19 de junio de 2015

Análisis de comportamiento en tiempo real de los usuarios de Twitter

Universidad Carlos III de Madrid

Aarón García-Cuerva Fernández

Autor

José Antonio Iglesias Martínez

Tutor

Grado en Ingeniería Informática (Plan 2011)

Universidad Carlos III de Madrid

ABSTRACT

Introduction

The purpose of this work is to develop a tool that analyzes Twitter profiles in real time. The framework where every part of the project is included is detailed in each section, establishing the relationship between each one of them. The dataset analyzed comes from profiles of the social network Twitter, and the analysis algorithms used in this work are based on the philosophy of Evolving Systems. Thanks to them we can create models representing all the data analyzed with a low cost storage.

The decision to develop an application of this type is based on the rise that nowadays has the analysis of data from social networks. Social networks today are a constant reflection of the daily lives of many people, and thanks to the massive data collection and analysis of this data we can obtain useful knowledge applicable in multiple areas. In our case, Twitter is the third most used social network worldwide, so its influence in society is very high.

Some possible applications of our tool are for example: advertising campaigns, television events tracking or analysis of the impact of broadcasts on Twitter.

State of the art

Within the scope of our study must take into account the current situation of the two main factors: analysis of data from social networks and Evolving Systems.

As noted above, the impact of social networks in society nowadays is very high, so an analysis of the data collected has a great value to business and sociological level. This analysis is also done in many cases by their own social networks to provide personalized services to its users.

If we focus our attention on our case and look for related studies, we can see that very interesting applications are related with spam. In particular, we can pay attention to three studies that can be understood in a row. The first deals with the detection of abnormal profiles on Twitter, the second on the detection of users as spam, and the third on adaptive algorithms that can overcome the manoeuvres of spammers to avoid detection. As we can see, there are lines of research closely related to the aim of our project; however the advantage of our tool in this field is its adaptation to changing situations and its little storage required.

Moreover, we should distinguish that this work is not merely a theoretical study, but is a tool that allows analysis according to the user's interests. During the search for similar applications we have found multiple web applications, but most of them are based on the analysis of individual data for each profile. In any case modelling techniques of massive data and general trends are applied. In addition, other tools found made an analysis based on Text Mining

techniques, so in that case we cannot compare them with our project since there is no direct relationship.

In addition, and this is an important advantage of our approach, we need to consider that the profiles on Twitter are not fixed but rather they evolve/change. Thus, it is essential to take into account these changes when we analyse a profile. For this reason, we have applied in this work the algorithms in which the Evolving Systems are based. These systems allow the profile to be dynamic, to evolve.

The cornerstone of these Evolving Systems is the algorithm ***RDE***. This algorithm calculates the recursive density within the current dataset. It is important to highlight that this calculation is done recursively. Thus, since it is not necessary to store all the data to calculate the density, this algorithm can be applied in real time in a very effective manner. In addition, by analysing the density of the different data, we can display the correlation between data, and identify the strangest examples of the set (*outliers*).

Taking into account the RDE algorithm, we can use it for clustering and classification:

eClustering is an approach to clustering based on the density of the data which we want to cluster.

eClass0 is a cluster-based classifier which generates a set of clusters for each possible class in the training set. In the testing phase, an example is assigned to the class whose similarity is greater (different similarity distances could be used).

System analysis

In the phase of system analysis the idea of the project is formalized to a more technical aspect. The formalization is done by defining system requirements, which are the basic pillar for the development of our tool. From them it is where they must derive the design decisions.

The classification of the requirements is performed according to the nature of each requirement. On the one hand the functional requirements that are related to the core functionality of the system, and on the other hand, the non-functional ones, that deal the technical aspects of which the user is not conscious a priori, but it is necessary to take into account to the proper compliance of the functional requirements.

Within functional requirements, it is been collected the need to offer the user the necessary mechanisms to create and store analysis, and to extract and analyze data according to the analysis algorithms chosen. Basic aspects of navigability and tool interface are also discussed.

Moreover, the non-functional requirements deal issues such as visual feedback during the internal work of the program, reduced time for analysis and management of file-level data to store information of an analysis between different sessions.

After seeing a generic approach to our tool and having made a brief technical description through the requirements, now we must further develop the program design itself. The phases of work performed are as follows:

Data extraction

The extraction of data from Twitter is performed by its public API. The program implemented for it is coded in Java, using a library to use the Twitter API. The overall dynamic of profile extraction is to search tweets according to the search word chosen by the user. Then, the user who posted this tweet is extracted and the data of his/her profile are collected. Additionally, in order to use this data later in the algorithms used for analysis it is necessary to make a previous pre-process.

Since for each Twitter search we only can get a maximum of 100 tweets (in our case related to 100 profiles), it is desirable to obtain a greater number of profiles in one run. Therefore, it was decided to conduct a global survey of density of publication, so that in each run several searches are carried out in different geographical areas, selected according to the observed densities. The selected locations are called geo-locations and are stored in an editable file.

The data that have been selected for the analysis are those giving by the Twitter API. However, not all of them can be used, so it has been necessary to remove some of them for the following reasons: 1) identifying values (unique values, which offer neither repetition nor trend), 2) text fields do not valid to our analysis, or 3) simply fields with constant values or very little variety.

Adapting the format of the data from the direct extraction of API to how they should be processed by analysis algorithms is different depending on the type attribute we are treating. The fundamental requirement is to get all numeric fields, because the analysis algorithms proposed work with numerical values. There are four types of adaptation:

- Enumerated fields: Those attributes with repetitive values in a range treatable. In our case are the language and time zone. The pairs of values index-value are stored in a separate file for each enumerated type.

- URL: Each profile can have a personal URL specified. It is a text field and we could say that almost identifier, so we could discard it. But to keep this information, it has been decided to turn it into a Boolean field whose values 1 and 0 respectively correspond to the existence of a defined URL or not.

- Creation date: Although Twitter offers the full date of account creation, we only use the most useful values for our analysis: year and month. Thus, we create a single numerical value composed of the last two digits of the year and two digits for the month. So we keep the chronological relationship and a useful semantic with a

decreased value.

- Colours: The coding offered by the Twitter API is a hexadecimal colour coding, which consists of 6 hexadecimal digits. These digits represent, grouped in pairs, the intensity of each of the RGB colours. Therefore we must reduce this coding so that the total number of colours represented is lower. The solution is a conversion of values, from base 16 to base 4. In this way we obtain a total of 64 representing values of the original encoding, keeping the similarity between them (similar values are grouped). The only drawback is that there not exists a linear relationship in the scale of values as they are three factors that influence (Red, Green and Blue) and cannot be grouped linearly.

To complete the data extraction program a graphical interface as a feedback to the user has been added. Thus the progress of extraction and the final information is displayed. A log file, in which all relevant information extraction is stored, is also used. This information will be useful for the interpretation of the test results.

MATLAB program

The tool has been implemented in MATLAB. The choice comes from the need to compute large amounts of matrix data in the shortest time. It consists of a graphical interface that groups the modules for each of the required functionality.

The program allows you to create a new analysis or continue an existing one. Each analysis is identified by the time range in which it will be develop and representative search word (both selectable parameters by the user). Once the environment is already set and the data were extracted, the user is allowed to perform the analysis with RDE, eClustering and eClass0 algorithms. All the necessary data to resume later analysis are stored in a *.mat* file associated with that analysis. The parameters of each algorithm and considerations are detailed below:

- RDE: There exists a parameter called 'sigma' that acts as a threshold in the detection of *outliers*. The program automatically normalizes the data to better perceive the similarity between them. The analysis is individual for each run, so that no information is stored between sessions. Running the algorithm returns as output the list of *outliers* and a graphic with the density of the analyzed examples.

- eClustering: In this case the data is not normalized for two reasons: first, by the diversity of data, both among attributes and between sessions (the minimum and maximum reference value can change between sessions), and second to maintain a magnitude interpretable in the offered results. There are no configurable parameters because they are considered as a set as default in the code. Cluster models generated are stored in the analysis file. The outputs are provided by the algorithm. These outputs are: 1) The information both textual and graphical of clusters generated, 2) Details of the process (number of clusters and number of examples associated with each generated cluster).

- eClass0: There are two user-configurable parameters for analysis: the percentage of examples used in the training phase and the class attribute. The algorithm uses the last data column receive as a class attribute, so in the code we must change the order of the attributes in the data array depending on the selected attribute. Keep in mind that each performed classification generates a different pattern, which has to be treated individually respect to other classifications.

  However, and this is an important aspect in the algorithm used, it is not necessary to keep in memory all the attributes available as potential qualifying attributes. Not all the attributes are suitable to be treated in that way because a high number of possible classes in an attribute increase significantly the cost in memory and therefore the execution speed. Therefore it was decided to dispense with the date attributes, colours and enumerated as class attributes. In contrast, linear numerical range attributes (Tweets, Favourites, Followers, Followed and Lists) are discretized by ranges in 5 class values. Thus we make treatable the classification process.

  The information provided by this algorithm is the common of a sorting process: number of trained examples, hit rate, runtime, confusion matrix, number of clusters generated by class and detail of the attributes of each cluster.

Experimentation

In the experimental phase we have carried out tests to verify the effectiveness and usability of the program. First, a standard execution analysis is performed, creating it from scratch. The process of extracting data from Twitter is realized according to the selected search word in order to make analysis with each of the available algorithms. For each of them, it is explained the interpretation of the generic results obtained.

However, the process of experimentation is completed by comparing the previous base case with similar situations but with significant changes. Three executions were carried out in order to compare them with the rest and can draw conclusions.

First comparison consists of an analysis that is performed at different time, that is to say, it is carried out following the existing analysis to see how it influences over time in the results. For the interpretation of the results they are used both the outputs generated by the algorithms themselves and the log file generated during the extraction of data. Within each of them, we can get useful information that allows for firm conclusions about the correlation of the data and functioning of algorithms.

After that, the test basis is compared to a different analysis changing the search word. In this case the differences are greater and that completely changes the basis of the analysis, which is the extracted data. For this reason, it is easier to make a comparison with reference to the particularities of each analysis.

Finally, and derived from the above test, we execute an analysis changing the order of the geo-locations used in the search of profiles. The aim is to demonstrate the influence of this order in the results, since the algorithms operate such processing examples sequentially and in linear order. As a result of the comparison it has been showed that the influence of this factor is critical for the three algorithms. This means that at the time of making a particular analysis is necessary to analyze the nature of the extracted data, modifying the order of geo-locations if it is deemed appropriate to improve the analysis results.

Planning and Budgeting

To complete the project, the planning and budget followed are described. The objective of this section is to frame our work in a labour framework adapted to a real environment.

The planning followed has been influenced by the parallelization of the project with other university subjects, so that the average hours spent per day throughout the period of the project is estimated at low (2.5 hours / day). In total, it has been used 140 days, representing 350 hours of effective work.

Furthermore, the calculation of the budget is based on three factors: human capital, hardware and software used. The price for each part is estimated as approximate data set by the student. In total, we have concluded with a budget of 8858'68 €.

Conclusions and Future works

Finally, the conclusions of the work done and a series of future work are explained, derived from the needs identified during the testing phase. Most of them are related to the usability of the tool, making easier to the user the configuration of parameters and the adaptation of certain features according to the particular circumstances of each analysis and data extraction.

INTRODUCTION

Goal

The main goal of this project is **to develop a tool to analyze profiles of Twitter with algorithms based on Evolving Systems in real time**. Therefore, this study aims to check the result of combining the two factors, based on the possible existence of representative patterns in the analyzed data in real time.

In order to develop our work it will be necessary a preliminary study of the performance of Evolving Systems (particularly the algorithms that we are going to use) and the extraction of Twitter profiles in real time. A crucial part will be to design the system according to the individual requirements of both environments. The types of analysis that we want to get are:

- Analysis of the density of the current dataset, that is to say the relationship between them.
- Clustering of the dataset analyzed, looking for a representative model of the existing profiles on Twitter.
- Classification profiles according to their attributes.

Once developing of the tool has been finished, we will proceed to perform various analyzes to check the effectiveness of it, applied to the used dataset. Consequently it is not intended to get good test results, but the development of the tool to easily evaluate the results.

Motivation

Nowadays is great the influence of social networks in any sector whether business, brand or simply informative. Starting with the phrase “If you are not online, you do not exist”, it is clear that a good way to “be” for any company or organization is managing profiles on social networks, because it is a direct way to reach people. Recent studies show that Twitter is ranked as the third most used social network worldwide, only behind Facebook and YouTube. It is estimated that there are 500 million active users on Twitter [1], so the results detached from our program are guaranteed by the great impact of the network in which is based.

However, it is not enough to have a profile on one or more social networks, but what really matters is to know their functioning to get profit of it. With this knowledge it will be possible to decide what to publish, how and when to achieve the greatest possible impact. This is where comes in our tool, allowing to analyze the kind of Twitter profiles of a given sector by a search word (that can be related directly to a subject).

Therefore, situations in which our tool can be useful are diverse. An interesting feature will be to target certain actions to a limited group of users, maximizing the impact and optimizing the number of publications and broadcast time spent. Here are some examples of use:

- Advertising campaigns: Any company or organization interested in promoting an advertising campaign will be able to specify a sector of users marked as target.
- Broadcasting of television events: Nowadays any television program offers to interact in social networks by hashtag. With our tool, you can analyze which are the main types of users that are interacting with a given program in real time.
- Specific impact analysis on Twitter: Through the analysis of a specified search word, it is possible to study the impact of the sector related to the word and get customized conclusions to the purpose of the analysis, which is the basic functionality of our program.
- Temporal comparison of the impact on Twitter: For example you can take as word search a brand or personal identifier of a company or organization, and make sequential analysis at different times. These moments may coincide with the introduction of changes in the broadcast mode in Twitter allowing measure the effectiveness of the changes taken. A clear example of this utility is to see how many followers have the majority profiles that tweet about the personal branding, so that you can make a measurement of the scope of the brand (if a user with many followers tweeting about something, that information reaches more people than if you do a user with few followers).

CONCLUSIONS AND FUTURE WORKS

Conclusions

After finishing the project, we can say that the main objective of the project (develop a tool to analyze Twitter profiles through the Evolving Systems) has been fulfilled. The most decisive part has certainly been the system design. Through the design decisions taken and the technical adjustment in programming issues of them, it has managed to achieve a useful tool that has allowed us a satisfactory analysis.

The project documentation has been a trivial task, because its development cannot take place without written support of each of the phases made.

Apart from that, from an external point of view, the results of the tests have shown that the domain data of Twitter profiles does not follow a pattern such identification as to perform a general and powerful classification. Individual clustering, meanwhile, does allow for a quick picture of the profiles analyzed, but is also influenced by the nature of the domain of the data.

The result of the project is good, because it has achieved the objectives set and display performance data from Twitter and Evolving Systems.

Future works

From the current development it is possible to add multiple functionalities to the tool for ease of use. Most are technical aspects which pursue on the one hand making it versatile use, and on the other to seize this versatility to improve the quality of the analysis obtained.

Additional features that may arise from the current state of the tool are, among others, the following:

- To make easier the choice of geo-locations configurable in the interface itself, avoiding the direct modification of the file that contains them.
- Balancing automatically searches on each geo-location to get to get a number of concrete extracted profiles. It will consists mainly in performing an amount of searches proportional to the publication density of each geo-location, considering the selected search word and the publication density of other geo-locations.
- Parameterize constants (or at least their initial values) of the algorithms used. Currently these values are set inside the code of the algorithm itself, so by the tool we cannot know how it would affect modifying them to the outcome of the analysis.

In the field of experimentation, you can also make an empirical comparison between eClass0 and other incremental classifiers. The goal is to measure the effectiveness of our classifier respect to others, using the same data for comparison. In this way we know how good it would be our classifier without considering the influence of the correlation between domain data.

Finally, one of the features that emerged at the beginning of the project and that has not come to implement is the inclusion of a profile entered by the user in the models generated with the algorithms of the tool. For this, it is necessary to collect and pre-process real-time data of profile introduced in the same way as the rest of analyzed data, and compare it with the results of RDE, eClustering and eClass0. Thus, the user can see the profile entered affinity with the current analysis. The main incentive for the development of this functionality is the interaction with the user, giving the tool a friendlier look.

Índice

Tabla de Ilustraciones

1. INTRODUCCIÓN

1.1. Objetivo

El principal objetivo de nuestro proyecto es **desarrollar una herramienta que permita analizar perfiles de Twitter con algoritmos basados en Sistemas Auto-Adaptativos en tiempo real**. Por lo tanto nuestro estudio persigue comprobar el resultado de combinar ambos factores, atendiendo a la posible existencia de patrones representativos en los datos analizados en tiempo real.

Para poder desarrollar el trabajo será necesario un estudio previo tanto del funcionamiento de los Sistemas Auto-Adaptativos (y particularmente los algoritmos que vamos a utilizar) así como la extracción de los perfiles de Twitter en tiempo real. Una parte crucial será diseñar el sistema según los requisitos propios de ambos entornos. Los tipos de análisis que se buscan principalmente son:

- Análisis de la densidad del conjunto vigente de datos, es decir, comprobar la relación entre ellos.
- Clusterización de los datos analizados, buscando un modelo representativo de los perfiles existentes en Twitter.
- Clasificación de perfiles según sus atributos.

Una vez que se haya conseguido finalizar la herramienta, se procederá a realizar varios análisis para comprobar la efectividad de la misma aplicada a los datos utilizados. Por lo tanto no se pretende obtener buenos resultados de análisis, sino el desarrollo de la herramienta para poder evaluar fácilmente dichos resultados.

