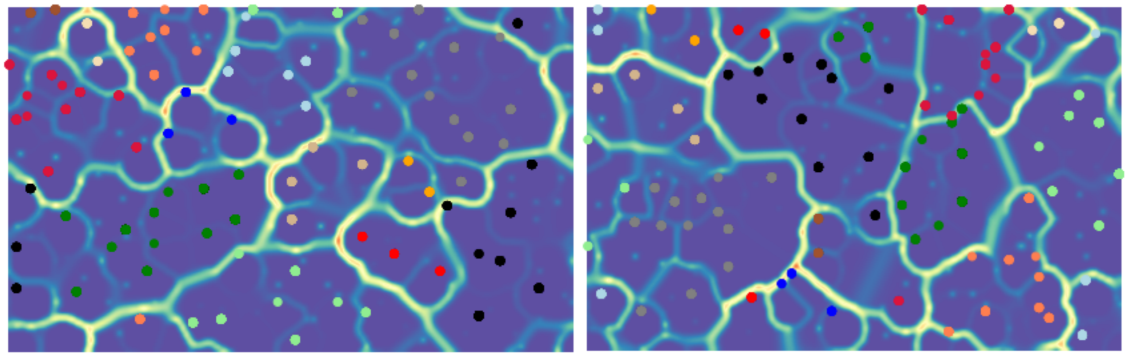

Figure 4: With default initialization on the left, with PCA initialization on the right

6 Conclusions

To put it in a nutshell, the results obtained in the project have demonstrated that the probabilities of each sport are correlated and, therefore, the self-organizing map can cluster tweets in strongly defined groups based on them. Furthermore, we have concluded that the discrete values for the sentiment feature contributes to a more defined separation between clusters as it is more relevant for the neural network. Finally, this results show that self-organizing maps can be useful in many situations to discover patterns on new previously unseen data, for example on the marketing department of companies, for clustering the preferences of clients to offer them products on the same group that the ones they consume.

7 Questions Answers

1. **What class of problems can be solved with the NN?**

   The neural network can be used to solve pattern recognition problems. It solves unsupervised problems, as it is explained in section 5.1.

2. **What is the network architecture?**

   The self-organizing map consist of two layers, the input layer and the output/competition layer. The parameters of the self-organizing map are the features and the values of the weight vectors. SOMs in these domains have generally been restricted to static sets of nodes connected in either a grid or hexagonal connectivity and planar or toroidal topologies.

3. **What is the rationale behind the conception of the NN?**

   The goal of learning in the self-organizing map is to cause different parts of the network to respond similarly to certain input patterns. This is partly motivated by how visual, auditory or other sensory information is handled in separate parts of the cerebral cortex in the human brain.

4. **How is inference implemented? Type of prediction or type of inference process.** & **What are the learning methods used to learn the network? Algorithms used for learning the network.**

   As it is explained in section 5.1, the algorithm calculates the models that best describe the domain of observations. The models are arranged in the two-dimensional grid so that similar models are closer each other than the different ones. The SOM keeps a neighborhood relation between the original distribution of N-dimensional data and the topological low-dimensional grid. Finally, a learning algorithm is used in order to calculate the associated weights for each competition layer neuron and a weight update is performed so that the winning neuron is approached to the input observation.

8 Implementation

All the project steps were implemented in Python. We used Tweepy, Pandas, Numpy, FastText and Pattern for preprocessing the data (collecting the data and feature extraction) and somoclu for the self organizing map. We illustrate how the implementation works in the Python notebook `Project21-Scola-Miner-Notebook.ipynb`. Inside of the compressed file that we submitted, there are three txt files, those are the training data files. There are two csv files which contain data that we collected previously. This data can be loaded following the notebook's instructions.

References

[1] Twitter developer documentation: https://developer.twitter.com/en/docs/developer-utilities/twitter-libraries

[2] Omaima Almatrafi, Suhem Parack, and Bravim Chavan. 2015. Application of location-based sentiment analysis using Twitter for identifying trends towards Indian general elections 2014. http://dx.doi.org/10.1145/2701126.2701129

[3] Tweepy documentation: https://tweepy.readthedocs.io/en/v3.5.0/

[4] FastText webpage: https://fasttext.cc/

[5] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 .

[6] Pattern webpage: https://www.clips.uantwerpen.be/pattern

[7] Tom De Smedt and Walter Daelemans. 2012. Pattern for Python. CLiPS Computational Linguistics Group. http://www.jmlr.org/papers/v13/desmedt12a.html

[8] Pattern documentation: https://www.clips.uantwerpen.be/pages/pattern-en#sentiment

[9] Wittek, Peter & Gao, Shi Chao & Lim, Ik Soo & Zhao, Li. (2017). Somoclu: An Efficient Parallel Library for Self-Organizing Maps. Journal of statistical software. 78. 1-21. 10.18637/jss.v078.i09.