1.2. Motivación

Hoy en día es muy grande la influencia que tienen las redes sociales en cualquier sector ya sea empresarial, de marca o simplemente divulgativo. Partiendo de la frase “Si no estás en Internet, no existes”, es evidente que una buena forma de “existir” para cualquier empresa u organización es la gestión de perfiles en redes sociales, ya que es una manera directa de llegar a las personas. Según estudios recientes, Twitter se sitúa como la tercera red social más utilizada a nivel mundial, solamente por detrás de Facebook y Youtube. Se estima que actualmente hay 500 millones de cuentas creadas, de las cuales hay 266 millones de usuarios activos en Twitter [1], por lo que los resultados desprendidos de nuestro programa están avalados por el gran impacto de la red social que toma como base.

Ilustración 1 [2]

Sin embargo, no es suficiente con poseer un perfil en una o varias redes sociales, sino que lo que realmente importa es conocer el funcionamiento de las mismas para poder sacar provecho. Gracias a éste conocimiento será posible decidir qué publicar, cómo y cuándo para alcanzar el mayor impacto posible. Aquí es donde entra en juego nuestra herramienta, pues permite analizar la tipología de perfiles de Twitter de un sector determinado según una palabra de búsqueda (que se pueda relacionar directamente con una temática).

Por lo tanto, las situaciones en las que nuestra herramienta puede resultar de utilidad son muy variadas. Una funcionalidad interesante será la de focalizar unas determinadas acciones a un grupo de usuarios reducido, maximizando el impacto y optimizando el número de publicaciones y tiempo empleado en la difusión. A continuación se exponen algunos ejemplos de uso:

- Campañas publicitarias: Cualquier empresa u organización que esté interesada en promover una campaña publicitaria podrá especificar un sector de usuarios marcados como objetivo.
- Difusión de eventos de televisión: Hoy en día cualquier programa de televisión ofrece un hashtag mediante el que interactuar en las redes sociales. Gracias a nuestra herramienta, se podrá analizar en tiempo real cuales son los tipos mayoritarios de usuarios que interactúan con un programa determinado.
- Análisis específico de impacto en Twitter: Por medio del análisis de una palabra de búsqueda especificada, es posible estudiar el impacto del sector relacionado con dicha palabra y llegar a conclusiones personalizadas según el objetivo de dicho análisis, que es la funcionalidad básica de nuestro programa.
- Comparación temporal de la repercusión en Twitter: Por ejemplo se puede tomar como palabra de búsqueda una marca o identificador personal de una empresa u

organización, y realizar análisis secuenciales en diferentes momentos de tiempo. Dichos momentos pueden coincidir con la implantación de cambios en el modo de difusión en Twitter lo que permitirá medir la efectividad de las medidas adoptadas. Un ejemplo claro de esta utilidad es observar cuántos seguidores tienen los perfiles mayoritarios que twittean sobre la marca personal, de tal manera que se puede hacer una medición del alcance de la marca (si un usuario con muchos seguidores twittea sobre algo, esa información llega a más personas que si lo hace un usuario con pocos seguidores).

2. ESTADO DEL ARTE

Para desarrollar nuestro trabajo es importante enmarcar nuestra herramienta dentro del contexto actual de sus áreas relacionadas. En nuestro caso existen dos ámbitos con los que el programa está intrínsecamente relacionado y que determinan su utilidad. Por una parte el análisis genérico de datos procedentes de redes sociales y por otra el uso de algoritmos Auto-Adaptativos (*Evolving Systems*), haciendo especial hincapié en los algoritmos concretos que implementa nuestro programa.

2.1. Análisis de datos de redes sociales

Hoy en día las redes sociales suponen el reflejo de la vida diaria de muchas personas. Actúan como diarios on-line, de los cuales se pueden recopilar datos de forma masiva. La tendencia de analizar datos para obtener un fin (económico, orientativo, académico, sociológico…) es una práctica cada vez mayor. Incluso las propias redes sociales ofrecen un API con el que poder sacar partido a todos los datos de los que son propietarias.

Por otro lado, si analizamos los servicios que nos ofrece una red social como usuarios, podemos observar que detrás de muchos de ellos se encuentra un análisis de datos exhaustivo. Por ejemplo, cada vez que se nos muestran sugerencias (de amistad, de temas interesantes, anuncios,…) es porque el sistema que estamos utilizando ha analizado nuestro patrón de comportamiento y sabe que hay mucha probabilidad de que esas sugerencias sean efectivas (en lugar de asignarlas al azar).

Ya que es muy elevado el uso social que se hace de estas plataformas, es habitual encontrar perfiles que son utilizados masivamente como spam. Esto supone un problema para los propietarios de las redes sociales, pues se empeora la satisfacción del usuario y puede conllevar la disminución de usuarios activos. Existen estudios detallados sobre cómo detectar el spam, por ejemplo en [3] se analizan algoritmos para detectar perfiles o comportamientos anómalos dentro de Twitter.

2.1.1. Líneas de investigación relacionadas

En estudios como [4] se trata el análisis de un número masivo de perfiles, teniendo en cuenta determinados atributos para hacer una clusterización. El objetivo principal es la detección de cuentas que actúen como spam.

Se utilizan dos algoritmos para la clusterización, y se comparan los resultados empíricos tanto individuales como en conjunto de ambos. La idea básica es generar un clúster mayoritario de perfiles normales, considerándose cualquier ejemplo lejano a dicho clúster como *outlier* y por tanto marcado como spam.

Derivado de estudios como el anterior, también surgen nuevas situaciones que requieren la adaptación de los sistemas empleados. Es el caso de [5]. En este artículo se realiza un estudio sobre las cuentas de Twitter que actúan como spam. Los *spammers* son conscientes de las técnicas actuales que se llevan a cabo para su detección, por lo que ellos mismos tienen que desarrollar mecanismos de evasión de las mismas para poder continuar con su actividad. El objetivo de este estudio es analizar dichas técnicas de evasión, así como encontrar nuevas formas de detección del spam en Twitter.

Todo el estudio se basa en el análisis empírico de los algoritmos utilizados, comparando cuáles ofrecen mayor robustez y la capacidad de los mismos para combatir las nuevas técnicas de evasión emergentes.

2.1.2. Aplicaciones similares

- Twitalyzer [6]: Analiza y contrasta muchos datos, pero basándose en un solo perfil. También exporta a .csv, lo que facilita la extracción de reportes o informes de los análisis que se efectúen sobre los datos. La diferencia de nuestro proyecto es que esta herramienta web analiza y es capaz de medir métricas más complejas que un simple sondeo de los datos del perfil. Es capaz de medir el impacto de una cuenta concreta, además de encontrar relaciones de actividad entrante y saliente, etc. En definitiva también hace un análisis del patrón de uso de la cuenta, pero a un nivel superior al que vamos a desarrollar nosotros. El principal inconveniente es que es una herramienta de pago.

Ilustración 2

- Xefer [7]: Al igual que en el caso anterior, únicamente analiza una cuenta individualmente. Se limita a analizar tweets según hora y día de la semana, detallando la actividad de la cuenta en: tweets, retweets y menciones.

Ilustración 3

Debido a la popularidad de la red social Twitter y al libre acceso que otorga ésta sobre sus datos (siempre que el usuario decida hacer público su perfil), en Internet podemos encontrar infinidad de herramientas relacionadas con el análisis y extracción de conocimiento relacionadas con la información obtenida de *hashtags* y tendencias. Estas herramientas hacen uso de técnicas de *Text Mining*, por lo que no se pueden comparar con nuestro proyecto. Además también hay un gran número de herramientas que basan su utilidad en la recopilación de la información procedente de los seguidores, seguidos, favoritos, retweets,… en definitiva, la red que está más relacionada con una cuenta concreta.

Sin embargo, en nuestro caso no nos interesa dicha información. Lo que queremos analizar es el patrón de uso de la red social, independientemente de la temática que se trate o sobre lo que se escriban tweets. Es por este motivo por el que la selección de referencias en Internet, que estén relacionadas con este hecho, ha sido una labor más compleja que buscar simplemente herramientas que analicen los *hashtags*.

2.2. Sistemas Auto-Adaptativos (*Evolving Systems*)

En los últimos años, muchos sistemas deben tratar y analizar grandes cantidades de datos. Por este motivo, los retos a los que hacer frente en el procesado de información, y en particular en su clasificación, están relacionados con: 1) Necesidad de tratar y analizar grandes cantidades de datos. 2) Procesar flujos de datos on-line y/o en tiempo real.

Uno de los paradigmas que trata de crear sistemas que aborden estos dos aspectos son los denominados (en inglés) *Evolving Systems* [8] (en castellano, también son conocidos con el término Sistemas Auto-Adaptativos). A lo largo de este trabajo utilizaremos ambos términos indistintamente.

Así, los *Evolving Systems* son sistemas que pueden desarrollarse y aprender en base a los datos recibidos y en los que tanto sus parámetros como su estructura se adaptan al entorno teniendo en cuenta cómo éste varía. Debido a que utilizan algoritmos recursivos, estos sistemas son capaces de recibir, tratar y analizar gran cantidad de datos en tiempo real.

Dada la gran cantidad de datos que muchos sistemas pueden recibir en tiempo real, estos sistemas no necesitan almacenar todos los datos que se reciben, ya que determinadas variables almacenan cierta información que permite omitir gran cantidad de datos sin disminuir la calidad del sistema. Otra característica importante de los *Evolving Systems* es su posible interpretación; es decir, los modelos que obtienen pueden ser interpretados por los humanos.

Además, Los *Evolving Systems* están basados en sistemas borrosos. Así, El marco estructural utilizado por los SAA son los sistemas borrosos basados en reglas del tipo *Takagi-Sugeno*. Estos sistemas pueden ser descritos por reglas de la forma *si-entonces* con una estructura de antecedente y consecuente definida siguiente modo:

$Regla_i$ = *SI* (x_1 *es similar a* x_{i1}) *Y* (x_2 *es similar a* x_{i2}) *Y …*
… Y (x_n *es similar a* x_{in}) *ENTONCES* ($y_i = a_{i0} + a_{i1}x_1 + … + a_{in}x_n$)

donde:

- $Regla_i$ es una de las R reglas borrosas (i=1,2,...,R) almacenadas en la base de reglas.
- x_j (j=1,2,...,n) representan las variables de entrada.
- y_i denota la salida de la regla borrosa i.
- x_{ij} representa el prototipo de la regla borrosa i.
- a_{ij} es el parámetro j de la regla borrosa i.

Teniendo en cuenta que estos sistemas son recursivos y que están explicados en detalle en [9], en los siguientes sub-apartados comentaremos brevemente los tres algoritmos basados en *Evolving Sytems* que utilizaremos en este proyecto: *RDE, eClustering y eClass0*.

2.2.1. RDE

El algoritmo RDE (*Recursive Density Estimation*) sirve para detectar ejemplos cuyos valores difieran notablemente del resto de ejemplos dentro de un conjunto de datos. El procedimiento que sigue es analizar secuencialmente cada uno de los ejemplos, estableciendo una densidad o distribución típica entre sus atributos. Al ser recursivo, ese dato significativo se verá influido por cada ejemplo analizado, de tal manera que se marcará una densidad media propia del conjunto de datos. Además, como ya hemos comentado, dado que no es necesario almacenar los datos, el algoritmo es muy rápido y puede implementarse en tiempo real (independientemente de la cantidad de datos recibida).

2.2.2. eClustering

El algoritmo eClustering es un clusterizador de datos, es decir, un mecanismo de agrupación según los valores más comunes de cada atributo. En nuestro caso será muy útil para conocer los tipos de perfil más comunes en los datos analizados. Cómo realizar estos clústeres se basa también en las fórmulas recursivas de RDE y su funcionamiento está detallado en [10] .

2.2.3. eClass0

El algoritmo eClass0 es un clasificador basado en clústers. Dentro del ámbito de nuestro programa será útil para evaluar la correlación que puede existir o no entre los atributos, para determinar el valor de uno de ellos a partir de la combinación de valores del resto.

El funcionamiento del mismo se basa en el algoritmo anterior eClustering. Divide el conjunto de datos según la clase a la que pertenezcan y realiza un agrupamiento con todos los ejemplos pertenecientes a una misma clase. Para el proceso de testeo, evalúa la similitud de un ejemplo dado con los clústers de cada clase, y se le asigna por tanto la clase más similar. Toda la formulación de este algoritmo está en detalle en [9].

3. ANÁLISIS DEL SISTEMA

La fase de análisis del sistema recoge los requisitos que van a permitir realizar un diseño adecuado de nuestra herramienta, según las necesidades que pretende satisfacer. El caso que abordamos no ofrece grandes complicaciones respecto a la dinámica de uso y consecución de objetivos, por lo que los requisitos que se pueden extraer son sencillos e intuitivos.

3.1. Requisitos del sistema

Los requisitos del sistema pueden dividirse en: Funcionales y No Funcionales, según el área en el que estén encuadrados. A continuación detallamos cada uno de ellos:

3.1.1. Requisitos Funcionales

Los requisitos funcionales son aquellos que están relacionados con la funcionalidad explícita que pretende ser satisfecha al usuario. Es decir, albergan la funcionalidad básica del sistema.

Código	RF-001
Descripción	La herramienta deberá permitir al usuario la extracción de datos de perfiles de Twitter masivamente en tiempo real.

Código	RF-002
Descripción	La herramienta deberá permitir al usuario decidir si iniciar un nuevo análisis o continuar uno ya existente.

Código	RF-003
Descripción	La herramienta deberá permitir al usuario decidir el nombre del fichero así como la palabra de búsqueda de datos al iniciar un nuevo análisis.

Código	RF-004
Descripción	La herramienta deberá permitir al usuario finalizar el análisis, además de poder iniciar otro distinto (nuevo o existente) sin salir del programa.

Código	RF-005
Descripción	La herramienta deberá permitir al usuario analizar los datos extraídos en la sesión vigente con el algoritmo RDE, y ofrecer los resultados tanto de forma gráfica como textual.

Código	RF-006
Descripción	La herramienta deberá permitir al usuario fijar un valor de sigma para el análisis con el algoritmo RDE.

Código	RF-007
Descripción	La herramienta deberá permitir al usuario analizar los datos extraídos en la sesión vigente con el algoritmo eClustering, y ofrecer los resultados tanto de forma gráfica como textual.

Código	RF-008
Descripción	La herramienta deberá permitir al usuario analizar los datos extraídos en la sesión vigente con el algoritmo eClass0, y ofrecer los resultados tanto de forma gráfica como textual.

Código	RF-009
Descripción	La herramienta deberá permitir al usuario fijar un porcentaje de entrenamiento y seleccionar el atributo clasificatorio para cada análisis realizado con el algoritmo eClass0.

3.1.2. Requisitos No Funcionales

Los requisitos no funcionales recogen los requerimientos del sistema que a priori el cliente no conoce, pero que son necesarios para el completo diseño del sistema. Son restricciones que afectan más al ámbito técnico del proyecto y deben ser tenidos en cuenta durante todo su desarrollo, de forma intrínseca a los requisitos funcionales.

Código	RN-001
Descripción	La herramienta deberá ofrecer un tiempo de respuesta para el análisis lo más reducido posible

Código	RN-002
Descripción	Cada análisis estará identificado con un nombre compuesto por el período de tiempo y la palabra de búsqueda escogidos por el usuario.

Código	RN-003
Descripción	La herramienta deberá almacenar la información necesaria para mantener entre distintas sesiones los modelos generados con los algoritmos eClustering y eClass0, siempre que se trate del mismo análisis.

Código	RN-004
Descripción	La información ofrecida visualmente por el algoritmo eClass0 deberá refrescarse cada vez que se modifique el atributo clasificador seleccionado, incluso aunque no se llegue a ejecutar el análisis.

Código	RN-005
Descripción	El sistema deberá ofrecer al usuario una retroalimentación visual del progreso de los procesos que se encuentren en ejecución (extracción de datos, eClustering y eClass0).

Código	RN-006
Descripción	El sistema deberá almacenar la información relativa al proceso de extracción de datos de manera textual y permanente, haciendo uso de un fichero de Log independiente para cada análisis.

3.2. Casos de uso

Los casos de uso del programa se pueden resumen en este diagrama. Por una parte, se muestra la interacción esperada entre las ventanas de inicio y general, y por otra se detallan las funcionalidades disponibles en la ventana general. En todo momento el usuario debe tener la libertad de navegar entre ambas ventanas según su voluntad, así como abandonar el sistema cuando lo desee.

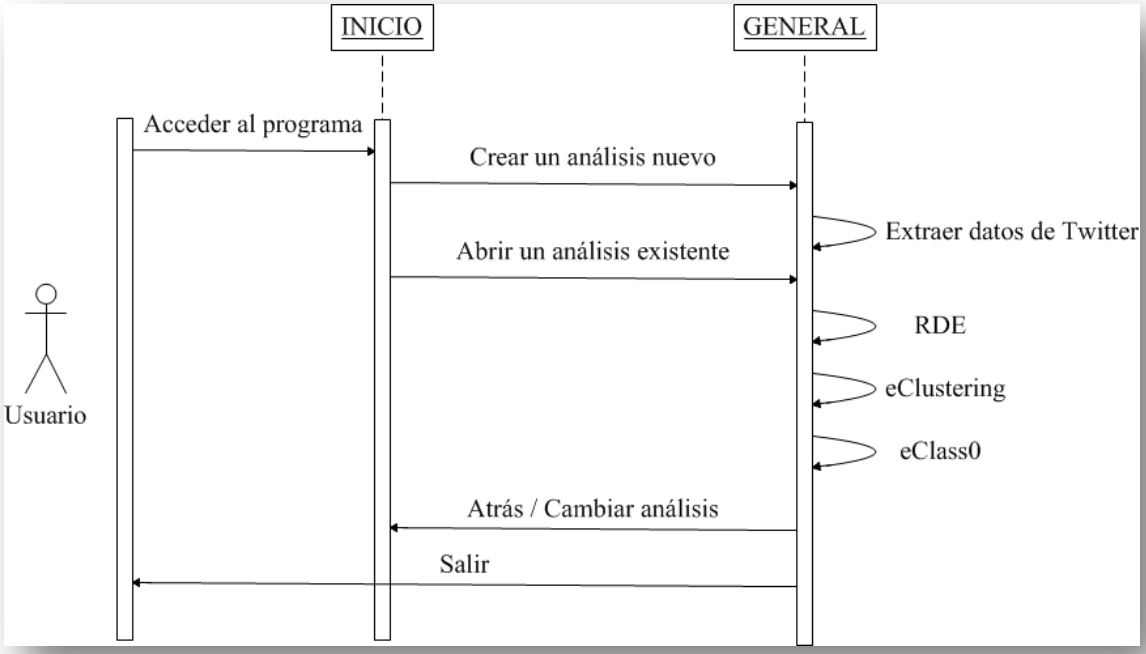


Ilustración 4

4. DISEÑO DEL SISTEMA

A continuación se detalla el proceso seguido para el diseño de nuestro programa. En cada una de las fases se expone su confección completa sin profundizar en los detalles técnicos propios de la programación utilizada. Las decisiones de diseño en cada parte están debidamente justificadas y adaptadas al entorno de nuestro sistema.

4.1. Extracción de datos

4.1.1. Elección de datos a analizar

La elección de los datos a analizar se basa en la búsqueda de patrones de usuario, en nuestro caso provenientes de redes sociales. La plataforma elegida para ello ha sido Twitter, por los siguientes motivos:

- API pública y amplia. Permite obtener gran cantidad de atributos de un perfil.
- Perfiles públicos (sin necesidad de autorización por parte de cada usuario para ceder sus datos al uso de la aplicación).
- Sencillez de implementación a través de la librería de Java: twitter4j [11], facilitando la extracción de datos en tiempo de ejecución.

Un detalle importante a tener en cuenta es que, dado que se hace uso de un servicio externo a nuestro desarrollo (API Twitter v1.1 [12]), nuestro programa estará expuesto a cualquier posible error procedente de los servidores de Twitter que nos impidan su uso. Sin embargo, son situaciones a las que no podemos hacer frente de una forma genérica (la variedad de errores devueltos por el servicio de Twitter es lo suficientemente amplia como para no poder tratar cada caso individualmente).

Respecto al límite de uso del API establecido por Twitter, existe una cota superior de búsquedas permitidas por una cuenta de desarrollador (180 búsquedas cada intervalo de 15 minutos). A priori puede suponer un problema para nuestro proyecto, ya que puede limitar la fluidez de uso del programa al no poder extraer datos para el análisis. Más adelante estudiaremos cómo afecta realmente este hecho en nuestro desarrollo en función del modo en que enfoquemos el uso de la API.

Además, también hay que contar con una limitación derivada del API de Twitter, y es que por cada búsqueda efectuada sólo permite obtener 100 tweets. Si redujéramos la extracción de nuestro programa a una sola búsqueda, dispondríamos de muy pocos datos para analizar. Por otro lado, si realizamos varias búsquedas sin parámetro secuencialmente, es muy probable que obtengamos muchos tweets repetidos. El API de Twitter nos ofrece la posibilidad de realizar búsquedas de tweets por geolocalización y amplitud de búsqueda, por lo que es una funcionalidad muy acertada para utilizar en nuestro caso. La solución más adecuada que se ha adoptado ha sido la búsqueda en varias geolocalizaciones para obtener un mayor número de tweets simultáneamente.

Retomando el objetivo del proyecto, la idea principal es extraer toda la información posible de perfiles de Twitter en tiempo real. ¿Qué perfiles? Lo que buscamos es una estandarización del perfil como tal, por lo que hay que eliminar cualquier tipo de sesgo que conlleve la extracción de los datos. Para ello se ha decidido elegir ubicaciones mundiales de las que extraer los datos. Por medio de la herramienta web '*tweetping.net*' [13], se obtiene un mapa como el de la siguiente imagen, en el que se muestra la densidad de actividad de Twitter a nivel mundial en tiempo real.

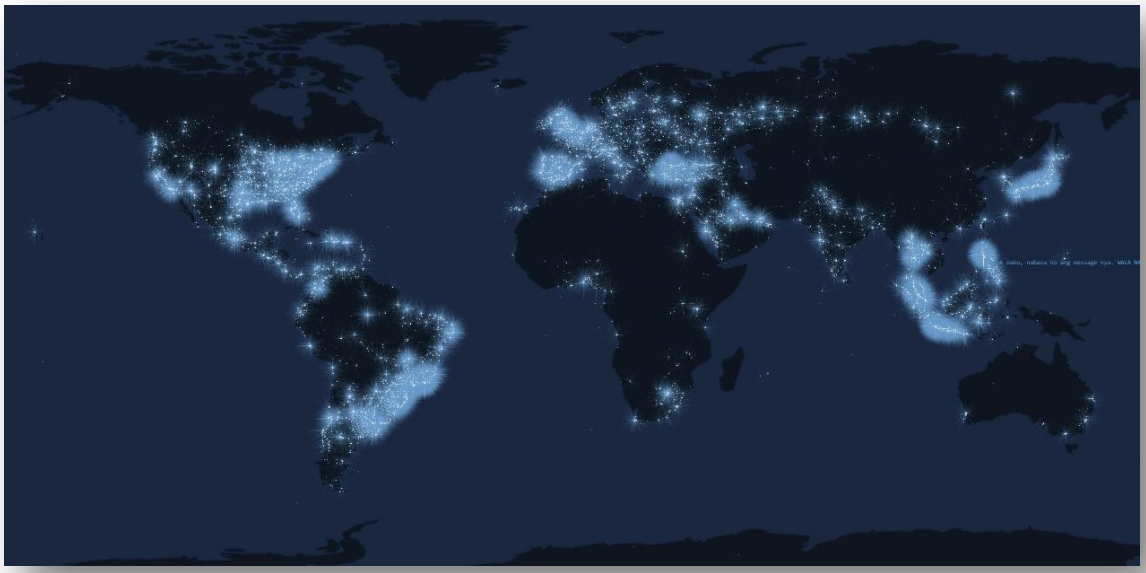


Ilustración 5

La imagen es el resultado de ejecutar la herramienta web entre 15 – 20 minutos. A priori se puede pensar que la búsqueda a nivel mundial en un instante determinado puede verse influida por la diferencia horaria entre las distintas partes del planeta (lo que nos haría replantearnos la filosofía de extracción). Pero analizando la densidad de actividad mundial de Twitter a distintas horas por medio de esta herramienta nos revela que se genera la misma distribución de densidad independientemente de la hora. El factor que tendremos que tener en cuenta por tanto para el rastreo de datos será la geolocalización y más concretamente aquellas zonas con más densidad de actividad.

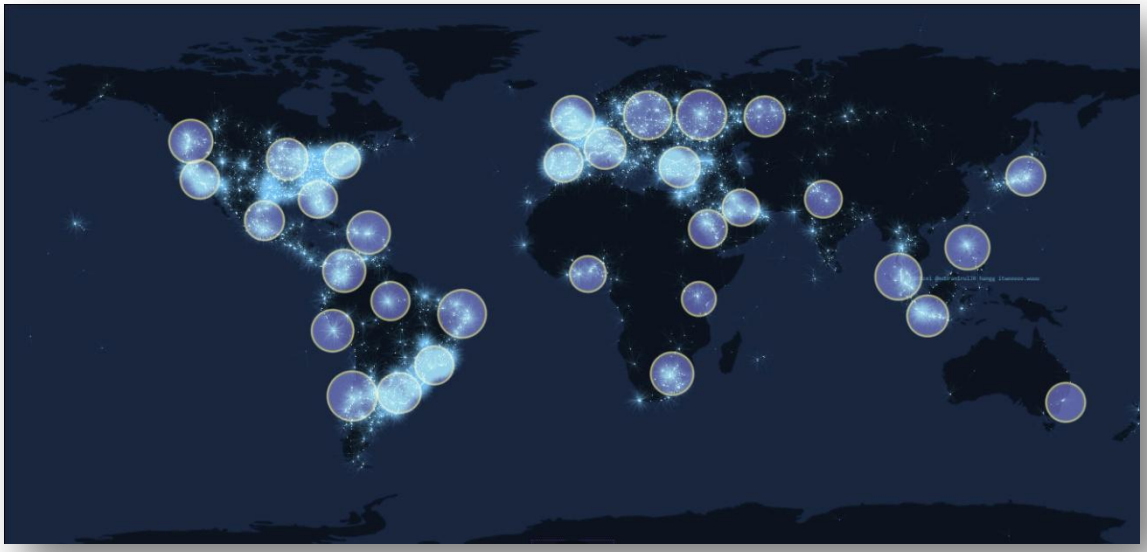


Ilustración 6

En esta imagen se muestran los puntos de búsqueda elegidos. Se han asignado de forma que abarquen todo el espacio de actividad mayoritaria, y proporcionalmente a la densidad de dicha actividad. La amplitud de los círculos que se aprecia en la imagen se corresponde con el radio real utilizado (600 Km).

Tanto el número de ubicaciones como el radio de búsqueda de cada una de ellas se ha elegido buscando un equilibrio entre cantidad de datos para analizar y rapidez de ejecución. En total se han tomado 32 geolocalizaciones como base, pudiéndose añadir o eliminar fácilmente modificando el fichero que las contiene.

Volviendo a la restricción indicada anteriormente, del límite de búsquedas del API de Twitter, procedemos a estudiar cómo influye en el desarrollo de nuestro programa:

- Dado que disponemos de 32 geolocalizaciones y en cada extracción de datos realizamos una búsqueda por geolocalización, tenemos 32 búsquedas por cada ejecución de análisis.
- El límite de Twitter es de 180 búsquedas cada 15 minutos, por lo que podemos realizar 5 extracciones sin sobrepasar el límite (32*5 = 160; 160 < 180) cada 15 minutos.
- Esto significa que, en promedio, si realizamos una extracción cada 3 minutos, podremos utilizar nuestro programa sin ningún problema.

4.1.2. Datos analizados

Datos de perfil de Twitter seleccionados para analizar:

Tipo de dato	Función	Descripción del dato devuelto
int	getStatusesCount()	Número de tweets escritos por el usuario
int	getFavouritesCount()	Número de tweets favoritos del usuario
int	getFollowersCount()	Número de seguidores del usuario
int	getFriendsCount()	Número de usuarios seguidos por el usuario
java.util.Date	getCreatedAt()	Fecha de creación de la cuenta
java.lang.String	getProfileBackgroundColor()	Color de fondo del perfil
java.lang.String	getProfileLinkColor()	Color de enlaces
java.lang.String	getProfileSidebarBorderColor()	Color del borde de la barra lateral
java.lang.String	getProfileSidebarFillColor()	Color de relleno de la barra lateral
java.lang.String	getProfileTextColor()	Color del texto del perfil
java.lang.String	getTimeZone()	Zona horaria del usuario
java.lang.String	getLang()	Lenguaje preferido del usuario
int	getListedCount()	Listas públicas que tiene el usuario. Valor -1 si no existen listas
java.lang.String	getURL()	URL del usuario
boolean	isDefaultProfile()	Indica si el usuario no ha modificado el tema o fondo del perfil
boolean	isProfileBackgroundTiled()	Indica si el fondo del perfil se muestra en mosaico
boolean	isProfileUseBackgroundImage()	Indica si existe una imagen de fondo en la cabecera del perfil

Otros datos no han sido escogidos por no ser considerados relevantes, ni válidos para poder realizar un análisis con ellos, bien porque son valores identificativos o extraños (campos de ID, nombre de usuario, descripción, localidad, …), o bien porque todos los perfiles tendrían el mismo valor para dicho campo (por ejemplo el dato booleano que indica si la cuenta es pública, o el que indica si está activada la geolocalización; todas las cuentas recopiladas son públicas y tienen la geolocalización activada porque de lo contrario no podríamos acceder a esos datos).

4.1.3. Preprocesado de datos

Primero debemos ser conscientes de la necesidad de un intermediario entre el algoritmo de análisis que vamos a utilizar, y los datos presentes en Internet. Para ello se ha implementado un programa en Java que realiza exactamente esa función: extrae datos de Twitter, los adapta a los requerimientos de los algoritmos, y los exporta a un fichero .csv que es lo que utilizan dichos algoritmos. Para ello se ha utilizado el entorno de desarrollo Eclipse [14].

Como ya se ha indicado anteriormente, se utiliza una librería que implementa el API de Twitter. Para poder extraer datos es necesario el registro previo como desarrollador de

Twitter. Es un registro gratuito, se basa en una cuenta de Twitter existente y se obtienen las claves necesarias para habilitar el uso de la API (validación que se hace al principio del programa). Por medio del uso de sus funciones, el método para extraer nuestros datos es el siguiente:

1. Se realiza una búsqueda de tweets por geolocalización. Hay que especificar coordenadas geográficas, amplitud o radio de búsqueda y parámetro de búsqueda (en nuestro caso utilizamos la cadena de texto recibida por parámetro).
2. De todos los tweets obtenidos nos interesa tomar el autor del tweet (perfil de usuario).
3. Para cada usuario obtenemos los datos del perfil que nos interesan.
4. Adaptamos (pre-procesamos) dichos datos a los requerimientos de cada campo para el análisis por los algoritmos que vamos a utilizar.

En el proceso de extracción de los datos, hay que adaptar dichos datos a nuestro uso particular, ya que tal y como los devuelve el API con el que trabajamos hace falta realizar un preprocesado. Los algoritmos que vamos a utilizar para realizar el análisis están preparados para tratar con datos numéricos. Por lo tanto tenemos que realizar las siguientes adaptaciones con los datos de los que disponemos:

1. Atributos Enumerados: Por una parte, hay determinados campos de texto que nos interesa analizar. Dado que son valores que se repiten, lo más adecuado es gestionarlos como un tipo enumerado. De esta forma, al algoritmo de análisis le van a llegar valores numéricos que se corresponden con valores textuales. La relación entre valor y número se almacenará en ficheros de tipo .csv (adecuados para almacenar pares de valores, sin ningún tipo de estructuración compleja). Dichos ficheros se utilizarán en la lectura de datos de Twitter, para leer el valor numérico asociado a una determinada cadena de texto, y en caso de que la nueva cadena no exista, añadirse a la lista enumerada.

   Se ha utilizado este tipo de gestión para los campos de: idioma y zona horaria. Para cada uno de ellos se crean ficheros .csv independientes, es decir, un tipo de enumerado diferente.

   Para facilitar esta implementación a nivel de código se ha creado una clase auxiliar de enumerado con dos atributos: índice y valor.

2. URL personal: Por otro lado, la URL que puede tener o no tener especificada el perfil, es información que sí que podemos tratar. Se ha adaptado a un valor booleano, de tal manera que es 1 si contiene URL y 0 en caso contrario. Así descartamos la información que no es de utilidad (contenido de la URL).

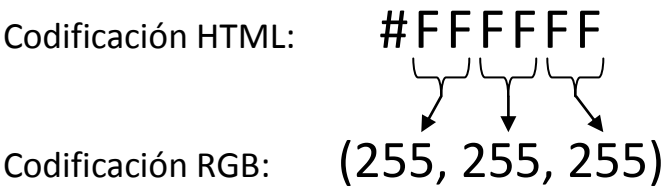
3. Fecha de creación (de la cuenta): Además, la fecha de creación del perfil también ha habido que tratarla y convertirla a un valor numérico único. Los valores que realmente pueden ser útiles son el mes y año, ya que el día y la hora con minutos y segundos es poco relevante cuando se trata de un momento puntual en un amplio rango de tiempo. Se podría haber almacenado cada uno de esos dos valores de forma independiente, pero al realizar el análisis se perdería la relación que hay entre ellos (es estrictamente necesario que mes y año sean una sola entidad para garantizar la relación cronológica).

Una solución válida ha sido por tanto crear un valor numérico que concatene año y mes. Así se relaciona con un orden numérico según el orden cronológico y las comparaciones entre perfiles se acercan más a la comparación temporal.

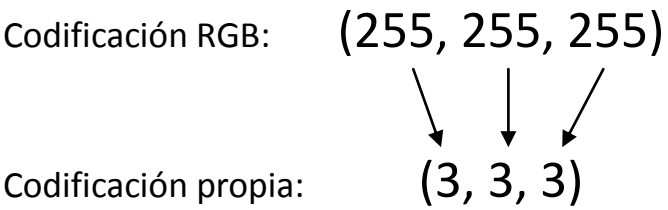
Adicionalmente se ha reducido el año a sus dos últimas cifras. Twitter se creó en 2006, por lo que todas las cuentas estarán creadas en un año 20…, y podemos descartar esas dos primeras cifras. Además, la relación cronológica se mantiene porque no se produce un cambio de siglo y no puede darse una sucesión 99->00->01 (1999->2000->2001) que rompa el orden numérico natural. De esta manera reducimos el rango numérico del atributo fecha, sin perder semántica en el valor del mismo, lo que reducirá la complejidad de cálculos que involucren este campo.

4. Colores: Hay varios atributos que representan colores. Dichos colores son extraídos de Twitter en formato HTML (6 dígitos hexadecimales). En un principio se optó por crear un tipo enumerado al igual que con los campos de idioma y zona horaria. Sin embargo, tras sucesivas ejecuciones de prueba, el número de colores que aparecen nuevos en el fichero enumerado es extenso. Por lo tanto perdemos la ventaja que queremos conseguir con la adaptación a enumerado (reducir valores basándose en la repetición). Debemos por tanto modificar la codificación en sí.

   Analizando la codificación de color HTML, sabemos que se compone de seis dígitos, que se agrupan de dos en dos para indicar la intensidad de color de los colores rojo, verde y azul, en éste orden. Por ejemplo, el color blanco se correspondería con la siguiente interpretación:



   Nuestro objetivo es reducir la cantidad de dígitos necesarios para representarlo, así como el rango de valores total (facilitando la relación entre datos en los ejemplos para simplificar el análisis por parte del algoritmo). Esto conlleva que en nuestra nueva codificación, un mismo valor agrupa varios valores de la codificación original. Para hacer una transformación eficaz, utilizaremos una codificación de tres dígitos decimales para representar los colores. Siguiendo la interpretación anterior (HTML -> RGB), transformaremos el valor RGB a una cifra en base 4. Dicha transformación se realiza aplicando al número original un factor de conversión: 3/255.



Gracias a este proceso, conseguimos dividir el espectro total de valores de color que ofrece la codificación HTML en tan sólo 64 valores (4^3; base 4, dígitos 3). Ésta es la secuencia de colores contenidos en dicho rango:

Ilustración 7

Observamos que hay poca correlación entre los tonos sucesivos según la numeración establecida. Sin embargo, aunque se ha intentado fijar el orden de codificación entre colores similares, ha sido imposible conseguirlo. La relación entre los tonos del espectro completo no sigue una similitud sucesiva lineal, sino que hay mínimo tres relaciones inmediatas para cada color (variando cada una de las tres componentes RGB). Hay que tener en cuenta que además del propio tono cromático, también existe la influencia del blanco y el negro (los cuales afectan a un componente RGB concreto por medio de combinación de valores del resto de componentes RGB) lo que dificulta establecer una sucesión lineal apta para que nuestra codificación se corresponda con una relación de colores adecuada.

Esto conlleva que a la hora de que analicemos los datos con los algoritmos, la relación entre valores sucesivos numéricos no se corresponda del todo con colores relacionados. Aún así, sigue siendo válido para agrupar tonos similares con la propia conversión de codificación.

Otro aspecto común en el preprocesado de cualquier conjunto de datos es la normalización. En nuestro caso, la implementación de los algoritmos que vamos a utilizar ya incorporan la normalización de los datos (configurable mediante parámetro). Más adelante se detallará el uso de dicha normalización en el contexto individual de cada algoritmo, ya que en unas ocasiones nos va a interesar normalizar los datos, pero en otras no resultará tan útil.

Una vez que ya hemos realizado el preprocesado de datos oportuno, vamos a ver el resto de detalles del programa para la extracción de datos.

4.1.4. Información de datos extraídos

Hay un factor en el entorno de nuestra herramienta que juega en contra. El hecho de que los datos que usa el algoritmo para entrenarse se obtengan en tiempo de ejecución es un proceso que tiene un coste en tiempo importante. Surge la necesidad, por tanto, de retroalimentar al usuario de la actividad del programa durante todo el tiempo de espera para que sea consciente de que el programa está trabajando.

Se ha añadido al programa una interfaz simple, de una sola ventana a modo informativo. La información que se muestra a lo largo de la ejecución es la tarea básica que se está realizando (leyendo ubicaciones, extrayendo datos de cada ubicación…) así como los perfiles obtenidos de cada ubicación y una barra de progreso que hace el trabajo del programa más visual. Por último se informa del total de perfiles extraídos y guardados, además de la fecha y hora del tweet más antiguo extraído.

Dentro del funcionamiento general de la plataforma, este programa se insertará como funcionalidad de la interfaz que contiene el algoritmo en sí, de tal modo que la ventana de extracción de datos aparecerá de manera emergente respecto al programa hecho en MATLAB para el resto del proyecto.

Uno de los datos que se le muestran al usuario es la fecha y hora del tweet más antiguo extraído. Este dato es necesario para que el usuario sea consciente de la densidad de tweets publicados a lo largo del tiempo, según el parámetro de búsqueda elegido. Por ejemplo, si se realiza una búsqueda por la cadena vacía (se obtiene cualquier tweet en la geolocalización especificada), la hora y fecha del más antiguo es muy próxima. Sin embargo, si se busca por una palabra poco frecuente o en un idioma poco popular (por ejemplo, la palabra “frijoles”), se pueden observar dos fenómenos: hay en muchas de las geolocalizaciones elegidas en las que no se encontrarán tweets con ese parámetro (muy probablemente por el idioma), y además la fecha y hora del tweet más antiguo puede variar en días anteriores a la fecha actual. Por lo tanto, surge el problema de que en sucesivas extracciones de datos en el mismo análisis con la misma cadena de búsqueda, se encuentren tweets repetidos ya que no se han publicado nuevos, o simplemente el número total de tweets antiguos y nuevos no supera los 100 tweets por búsqueda en cada geolocalización.

Podría haberse controlado la extracción selectiva de tweets según la fecha y hora de publicación posteriores al más reciente extraído anteriormente, lo que podría generar búsquedas sin resultados más recientes que una búsqueda anterior. Enfocándolo a un uso real dentro de la dinámica de funcionamiento de nuestro programa, se ha optado por notificar al usuario de la fecha y hora del tweet más antiguo extraído, quedando también dicha información reflejada en el fichero de log asociado al análisis (contiene tanto la fecha de extracción como la del tweet más antiguo para poder ser comparadas).

Con esta información, es el propio usuario el que decide en tiempo real si entrenar los modelos acumulados de cada algoritmo o no, sabiendo cuándo realizó el análisis anterior. Además, puede que quien realice el análisis prefiera entrenar los modelos acumulados con datos repetidos (que son una representación real de los datos que se pretenden analizar) a no entrenar nada.

Si analizamos la información almacenada en el fichero de log, también podemos realizar un análisis más exhaustivo de cómo de repetidos estarían los datos en sucesivas extracciones. La procedencia de los datos viene de tantas búsquedas distintas como geolocalizaciones hayamos especificado. Por lo tanto, una determinada palabra puede ser poco frecuente en una geolocalización pero muy popular en otra. Esto significa que los datos extraídos de la geolocalización con más frecuencia de publicación serían menos propensos a ser repetidos. En el fichero de *log* esto puede verse atendiendo al número de perfiles obtenidos de cada geolocalización. Si el número es 100 (número de perfiles máximos), entonces es muy probable

que se estén publicando tweets constantemente con la palabra de búsqueda elegida. Este hecho puede interpretarse de la siguiente forma: en una extracción posterior puede que los datos de geolocalizaciones con menos de 100 perfiles sí que estén repetidos, pero los datos de aquellas con 100 perfiles almacenados en la ejecución anterior es más probable que se trate de datos nuevos para entrenar nuestros algoritmos. De esta manera, se puede obtener más variedad de datos "nuevos" aunque el tweet más antiguo que figure tenga igual fecha que en la extracción de datos anterior.

4.1.5. Ficheros E/S

El programa Java se va a comunicar con el resto del proyecto con el fichero *.csv* que genera. Ya hemos detallado anteriormente las particularidades de los datos que se guardan en él. Su estructura general consiste en tuplas de datos: cada tupla corresponde a un usuario de Twitter y cada columna (atributos) a un campo diferente. Cada campo es un dato del perfil que guardamos para que el algoritmo lo analice en la siguiente fase del proyecto.

Adicionalmente, se utilizan los siguientes ficheros auxiliares:

- Un fichero de Log, que recoge la actividad de extracción de datos a lo largo de la ejecución.
- Un fichero de geolocalizaciones. Se utiliza por facilitar la adición o eliminación de lugares de los que extraer tweets. Es un fichero .csv que almacena nombres de ubicaciones y coordenadas. Para añadir o eliminar geolocalizaciones sólo habría que añadir o eliminar filas. El programa Java se encarga de leer el fichero y realizar la extracción de datos según las localizaciones que existen en tal fichero.
- Los ficheros pertenecientes a los tipos de enumerados especificados anteriormente.

4.2. Programa en MATLAB

Los algoritmos que vamos a utilizar para el análisis de datos están implementados como librerías de MATLAB. Esto es debido a que basan su funcionamiento en el cálculo matricial de datos, y para este fin, MATLAB [15] es una de las opciones más adecuadas. Gracias a la potencia de dicho lenguaje, el cálculo es rápido en comparación con otros lenguajes de programación que son menos efectivos en el cálculo matricial.

Por lo tanto, para nuestro proyecto implementaremos una interfaz en MATLAB que nos permita utilizar las librerías de los algoritmos de análisis, e integrar el programa Java de extracción de datos.

El patrón de uso de nuestro programa será el de analizar en tiempo real perfiles de Twitter según los algoritmos disponibles. De tal manera que las funcionalidades principales del programa son:

- Elegir un archivo existente para continuar el análisis o crear uno nuevo.
- Extraer en tiempo real información sobre perfiles de Twitter.
- Analizar los datos con el algoritmo RDE.
- Analizar los datos con el algoritmo eClustering.
- Analizar los datos con el algoritmo eClass0.

Para el funcionamiento del programa se utilizan los siguientes archivos:

- Un archivo .mat que almacenará el conocimiento acumulado a lo largo de las sucesivas ejecuciones de un análisis. Almacenará información propia del análisis, como los clústers generados o la palabra clave de búsqueda. Se nombrará dicho archivo con la palabra de búsqueda propia del análisis y el nombre introducido por el usuario para dicho análisis.
- Las librerías y archivos propios del programa MATLAB, necesarios para su funcionamiento.
- El ejecutable .jar que contiene el programa Java para obtener tweets en tiempo real.
- Los archivos asociados al funcionamiento de dicho programa Java. Ya que utilizamos algoritmos Auto-Adaptativos, la filosofía que debemos seguir no es almacenar todos los datos extraídos en todas las ejecuciones realizadas, sino únicamente el conocimiento obtenido de ellos (almacenado en el archivo *.mat*). Por este motivo, el archivo data.csv que genera el programa Java con los datos extraídos se elimina al finalizar cada ejecución.

Cada análisis se identificará por el periodo en que se realiza y la palabra de búsqueda de los tweets. Ambos se fijan en la primera ejecución, y no podrán cambiarse en las siguientes ejecuciones. Es decir, si queremos cambiar cualquiera de ellos, se debe crear un análisis nuevo.

El uso general de nuestra herramienta consistirá por tanto en:

- Analizar individualmente los perfiles asociados a un determinado texto de búsqueda.
- Comparar los análisis generados a partir de diferentes palabras de búsqueda.
- Ambas funciones se realizan con sucesivas ejecuciones (independientes para cada análisis) dentro del periodo de tiempo convenido por el usuario. Se entrenan los modelos generados con datos extraídos en tiempo real, por lo que cuantos más datos se extraigan, los modelos estarán más entrenados y el análisis será de mayor calidad.

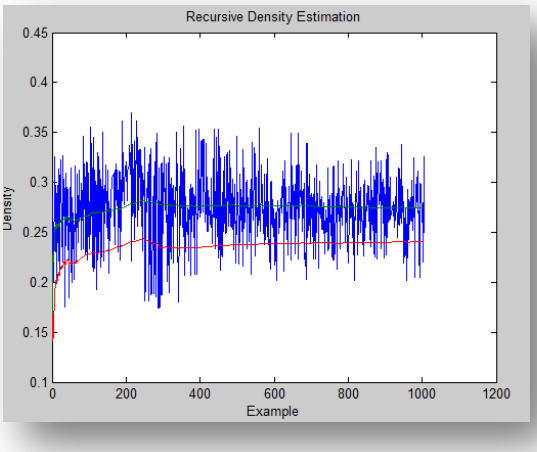
4.3. Tratamiento de datos

4.3.1. RDE

Existe un parámetro 'sigma' que servirá como umbral para marcar los ejemplos considerados como *outliers*. Se puede interpretar como un margen más o menos amplio en la consideración de dichos ejemplos extraños. Un sigma bajo dará como resultado un mayor número de *outliers*, mientras que un valor bajo de este parámetro reducirá dicha cantidad hasta el punto de no considerar ningún ejemplo como *outlier*. Debido a que para cada conjunto de datos la densidad generada puede ser muy variable, en nuestro programa se ofrece como un parámetro editable, lo que facilita el análisis de los datos por parte del usuario. De esta manera, se puede variar el número de *outliers* encontrados dependiendo del objetivo del análisis, así como observar si el conjunto de datos tiene mucha variedad interna (densidad alta) o por el contrario los datos son muy similares entre sí (densidad baja).

Otro factor a tener en cuenta es la normalización previa de los datos antes de ejecutar el algoritmo. Para percibir la diferencia entre normalizar o no los datos, hemos realizado una prueba con dos ejecuciones similares, únicamente variando la normalización de los datos y manteniendo sigma=1. El gráfico de resultados obtenido en cada caso ha sido el siguiente:

Normalizado:



Sin normalizar:

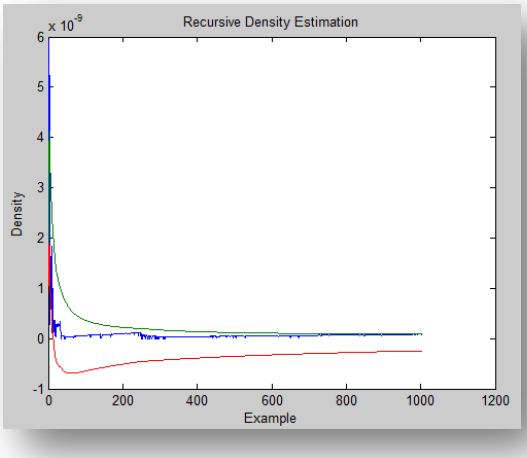


Ilustración 4

Apreciamos que para un análisis visual de los datos (que es nuestro objetivo), la ejecución normalizada permite observar más exhaustivamente el valor individual para cada ejemplo. Esto es debido a que por medio de la normalización, todos los datos van a estar dentro de una escala reducida y próxima, con valores muy próximos a cero.

Por el contrario, si nos fijamos en la gráfica sin normalizar, la diferencia entre valores de todos los ejemplos no permite ver claramente cada dato individual.

Por lo tanto se ha decidido fijar el parámetro de normalización para que el análisis sea siempre más intuitivo y claro.

4.3.2. eClustering

Los metadatos necesarios para trabajar con los clústeres generados son almacenados para poder continuar con la fase de entrenamiento a lo largo de las sucesivas ejecuciones. De esta manera conservamos el conocimiento adquirido en cada una de ellas. Para la carga de información proveniente de una ejecución anterior se utiliza una variable auxiliar a modo de indicador, de tal manera que sólo se cargará información anterior si ya se ha entrenado previamente un modelo de datos.

El número de clústers utilizados no es fijo. Es el propio algoritmo el que, tras el análisis de cada ejemplo individual, decide si asociarlo a un clúster existente o crear uno nuevo. Por este motivo, el número de clústers podrá aumentar siempre que el algoritmo lo considere oportuno según los datos analizados, pero nunca podrá disminuir. Por lo tanto podemos considerar que el algoritmo es relativamente autónomo y se adapta a los datos por sí mismo. Esto es una ventaja respecto a otros sistemas de clusterización a los que hay que establecer un número de clústers fijo antes de realizar el entrenamiento.

Una particularidad que tiene el uso de este algoritmo en nuestro análisis es la normalización de los datos. Ya hemos visto que en el caso del algoritmo RDE, la normalización suponía una mejora en el análisis de los resultados por parte de un humano. Sin embargo, para el algoritmo eClustering no es tan adecuado.

La ventaja que tiene la no normalización de datos en este proceso, es que se puede hacer una lectura casi inmediata de los valores de cada clúster (Tweets, Seguidores,…). Pero si aplicamos la normalización, todos los valores estarán entre 0 y 1, por lo que no podremos saber cuál es el valor real de cada campo debido a que cada atributo tiene un rango de valores distinto. Por ejemplo, no podemos comparar un valor normalizado de 1 entre el número de tweets y un atributo booleano, ya que ese valor 1, en el caso del número de tweets puede representar un valor real de miles de tweets y el valor booleano, simplemente 1. Además, al tratarse de un algoritmo incremental que entrenamos en sucesivas ejecuciones, puede que al normalizar en una primera ejecución se tomen como mínimos y máximos unos valores que no coinciden con los datos extraídos en la siguiente ejecución y la normalización no sería válida.

Por este motivo se ha decidido no normalizar los datos para poder realizar estas apreciaciones a la hora de hacer el análisis.

4.3.3. eClass0

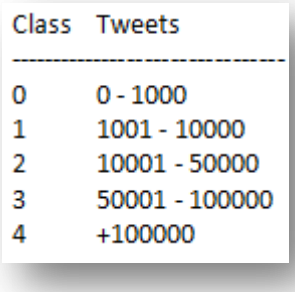
Para poder desarrollar correctamente este algoritmo ha sido necesario realizar un tratamiento de datos especial, el cual se realiza en memoria en tiempo de ejecución sobre una copia de los datos generales. Es necesario hacerlo de esta manera para no alterar los datos que usan el resto de algoritmos. Al igual que en el caso de eClustering, se ha decidido no llevar a cabo la normalización de datos en este algoritmo, para facilitar su interpretación directa.

Técnicamente hay que entender que por cada atributo clasificador seleccionado se generará un modelo clasificatorio diferente, por lo que hay que tratar individualmente la clasificación realizada por cada uno de los atributos (hay que realizar una copia y modificación de los datos generales en cada caso).

Por una parte hay que modificar la matriz de datos general, y reordenar sus columnas poniendo al final la correspondiente al atributo clasificatorio seleccionado. Éste paso se realiza porque el algoritmo está preparado para clasificar según la última columna de la matriz que reciba por parámetro.

Por otra, si nos fijamos en la variedad de valores que puede tomar cada algoritmo es muy diversa. Hay atributos binarios, con los cuales el clasificador puede trabajar perfectamente. Pero sin embargo hay otros atributos con rangos de valores más amplios, o con características especiales como pueden ser los colores o los campos enumerados. Además se suma el coste en memoria y tiempo que tiene, ya que por cada atributo clasificador y clase hay que realizar un clusterizador independiente. Veamos qué decisión se ha tomado en cada caso:

- Valores numéricos normales (Tweets, Favoritos, Seguidores, Seguidos, Listas): Dado que este tipo de atributos tiene una relación lineal de valores, lo más adecuado es establecer una discretización que reduzca la codificación numérica agrupando rangos determinados. En nuestro caso se ha decidido establecer 5 valores discretizados (de 0 a 4). Los rangos asignados a cada valor dependen de los valores reales que adopta cada atributo, por lo que se ha realizado un estudio de la dimensión y variedad de los mismos y se ha establecido la discretización adecuada a cada caso. Dicha discretización aparece en la interfaz gráfica a modo informativo, dependiendo del atributo clasificador que esté seleccionado en ese momento. Por ejemplo, si está seleccionada la clase Tweets, la información mostrada es la siguiente:



Class	Tweets
0	0 - 1000
1	1001 - 10000
2	10001 - 50000
3	50001 - 100000
4	+100000

Ilustración 5

- Fecha: En este caso se ha decidido no incluir éste atributo como clasificador. Aunque sigue un orden cronológico, no es posible establecer una agrupación discreta de valores sin perder información. Por ejemplo se podría haber agrupado por año, pero perderíamos la información relativa al mes y la clasificación ya no tendría sentido (sería poco precisa).

- Colores: Estos atributos son recibidos por el algoritmo, codificados según el pre-proceso anteriormente realizado. Recordemos que existen un total de 64 valores de colores (codificados), pero aún así es un valor demasiado extenso como para establecer una clasificación sobre él. Respecto a la agrupación de valores, es una tarea compleja que ya se nos presentó a la hora de decidir codificar el valor original

de este campo. No es viable reducir la cantidad de valores sin perder relación entre ellos, por lo que se ha decidido que los atributos de colores tampoco sean atributos clasificadores.

- Enumerados (Zona horaria e Idioma): Esos atributos son mucho más particulares que el resto para la tarea que nos ocupa. Tienen valores numéricos, pero no olvidemos que cada uno de esos valores representa un valor textual, creado en tiempo de ejecución la primera vez que es detectado. Esto significa que no existe ninguna relación real entre los valores numéricos que adopta cada enumerado, por lo que no tiene sentido agrupar valores para una discretización. Por otro lado, tampoco podemos tomar los valores directamente para ser utilizados por el clasificador porque alcanzan valores muy elevados que harían la clasificación una tarea intratable en memoria y tiempo para nuestro programa. Por estos motivos se ha decidido que no sean atributos clasificadores.

- Binarios: Este caso es el más sencillo de todos, pues el rango de valores es muy reducido (0 ó 1) y además los valores aparecen numéricos y ordenados. En este caso, el algoritmo clasificador puede usar éstos valores directamente y no es necesario realizar ninguna adaptación adicional. Por lo tanto estos atributos sí que forman parte de los atributos clasificadores.

4.4. Funcionamiento del sistema

La interfaz gráfica del programa se compone principalmente de dos ventanas: una para seleccionar entre un análisis existente o crear uno nuevo, y la ventana general de uso del programa. Adicionalmente se generan otras ventanas auxiliares, como por ejemplo el explorador de archivos del sistema o gráficos generados por los algoritmos.

4.4.1. Ventana de inicio

La ventana de inicio es la siguiente:

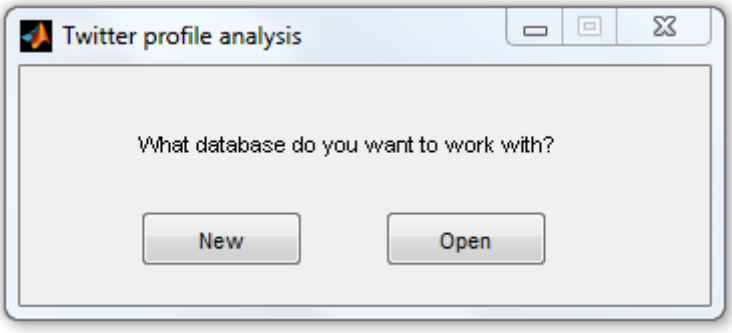


Ilustración 6

Las dos opciones que ofrece son:

- Crear un nuevo análisis. Se requerirá al usuario que introduzca un nombre para dicho análisis, sugiriendo que se utilice un rango de tiempo representativo para el análisis que se quiere realizar (número de semana, mes, año,…).
- Continuar un análisis ya existente. Para ello se abrirá un explorador de archivos locales en el que el usuario deberá seleccionar el archivo del análisis que quiera cargar.

Una vez que el usuario ha seguido uno de los dos caminos y ya está definido el archivo para el análisis, se lanza la ventana general y se carga en el programa dicho archivo.

4.4.2. Ventana general

En la ventana general podemos diferenciar cuatro partes principales, que agrupan el espacio necesario para su uso, individualmente del resto de zonas:

- Zona de selección de entorno y extracción de datos
- Zona del algoritmo RDE
- Zona del algoritmo eClustering
- Zona del algoritmo eClass0

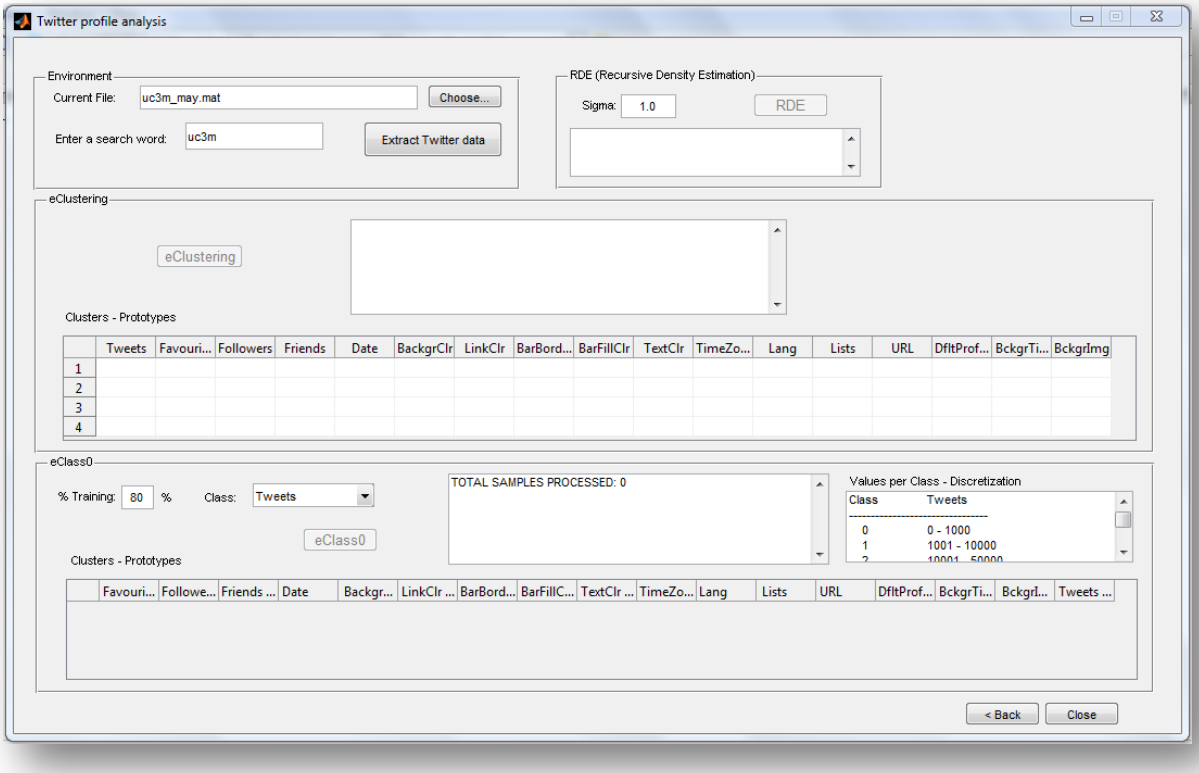


Ilustración 7

Los casos de uso y los flujos de ejecución posibles están limitados con la activación o desactivación de los botones adecuados en cada momento. De esta manera se evitan errores

inesperados o warnings por un mal uso de la herramienta. Por ejemplo, no se puede ejecutar ninguno de los algoritmos antes de extraer datos de Twitter, de igual manera que sólo se pueden extraer datos de Twitter una vez por ejecución, no más. El objetivo de este tipo de restricciones es la consistencia de datos internos del programa, de forma que partiendo de los datos extraídos se puedan ejecutar los algoritmos deseados, entrenando los modelos respectivos individualmente.

En la zona de entorno existe la posibilidad de cambiar el análisis actual por otro ya creado. Si queremos crear uno nuevo, se deberá pulsar el botón "Back" que nos retornará a la ventana de inicio. Además, la barra de búsqueda donde se especifica la cadena de texto por la que se quieren obtener tweets, es editable únicamente en la primera ejecución. El resto de veces que se abra el análisis este campo ya no será modificable ya que la cadena de búsqueda es una característica identificativa de cada análisis.

Por otro lado, para retroalimentar al usuario del progreso de los diversos algoritmos durante su ejecución, se ha implementado una ventana emergente que contiene una barra de progreso dinámica. Con ello conseguimos que, particularmente en ejecuciones más largas, el usuario sea consciente de que el programa está trabajando y a qué ritmo lo hace.

4.5. RDE

4.5.1. Información ofrecida

Un elemento significativo en este algoritmo es el concepto de '*outlier*', es decir, que se sale de la línea o tendencia marcada por el resto de ejemplos anteriores (tomando el orden secuencial de análisis). Por lo tanto, un *outlier* será aquel ejemplo que no siga una distribución de atributos similar a la establecida por el resto de ejemplos.

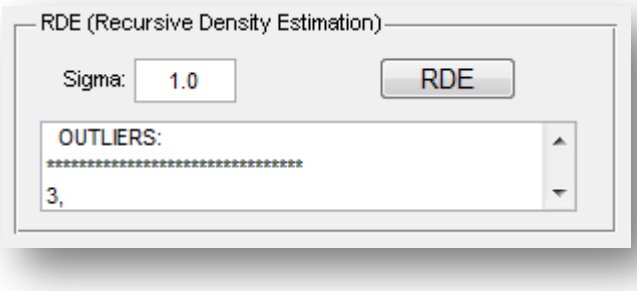


Ilustración 8

Este algoritmo es útil para detectar la presencia de ejemplos extraños en nuestro conjunto de datos. La ejecución del mismo genera un gráfico bidimensional con la densidad de cada ejemplo, lo que ofrece una comprensión visual muy rápida. Además, en el cuadro de texto se

indican los patrones detectados como *outliers*, numerados según su índice en el orden del conjunto de datos vigente.

Es necesario indicar que el conjunto de datos que analiza el algoritmo RDE en nuestro programa es únicamente el obtenido mediante la extracción de datos de Twitter en la vigente ejecución, por lo que no acumula conocimiento entre ejecuciones.

4.6. eClustering

4.6.1. Información ofrecida

Al ejecutar este algoritmo obtendremos la información detallada de los clústers generados:

- Gráfico con los valores de cada atributo para cada clúster.
- Detalle de los valores concretos de todos los atributos de cada clúster.
- Número de ejemplos pertenecientes a cada clúster (valores acumulados de las sucesivas ejecuciones del análisis).

Un ejemplo de gráfico de clústers es el siguiente. Están representados 11 clústers, cada uno con un color. Las diferencias más notables se observan en los atributos con un rango de valores más grandes (situados al principio). Por el contrario, los atributos con valores más bajos, particularmente los booleanos, no se aprecian bien en la gráfica. Para su análisis será preciso acudir al detalle numérico de valores, ofrecido dentro de la interfaz del programa.

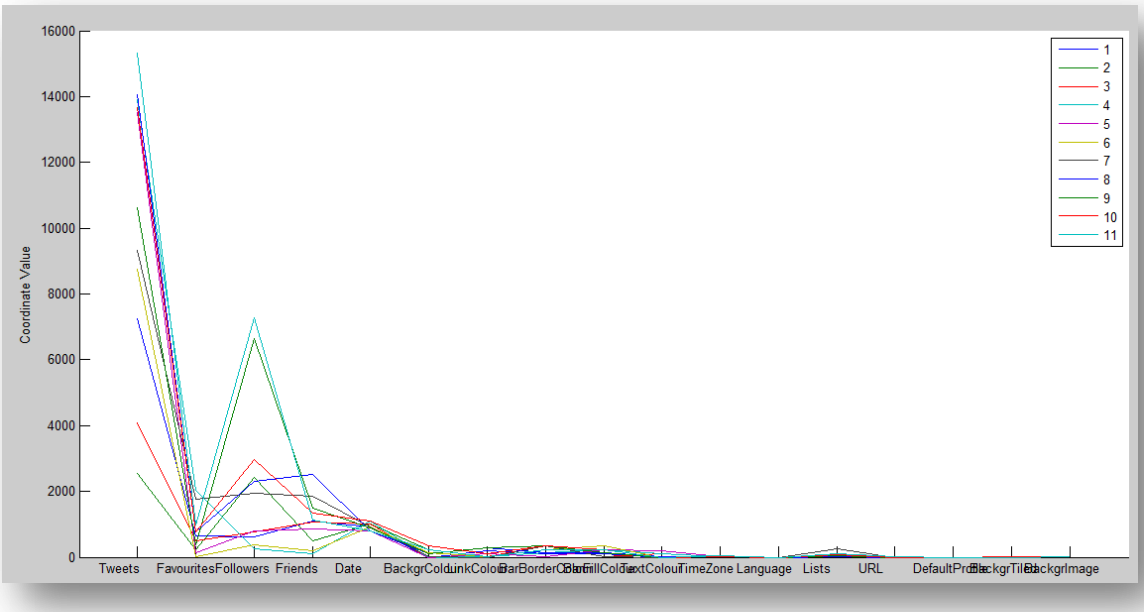


Ilustración 9

La lectura más interesante que podemos realizar aquí es la observación entre los diferentes valores de Tweets, Favoritos, Followers y Friends. Apenas hay coincidencia de valores entre

clústers diferentes, lo que nos indica los rangos de valores que agrupa cada clúster. Además, se aprecia la influencia del resto de atributos en la diferenciación de clústers con valores similares en un atributo concreto, particularmente se ve cómo hay varios clústers con aproximadamente 14000 tweets y que se diferencian por la combinación del resto de atributos.

Además, se muestra al usuario una información textual que sirve para tener una referencia numérica exacta del análisis. La información ofrecida es la siguiente:

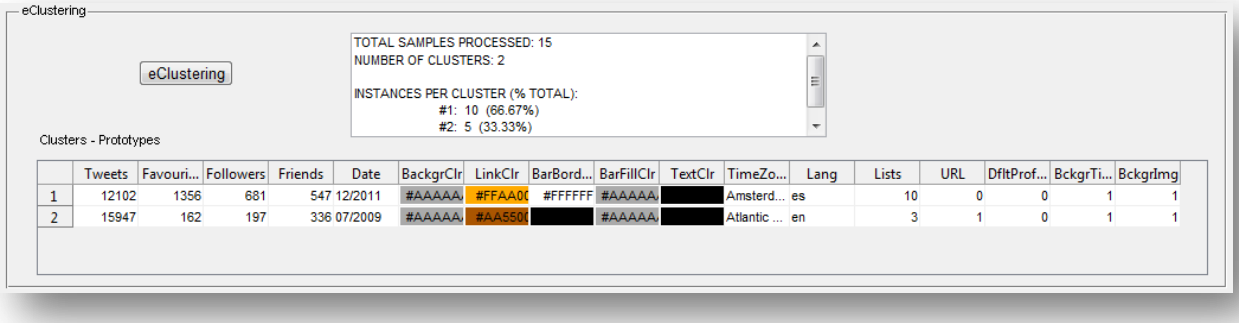


Ilustración 10

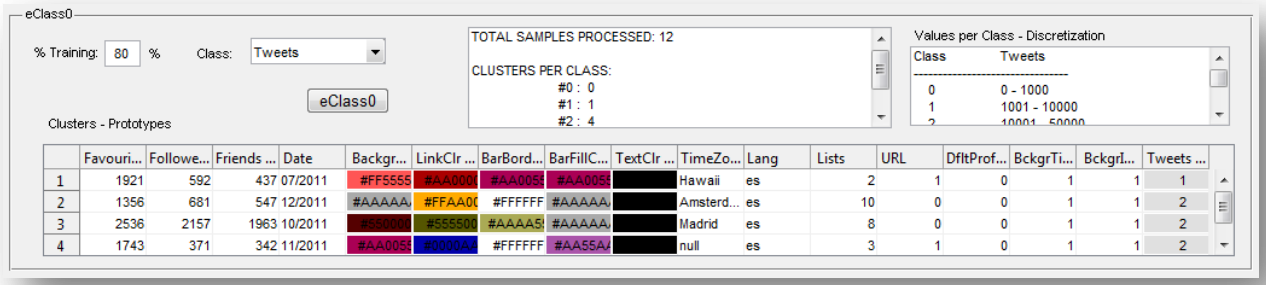
Por una parte podemos ver el número total de ejemplos procesados, es decir, todos aquellos utilizados para generar el modelo de clustering que se ofrece. También se muestra el número de ejemplos que agrupa cada clúster, así como el porcentaje del total que supone. Gracias a estos datos podemos ver qué clústers de los generados son los que más ejemplos agrupan, es decir, los más representativos.

Para completar los resultados del análisis, se detallan los valores de cada atributo para todos los clústeres. Es importante tener en cuenta que los datos extraídos de Twitter están pre-procesados y adaptados al algoritmo, por lo que para mostrar gráficamente la tabla informativa hay que decodificar cada campo (concretamente la fecha, los colores, la zona horaria y el lenguaje). Ésta adaptación permite al usuario poder ver claramente la información analizada, sin necesidad de consultar los ficheros de codificación utilizados internamente.

4.7. eClass0

4.7.1. Información ofrecida

Una vez que sabemos cómo trabaja el algoritmo, veamos qué información nos ofrece. La apariencia gráfica dentro de nuestro programa es la siguiente:



	Favouri...	Followe...	Friends ...	Date	Backgr...	LinkClr ...	BarBord...	BarFillC...	TextClr ...	TimeZo...	Lang	Lists	URL	DfltProf...	BckgrTi...	BckgrI...	Tweets ...
1	1921	592	437	07/2011	#FF5555	#AA0000	#AA0055	#AA0055		Hawaii	es	2	1	0	1	1	1
2	1356	681	547	12/2011	#AAAAAA	#FFAA00	#FFFFFF	#AAAAAA		Amsterd...	es	10	0	0	1	1	2
3	2536	2157	1963	10/2011	#550000	#555500	#AAAA55	#AAAAAA		Madrid	es	8	0	0	1	1	2
4	1743	371	342	11/2011	#AA0055	#0000AA	#FFFFFF	#AA55AA		null	es	3	1	0	1	1	2

Ilustración 8

Podemos observar que hay dos parámetros configurables para realizar el análisis. Por un lado el porcentaje de entrenamiento, que es un parámetro habitual en cualquier sistema clasificador. Lo que hace es dividir el conjunto de datos totales en dos partes, una para entrenar el modelo que genera el algoritmo, y otra para testear dicho modelo. De esta manera se puede ofrecer un resultado para medir la "calidad" de clasificación del modelo generado. El otro parámetro es el atributo por el cual se desea clasificar, es decir, el que se toma como clase.

Los resultados que se muestran una vez llevado a cabo el análisis son independientes según el atributo clasificador seleccionado. Son los siguientes:

- Número total de ejemplos procesados. Es un dato acumulativo que se incrementa cada vez que se entrena el algoritmo.
- Resultados de la fase de test del clasificador. Incluye tasa de acierto y tiempo empleado.
- Matriz de confusión.
- Número de clústers generados para representar cada clase.
- Valores detallados de cada clúster.

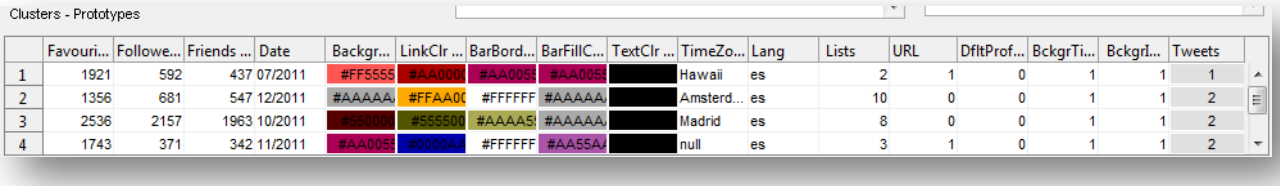
Un ejemplo de resultados es el siguiente. Se ha realizado sobre un conjunto de datos muy reducido, simplemente para mostrar la información necesaria.

```
TOTAL SAMPLES PROCESSED: 24
Class: Tweets - RESULTS:
        Correctly Classified: 1 (rate : 33.3333%)
        Incorrectly Classified: 2
        Total Elapsed Time: 0.72695 seconds

CONFUSION MATRIX:
   |   #0  #1  #2
#0 |    0   0   0
#1 |    0   0   2
#2 |    0   0   1

CLUSTERS PER CLASS:
        #0 : 0
        #1 : 1
        #2 : 5
```

Ilustración 9



	Favouri...	Followe...	Friends ...	Date	Backgr...	LinkClr ...	BarBord...	BarFillC...	TextClr ...	TimeZo...	Lang	Lists	URL	DfltProf...	BckgrTi...	BckgrI...	Tweets
1	1921	592	437	07/2011	#FF5555	#AA0000	#AA0055	#AA0055		Hawaii	es	2	1	0	1	1	1
2	1356	681	547	12/2011	#AAAAA	#FFAA00	#FFFFFF	#AAAAA		Amsterd...	es	10	0	0	1	1	2
3	2536	2157	1963	10/2011	#550000	#555500	#AAAA55	#AAAAA		Madrid	es	8	0	0	1	1	2
4	1743	371	342	11/2011	#AA0055	#0000AA	#FFFFFF	#AA55AA		null	es	3	1	0	1	1	2

Ilustración 10

Observamos cómo la información original se decodifica en los clústers para facilitar su comprensión por parte del usuario. También podemos ver la reorganización de columnas (el atributo Tweets, que es el elegido para realizar la clasificación, se sitúa el último) y el sombreado de la última, que es la clase.

Cada vez que el usuario modifica la selección de clase en el menú desplegable, el resto de campos se actualizan según el atributo seleccionado. Se modifica tanto la información textual de conocimiento acumulado y los clústers correspondientes (en caso de que exista un análisis previo), así como los valores informativos de la discretización concreta del atributo.

4.7.2. Aspectos técnicos

A nivel técnico, es necesario hacer hincapié en una decisión de diseño tomada. Como ya se ha indicado anteriormente, el algoritmo eClass0 hace uso de la clusterización para su funcionamiento. Dentro de cada atributo clasificador, existen tantos clústers como clases posibles haya. El problema surge a la hora de almacenar toda esa información para ser cargada en las próximas ejecuciones.

Ha sido necesario crear las variables necesarias para almacenar la información de todos los clústers que se utilizan. Además, existe una variable a modo de indicador que determina si se ha entrenado un determinado atributo clasificador en cada clase concreta (se ha decidido la creación de estas variables para evitar problemas durante la carga de datos, ya que puede darse el caso de que en un análisis no se creen clústers para todas las clases de un mismo atributo, como es el caso del ejemplo de la imagen anterior). Esto significa que hay que gestionar una gran cantidad de variables, cargándolas en memoria y escribiéndolas en fichero cada vez que sea oportuno.

La primera implementación que se realizó para conseguir la persistencia de conocimiento entre sesiones se basaba en cargar en memoria sólo las variables necesarias en cada iteración del algoritmo, dependiendo del atributo clasificador y de la clase entrenada en ese momento. Así mismo, dichas variables se guardarían actualizadas en el fichero en cada iteración.

El gran problema que surge a partir de este planteamiento es el coste en tiempo que conlleva. Al ejecutar el algoritmo se obtenían tiempos del orden de 2 a 3 minutos, tiempo excesivo considerando que en pruebas simples antes de implantar estos mecanismos de almacenamiento se conseguían ejecuciones más ligeras. Se analizó el código implementado y el mayor coste en tiempo detectado provenía de la constante lectura/escritura en el fichero en todas las iteraciones del algoritmo.

Para solventarlo no quedó otra alternativa que cargar todas las variables al comienzo del mismo, y guardarlas en el fichero al final de su ejecución. La mejora fue muy significativa, ya que se pasó de los anteriores 2-3 minutos a sólo 20-25 segundos por cada ejecución del algoritmo.

Sin embargo, de esta mejora se deriva una reflexión mucho más técnica. La solución adoptada supone acceder a todas las variables necesarias desde dos funciones diferentes, por una parte, en la función general del algoritmo para cargar/salvar los datos del fichero, y por otra, en la función específica que se realiza en cada iteración del algoritmo para modificar estas variables. Dado que utilizamos MATLAB, esto supone que debemos hacer todas estas variables globales, residiendo en memoria todas ellas, se usen o no. Se trata de aproximadamente 300 variables globales entre valores finitos y matrices de datos, lo que nos hace plantearnos el consumo en memoria que puede llegar a tener. Para calcularlo, podemos observar cuál es el tamaño del fichero de datos donde se almacena toda la información de forma persistente, ya que todas las variables que deben estar en memoria están contenidas en dicho fichero. Por lo tanto, el tamaño de estas variables en memoria nunca superará el tamaño en disco del fichero. Analizando varios de los ficheros generados durante el desarrollo del programa, ninguno de ellos supera los 200 Kb, incluso conteniendo todos los datos posibles que genera nuestro programa.

Podemos concluir por tanto que merece la pena implantar esta solución en nuestro proyecto, ya que el coste en memoria no es significativo, y la mejora ganada en tiempo de ejecución es muy notable.

5. EXPERIMENTACIÓN

En esta fase del proyecto se persiguen dos objetivos principales. Por una parte, dar una visión global del uso de nuestra herramienta, mostrando un ejemplo de análisis de datos y las conclusiones que se pueden deducir de ello. Por otra parte, se realizan pruebas más específicas a fin de obtener conclusiones más claras y bondades o carencias del programa en situaciones de análisis específicas.

5.1. Demostración de uso e interpretación

Para llevar a cabo una demostración del uso genérico de nuestro programa vamos a efectuar un análisis de datos según la palabra de búsqueda "music". A continuación, detallamos el proceso seguido, indicando cómo se realizaría un análisis desde cero.

5.1.1. Elección del entorno y extracción de datos

Cuando ejecutamos el programa, nos aparece la ventana de inicio, donde debemos seleccionar si crear un análisis nuevo o continuar con uno ya existente.

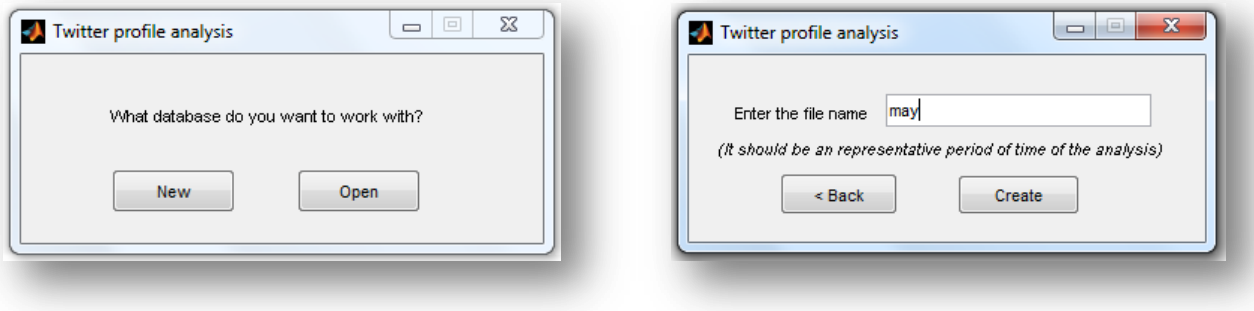


Ilustración 11

Seleccionamos "New" para crear un análisis nuevo. Posteriormente, introducimos un nombre para el fichero de análisis, que sea una unidad de tiempo representativa del mismo. En el caso que nos ocupa, introduciremos el nombre del mes en el que se realiza el análisis y pulsamos "*Create*".

Nos aparece la ventana general. Los botones relativos a los algoritmos aparecen desactivados, pues no serán utilizables hasta que se hayan recolectado datos para analizar. En esta fase del proceso debemos trabajar en la sección de entorno.

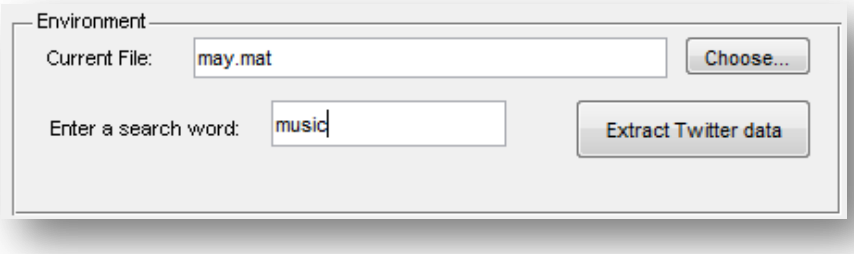


Ilustración 12

Introducimos la palabra de búsqueda en la barra de texto habilitada para ello y hacemos clic en "Extract Twitter data", con lo que activaremos la ejecución del programa en Java para la extracción y preprocesado de datos. En este apartado también podemos cambiar el fichero de análisis si así lo deseamos. Es necesario indicar que en la primera ejecución de un análisis, el nombre de fichero asociado se compone únicamente del período de tiempo especificado en su creación, pero a partir de la primera extracción de datos el nombre del fichero se compondrá de la palabra de búsqueda más el período temporal, de la siguiente forma:

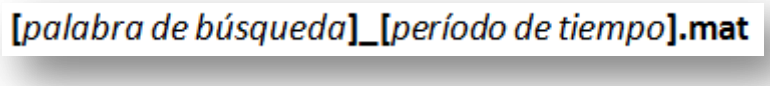


Ilustración 13

Durante la extracción de datos, el programa Java ofrecerá retroalimentación del proceso:

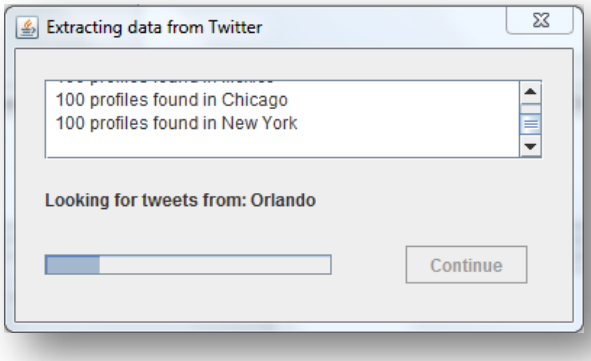


Ilustración 14

Y al finalizar el mismo mostrará por pantalla la siguiente información:

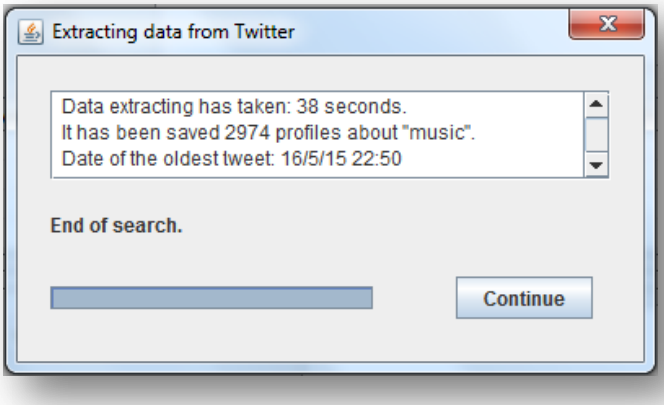


Ilustración 15

Se detalla el tiempo utilizado para la extracción y preprocesado de los datos. Este dato varía en función de la velocidad de la conexión a Internet del dispositivo desde el que se ejecute nuestro programa. Además, se notifica el número total de ejemplos guardados y la fecha y hora del tweet más antiguo del que se ha extraído información. Este dato es útil para conocer la densidad temporal de publicación que tiene una palabra determinada en Twitter. Es importante que el usuario sea consciente de ello si desea realizar más extracciones de datos en el mismo análisis posteriormente, ya que si el rango de tiempo es reducido en relación a la densidad de publicación, obtendrá muchos perfiles de Twitter repetidos porque la nueva búsqueda contendrá datos que ya fueron procesados en el análisis anterior. Como ya se ha detallado en el apartado de "Extracción de datos -> Interfaz gráfica", no se controla dicha extracción selectivamente según la fecha de publicación al dejarse en manos del usuario dicha libertad de selección (eligiendo volver a entrenar con los mismos datos o decidiendo no entrenar si desea datos nuevos).

En nuestro caso, la extracción se ha realizado el día 25/05/2015 a las 19:00, por lo que todos los perfiles encontrados se encuentran en un rango de 9 días. Esto significa que la densidad de publicación es baja, por lo que si se desea entrenar los modelos con datos nuevos, debe hacerse cada 9 días aproximadamente.

Una vez que se ha realizado correctamente la extracción de datos, el botón "Extract Twitter data" se desactiva para evitar inconsistencia de datos, y se activan los botones de cada algoritmo.

5.1.2. RDE

Para el análisis de datos con el algoritmo RDE, deberemos prestar especial atención al parámetro 'sigma'. Según sea el valor de este parámetro, podremos deducir una interpretación de los datos analizados. Hacemos clic sobre el botón "RDE" para ejecutar el algoritmo.

Si mantenemos el valor 1.0 de sigma que el programa ofrece por defecto, obtenemos un número muy elevado de *outliers*:

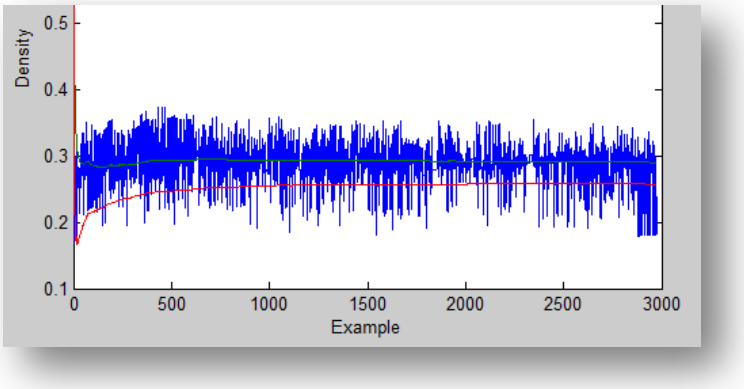


Ilustración 16

Esto significa que los datos están muy dispersos y no guardan una relación muy directa entre sí. Probamos diferentes valores de sigma hasta conseguir un número reducido de *outliers*, a base de prueba y error. Hay que tener en cuenta que los valores de este parámetro suelen oscilar entre 0 y 4 unidades, por lo que probaremos con un número dentro de ese rango. En nuestro caso hemos tomado el valor 3.4 como válido para interpretar nuestros datos:

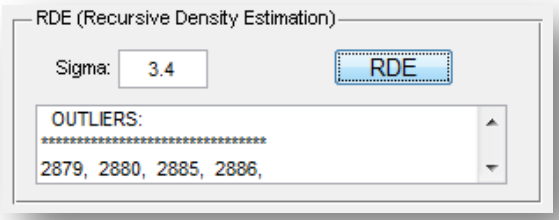


Ilustración 17

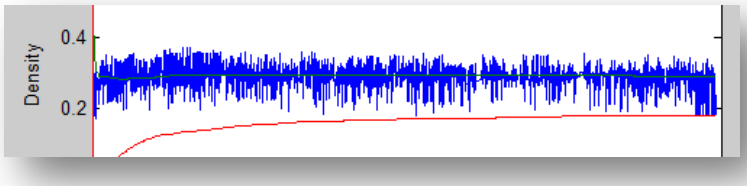


Ilustración 18

Dado que es un valor relativamente alto, podemos determinar que los datos no están muy relacionados entre sí. Si observamos la gráfica, vemos cómo los *outliers* más significativos corresponden con los últimos ejemplos procesados. En general la densidad de los datos se mantiene a lo largo de todo el análisis, por lo que no se puede fijar una relación clara.

5.1.3. eClustering

En el caso del algoritmo eClustering no hay parámetros configurables. Tanto la tasa de aprendizaje como otro tipo de variables que determinan la amplitud y umbral de pertenencia en la creación de los clústers, están fijados en el código del programa. El número de clústers tampoco se especifica ya que, como se ha explicado en apartados anteriores, es una cantidad variable modificada por el propio algoritmo a lo largo de las sucesivas iteraciones.

Los resultados que se ofrecen son, por un lado, un gráfico que representa los clústers visualmente, permitiendo comparar los valores de cada atributo entre clústers diferentes, y por otro, un conjunto de datos textuales que detallan el proceso numéricamente. Además, también se muestra una tabla con los valores de cada atributo para todos los clústeres.

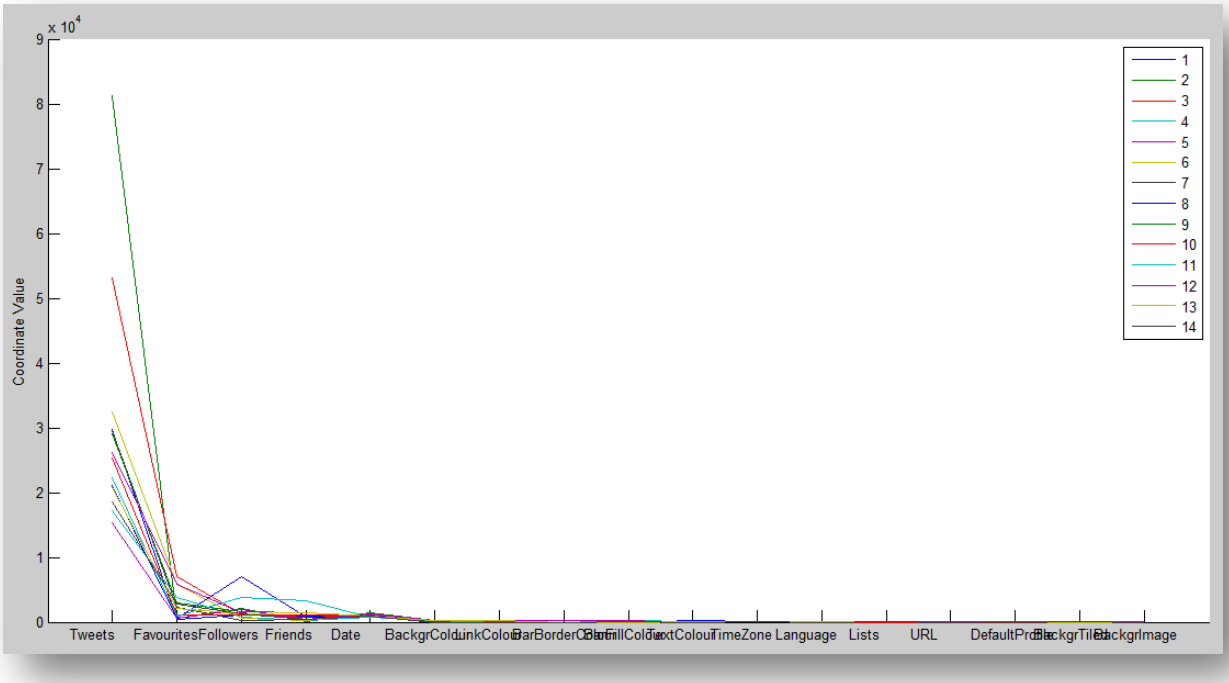
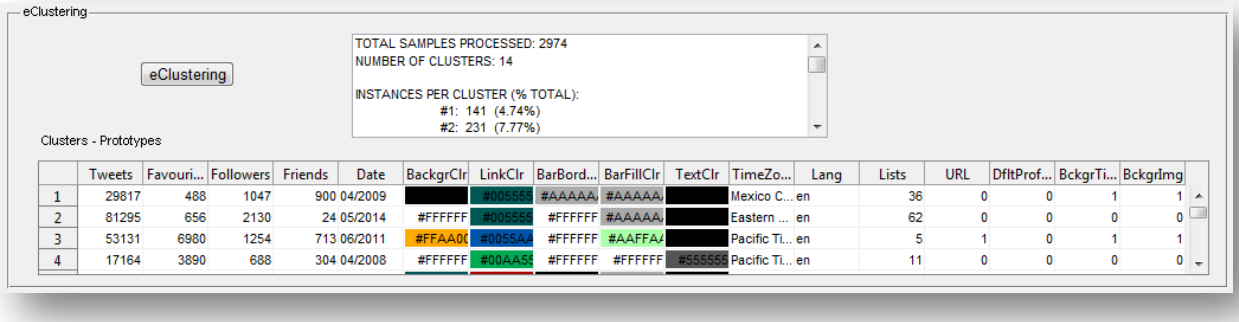


Ilustración 19

En el gráfico observamos la gran diferencia de magnitud que existe entre los valores de los diferentes atributos. El campo más distinguido es el de 'Tweets', ya que ofrece los valores más altos. Es una buena muestra de cómo está repartida la densidad de valores en un atributo concreto. En la imagen podemos apreciar cómo hay un mayor número de clústers en el rango de 15000-30000 y tan sólo dos más en representación de valores más altos. Aparte de comprobar cuál es la tendencia mayoritaria en el número de tweets, a través de esta interpretación se demuestra que el algoritmo mantiene una relación entre el número de clústers y la densidad de ejemplos contenidos en un rango de valores determinado (de no ser así, podría haber empleado un número menor de clústeres para abarcar esa zona de valores).

Por otra parte no hay que perder de vista la influencia de las combinaciones con los demás atributos, ya que para la formación de cada clúster no se valora cada atributo individualmente, sino el conjunto de ellos con la relación establecida por cada ejemplo procesado.

Todos estos valores pueden verse más detallados en la tabla de clústeres, relacionando cada línea en el gráfico con su color según la leyenda del mismo.



	Tweets	Favouri...	Followers	Friends	Date	BackgrClr	LinkClr	BarBord...	BarFillClr	TextClr	TimeZo...	Lang	Lists	URL	DfltProf...	BckgrTi...	BckgrImg
1	29817	488	1047	900	04/2009		#005555	#AAAAA	#AAAAA		Mexico C...	en	36	0	0	1	1
2	81295	656	2130	24	05/2014	#FFFFFF	#005555	#FFFFFF	#AAAAA		Eastern ...	en	62	0	0	0	0
3	53131	6980	1254	713	06/2011	#FFAA00	#0055AA	#FFFFFF	#AAFFAA		Pacific Ti...	en	5	1	0	1	1
4	17164	3890	688	304	04/2008	#FFFFFF	#00AA55	#FFFFFF	#FFFFFF	#555555	Pacific Ti...	en	11	0	0	0	0

Ilustración 20

En la zona de información textual se muestra el número de ejemplos procesados (es un valor acumulativo, incluye todos los ejemplos de todas las veces que se ha entrenado el modelo) y el número de clústers generados. Éste último es un valor que, dado el diseño del algoritmo, puede aumentar en las sucesivas iteraciones del mismo, pero nunca decrecer. Esto es debido a que, una vez procesado un determinado ejemplo, sólo se utiliza para modificar el modelo vigente, pero es desechado y no vuelve a ser tenido en cuenta de forma individual. De ésta manera, la influencia que ejerce un ejemplo en el modelo depende del momento en el que sea procesado, ya que dependiendo del estado del modelo (número de clústers, radio de cada uno…) dicho ejemplo puede crear un nuevo clúster, modificar uno existente, o simplemente ser asignado a uno existente sin modificarlo.

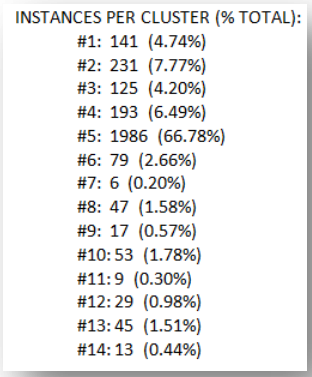


Ilustración 21

Otra información relevante que podemos utilizar el número de ejemplos que agrupa cada clúster, es decir, cuántos ejemplos han influido directamente en la conformación de ese clúster. Como dato adicional, el porcentaje que supone en el total de ejemplos nos facilita diferenciar cuál o cuáles son los clústers más representativos de todos los datos analizados.

En nuestro caso se aprecia claramente un clúster mayoritario, que es el número 5. La diferencia con la población del resto de clústeres es muy grande, aunque no siempre suele ser tan notoria. Lo más interesante de la información detallada en la tabla de clústeres es que

podemos ver qué valores concretos tiene cada clúster, y en este caso nos sirve para determinar qué tipo de perfil es el más popular del análisis en el que nos encontramos. Al ser un porcentaje muy alto de representación de los datos (66.78% de ejemplos agrupados) es un indicador muy significativo.

	Tweets	Favouri...	Followers	Friends	Date	BackgrClr	LinkClr	BarBorderClr	BarFillClr	TextClr	TimeZone	Lang	Lists	URL	DfltProf...	BckgrTi...	BckgrImg
5	15408	510	1969	384	01/2011	#005555	#AA0000		#AAAAAA		Pacific Tim...	en	3	1	0	1	1

Ilustración 22

Por lo tanto el perfil más popular entre los usuarios que publican tweets sobre 'music' tiene las siguientes características:

- 15408 Tweets publicados
- 510 Tweets marcados como favoritos por el usuario
- 1969 Seguidores del perfil
- 384 Perfiles seguidos por el usuario
- Fecha de creación de la cuenta: Enero de 2011 (4 años de antigüedad en la fecha del análisis)
- Colores de fondo, enlace, barra lateral y texto que se aprecian en la imagen
- Zona horaria: Pacific Time (US&Canada)
- Idioma: Inglés (es un dato lógico ya que la palabra de búsqueda está en inglés)
- 3 Listas de perfiles clasificados (nombradas y confeccionadas a gusto del usuario)
- URL personal introducida
- NO conserva el diseño de perfil por defecto
- SÍ tiene el fondo de la cuenta configurado en mosaico
- SÍ tiene el fondo establecido con una imagen personalizada

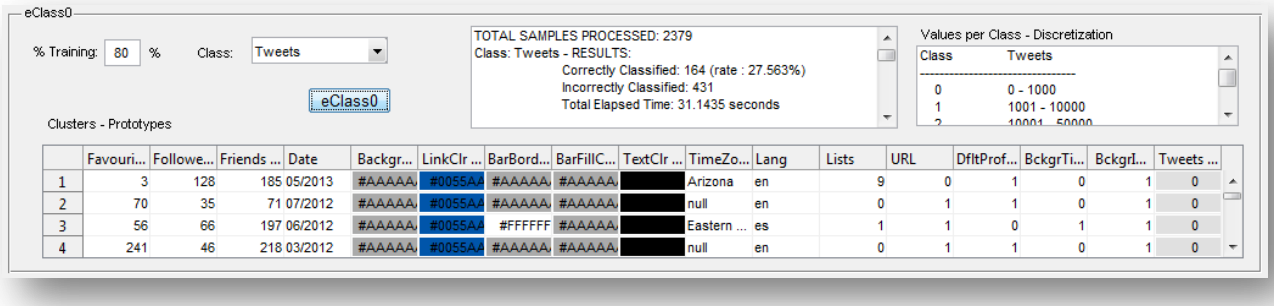
Sin embargo, este análisis es fruto de un solo entrenamiento con menos de 3000 datos. Los resultados obtenidos son interesantes, pero para poder sacar conclusiones firmes sería conveniente analizar un mayor número de ejemplos hasta encontrar una tendencia a un modelo constante.

5.1.4. eClass0

El clasificador que utilizamos tiene un uso similar a cualquier otro clasificador común. Por una parte se fija el porcentaje de ejemplos utilizado para la fase de entrenamiento, dejando el resto para la parte de test, y por otra se elige el atributo que se desea tomar como clase. Tomaremos un valor del 80% de entrenamiento para todas las pruebas que vamos a mostrar. Debido a que podemos clasificar por varios atributos, para la demostración de uso e interpretación de resultados del algoritmo vamos a desarrollar dos casos individualmente: la clasificación con un atributo con 5 valores posibles, y con un atributo binario de 2 clases únicamente.

La ejecución del algoritmo genera una información textual y el detalle de los clústers del modelo. El resultado más significativo es la tasa de acierto, que es lo que realmente interesa en el empleo de un clasificador. Sin embargo, este dato por sí mismo no tiene sentido si no se analiza en función del resto de información que se muestra. Otro dato interesante es el tiempo empleado en el proceso. El tiempo de cómputo es un factor muy determinante a la hora de comparar diferentes clasificadores, por lo que mostrándolo podremos comparar nuestro clasificador con otros en función de efectividad (% de acierto) y tiempo empleado.

– Clasificación por Tweets (5 valores)



	Favouri...	Followe...	Friends ...	Date	Backgr...	LinkClr ...	BarBord...	BarFillC...	TextClr ...	TimeZo...	Lang	Lists	URL	DfltProf...	BckgrTi...	BckgrI...	Tweets ...
1	3	128	185	05/2013	#AAAAA	#0055AA	#AAAAA	#AAAAA		Arizona	en	9	0	1	0	1	0
2	70	35	71	07/2012	#AAAAA	#0055AA	#AAAAA	#AAAAA		null	en	0	1	1	0	1	0
3	56	66	197	06/2012	#AAAAA	#0055AA	#FFFFFF	#AAAAA		Eastern ...	es	1	1	0	1	1	0
4	241	46	218	03/2012	#AAAAA	#0055AA	#AAAAA	#AAAAA		null	en	0	1	1	0	1	0

Ilustración 23

Lo primero que hay que observar es el número de ejemplos que se han procesado para generar el modelo clasificatorio. Si es un número bajo, es normal que el modelo necesite más ciclos de entrenamiento con más datos para afinar los resultados. De lo contrario, si el número de ejemplos procesados es alto, ya debemos considerar los resultados como definitivos.

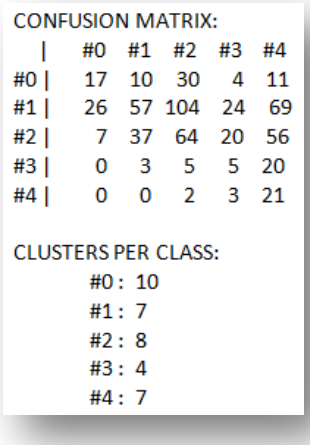


Ilustración 24

Por otro lado, el programa devuelve la matriz de confusión del proceso de testeo, así como los clústers utilizados para representar cada clase en el modelo. En la matriz de confusión se muestra la clasificación real en las filas contrastada con la clasificación que realiza el algoritmo en las columnas. En nuestro caso podemos apreciar que el algoritmo tiende a clasificar los perfiles con más tweets de los que realmente tienen, ya que en general asigna clases más altas

que las reales. Además, se puede leer información de la densidad de perfiles por clase. Por ejemplo, observamos que la clase mayoritaria es la #1 porque si sumamos todos los valores de la fila correspondiente obtenemos el valor más alto. Por el contrario la clase #4 es la minoritaria.

Respecto a los clústers por clase, se aprecia un ligero equilibrio, siendo la clase #0 la más numerosa. Esto significa que, teniendo en cuenta los ejemplos utilizados para el entrenamiento, la clase #0 tiene mayor variedad y/ó densidad. Es decir, es la que más ejemplos agrupa, o al menos una mayor variedad entre ellos. Debido a este reparto y configuración del modelo entrenado, es importante tener en cuenta que a la hora de realizar la fase de test, es más probable que un ejemplo se asocie a una clase con mayor número de clústers que a una con poca representación. Por lo tanto, además de la propia combinación de valores de los ejemplos y clústers, el reparto de clústers por clase también es un factor muy determinante en el porcentaje de acierto final del algoritmo.

Volviendo a la matriz de confusión, con las reflexiones descritas, vemos un desequilibrio entre el conjunto de ejemplos de entrenamiento y test. Mientras que la clase más representativa en entrenamiento es la clase #0 (10 clústers generados), en el conjunto de test es la clase #1 (es la que más ejemplos contiene).

Podemos concluir que el porcentaje de acierto es muy bajo. Los factores que influyen en eso pueden ser: un número bajo de ejemplos procesados, la variedad de clases y la diferencia detectada entre conjunto de entrenamiento y test. Además existe otro factor importante que es la correlación existente entre los atributos de decisión y el atributo de clase. Sin embargo, dada la variedad de causas posibles es complejo encontrar un motivo concreto que provoque este bajo porcentaje de acierto.

- Clasificación por Default Profile (2 valores)

Pasamos ahora a analizar los resultados de la clasificación con una clase binaria, en este caso Default Profile. En las pruebas realizadas durante la extracción de datos se ha detectado que es un atributo con bastante tendencia a adoptar el valor '0'. Además tiene influencia directa de otros atributos como por ejemplo los colores, ya que una condición para considerar que el usuario mantiene la configuración de perfil por defecto, es que mantenga los colores predefinidos en la creación de la cuenta.

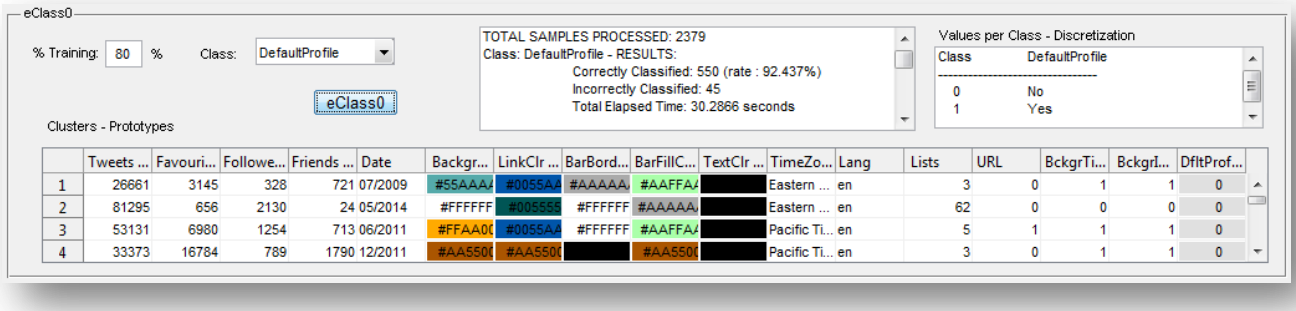


Ilustración 25

Al contrario que en el análisis de clasificación con el atributo Tweets, vemos cómo en este caso el porcentaje de acierto es muy alto. Las circunstancias expuestas anteriormente tienen mucho que ver en este resultado, pero vamos a compararlo con las posibles causas de error del caso anterior partiendo de situación y conjunto de datos iguales. Deducimos que la reducción de 5 a 2 clases posibles es un motivo más que evidente en la obtención de este resultado. Si hablamos en términos de probabilidad, podemos decir que es más difícil que el algoritmo se equivoque.

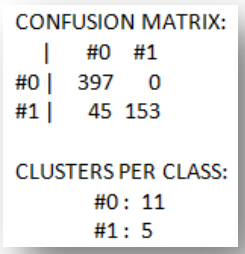


Ilustración 26

Si analizamos la matriz de confusión con los clústers por clase de igual manera que lo hemos hecho en el ejemplo anterior, aquí vemos una relación entre ambos. La clase mayoritaria es la clase #0, tanto en la matriz de confusión como en el número de clústers que la representan. Además, coincide que en ambos indicadores el valor de la clase #0 es el doble que la clase #1 y se puede establecer una relación directa entre ejemplos agrupados por clúster.

Los únicos errores que se producen pertenecen a una incorrecta clasificación de ejemplos de clase #1 clasificados como #0. Esto se debe a que al estar muy desequilibrado el número de clústers por clase, es bastante probable que un ejemplo en fase de testeo encuentre más similitud con un clúster de clase #0 que de #1. En cualquier caso, es un error bajo que deja en evidencia otros factores como causantes de la buena tasa de acierto. Como se ha detallado anteriormente, existe una buena correlación entre los atributos de decisión y el atributo de clase, por lo que es la causa más probable del éxito obtenido.

5.2. Comparativa de modelos según intervalo de tiempo

En este apartado vamos a retomar el análisis desarrollado anteriormente (palabra de búsqueda: 'music') y vamos a ejecutar de nuevo los mismos algoritmos. Lo que se pretende es analizar el efecto temporal en la extracción de datos, así como la acumulación de ejemplos procesados por cada algoritmo.

5.2.1. Extracción de datos

El primer paso es la extracción de datos. Recordemos que la primera ejecución del análisis se realizó el 25/05/2015 a las 20:07:45, tal y como podemos leer en el fichero de log. La fecha del tweet más antiguo extraído fue el 16/05/2015 a las 22:50. La nueva extracción se realiza el 28/05/2015 a las 10:34:37, y recolectan 2901 perfiles. La fecha del tweet más antiguo es el 19/05/2015 a las 19:04, por lo que es muy probable que existan muchos datos repetidos.

Ilustración 27

¿Cuántos? Para ello observamos de nuevo el log del análisis y vemos cómo casi todas las geolocalizaciones tienen 100 perfiles encontrados, es decir, la densidad de publicación es más alta y por lo tanto es menos probable que sean datos repetidos. Por el contrario, vemos cómo en 5 geolocalizaciones (Manaos, Santiago de Chile, Vilna, Omsk y La Meca) se recolectan menos perfiles. El dato más antiguo que distorsiona nuestras conclusiones es muy probable que se encuentre aquí, por lo que si pretendemos asegurarnos de que todos los datos que vamos a procesar son diferentes de la ejecución anterior, sería muy acertado eliminar estas geolocalizaciones. Ya que aún no hemos entrenado ningún algoritmo, vamos a probar el cambio, eliminando las geolocalizaciones con poca densidad sólo para este análisis y observando la fecha del tweet más antiguo.

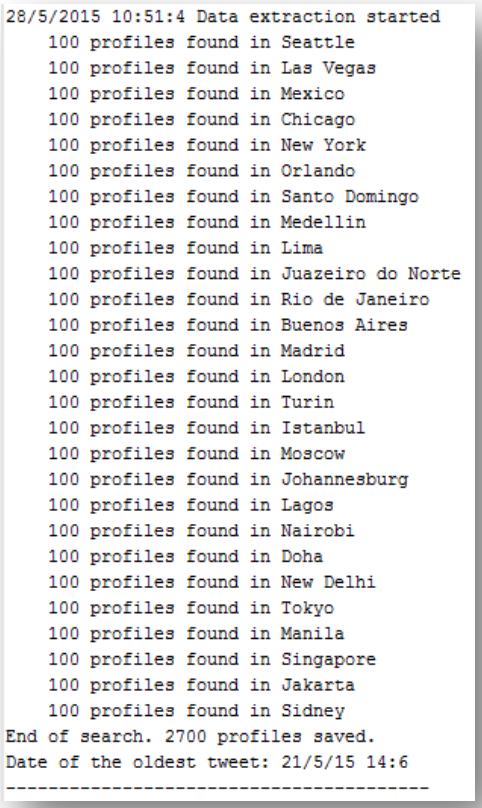


Ilustración 28

Efectivamente, ahora todas las geolocalizaciones tienen 100 perfiles encontrados. Sin embargo hay un dato que no cumple nuestras expectativas. La diferencia de fecha del tweet más antiguo comparada con la extracción anterior no es tan grande como esperábamos. El margen que existe entre el último tweet y la fecha del último análisis es de 4 días. Por lo tanto no podemos saber con certeza cuántos perfiles repetidos contiene el conjunto de datos, en relación con el análisis anterior. En cualquier caso procedemos a ejecutar los algoritmos con esta última extracción de datos.

5.2.2. RDE

En la ejecución anterior utilizamos un valor de sigma de 3.4 para analizar el conjunto de datos, obteniendo 4 *outliers*. En este caso, si utilizamos ese valor no obtenemos ningún *outlier*, por lo que debemos reducir el valor de sigma. Mediante prueba y error, llegamos a un valor de 2.7, con el cual obtenemos 3 *outliers*.

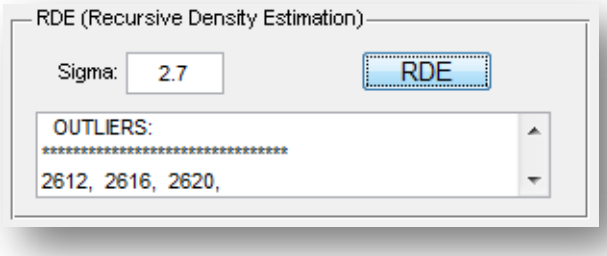


Ilustración 29

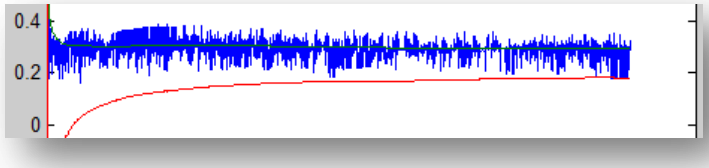
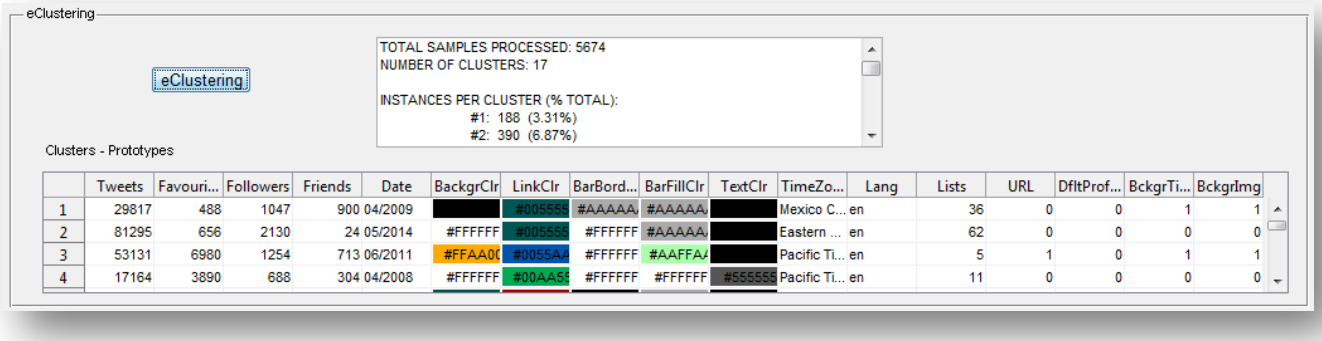


Ilustración 30

La conclusión que podemos sacar de éste algoritmo en relación a la ejecución anterior es que en este caso el conjunto de datos está menos disperso y los ejemplos tienen mayor relación entre sí. Si analizamos el gráfico y la numeración de los *outliers*, observamos que éstos se encuentran al final del conjunto de datos, de igual manera que en la ejecución anterior. Este hecho nos puede llevar a suponer que las últimas geolocalizaciones de las que se extraen datos (más concretamente la última, que es Sidney) tienen menor similitud con los perfiles del resto de geolocalizaciones.

5.2.3. eClustering

Vamos ahora a analizar cómo evoluciona el modelo asociado al conocimiento acumulado en el algoritmo eClustering. Lo primero que percibimos es un aumento en el número de clústers. Como ya se ha especificado anteriormente, es una cantidad que suele aumentar, pero nunca puede reducir, simplemente por la naturaleza del algoritmo.



	Tweets	Favouri...	Followers	Friends	Date	BackgrClr	LinkClr	BarBord...	BarFillClr	TextClr	TimeZo...	Lang	Lists	URL	DfltProf...	BckgrTi...	BckgrImg
1	29817	488	1047	900	04/2009		#005555	#AAAAA	#AAAAA		Mexico C...	en	36	0	0	1	1
2	81295	656	2130	24	05/2014	#FFFFFF	#005555	#FFFFFF	#AAAAA		Eastern ...	en	62	0	0	0	0
3	53131	6980	1254	713	06/2011	#FFAA0(	#0055AA	#FFFFFF	#AAFFA4		Pacific Ti...	en	5	1	0	1	1
4	17164	3890	688	304	04/2008	#FFFFFF	#00AA55	#FFFFFF	#FFFFFF	#555555	Pacific Ti...	en	11	0	0	0	0

Ilustración 31

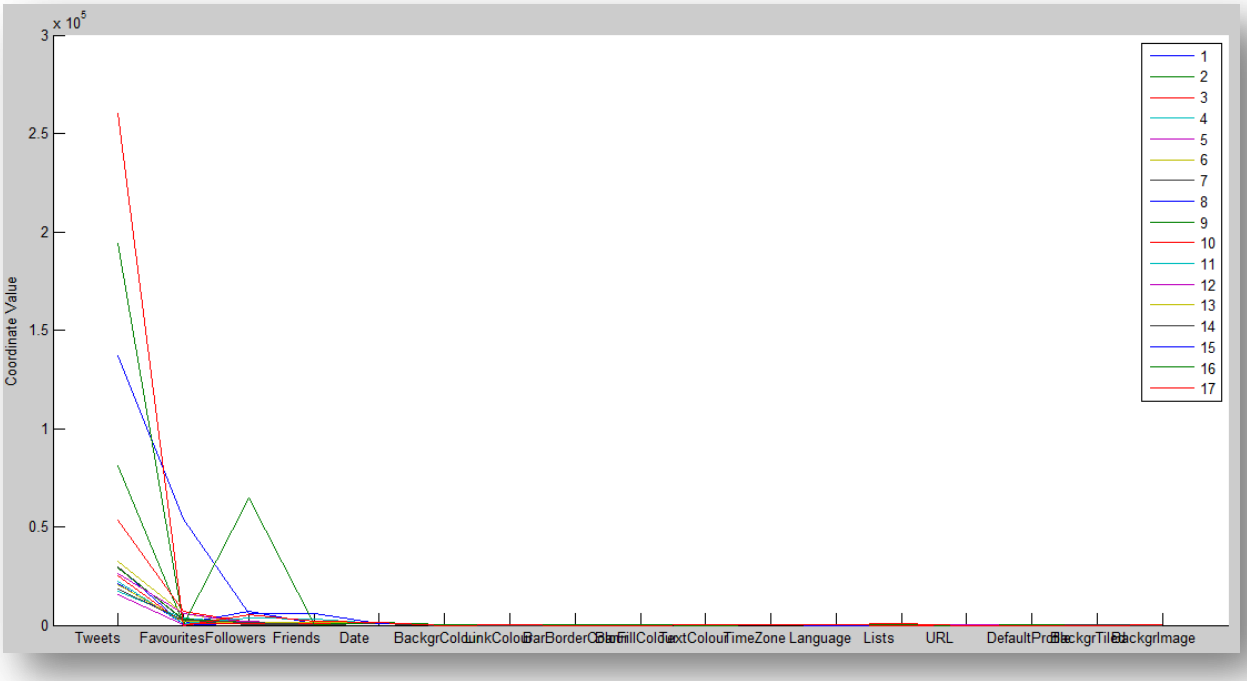


Ilustración 32

Si observamos el gráfico, vemos cómo en el atributo Tweets (que es el más apreciable) se ha producido un aumento considerable de la escala de valores. En el caso anterior existía un clúster con un número de tweets de 80000, mientras que ahora hay 3 clústeres que superan con creces esta cifra, superando incluso a los 250000 tweets. Vamos a analizar los valores exactos de los nuevos clústers creados (los tres últimos) para ver si se corresponden con los datos altos que vemos en el gráfico.

	Tweets	Favouri...	Followers	Friends	Date	BackgrClr	Link
14	18557	2758	1286	498	04/2010		
15	137225	53758	5689	5710	07/2009	#AAAAA	#00
16	194239	4	64876	9	02/2013	#AAAAA	#005
17	259995	0	5595	1962	12/2014		#00A

Ilustración 33

Efectivamente se corresponden con los elevados valores que se detectan visualmente. Esto significa que son los tres tipos de perfil representativo que se han encontrado distintos respecto al modelo que ya existía. Analizando más detenidamente cada uno de los nuevos clústeres, se observa que el último tiene una tendencia muy diferente al resto. En la fecha de realización del análisis, la cuenta tiene una antigüedad de 6 meses, y sin embargo tiene un elevado número de tweets publicados. Esto nos puede llevar a suponer que se trata de una cuenta masiva con una serie de publicaciones programadas, ya que además no tiene marcado ningún tweet como favorito (acción que se suele asociar a un humano directamente). Si

realizamos cálculos, 259995 tweets publicados en 6 meses resultan en 1428 tweets por día, y 60 tweets por hora.

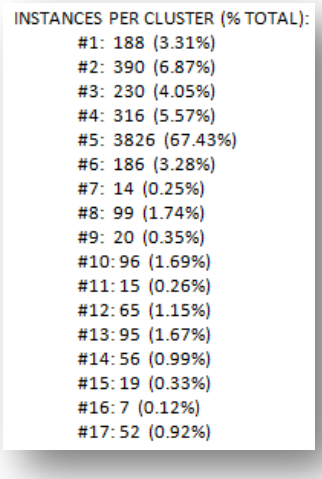
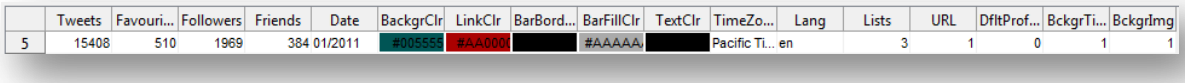


Ilustración 34

Atendiendo a los ejemplos asociados a cada clúster vamos a hacer dos reflexiones. Por un lado, continuando con el estudio del clúster #17, vemos que contiene 52 perfiles. A priori puede resultar significativo, pero dada la conclusión a la que hemos llegado anteriormente, es un dato que altera nuestro análisis. Recordemos que el programa extrae tweets individuales, de los cuales obtiene los datos del usuario que publica dicho tweet. Por lo tanto, si la cuenta asociada al clúster #17 twittea un promedio de 60 tweets por hora, es muy probable que los 52 ejemplos asociados a dicho clúster se relacionen con la misma cuenta.

Por otro lado, con una visión más general, nos fijamos en el clúster mayoritario. En este caso sigue siendo el número #5 al igual que antes, e incluso ha aumentado el porcentaje de ejemplos agrupados, subiendo el nivel de representatividad.



	Tweets	Favouri...	Followers	Friends	Date	BackgrClr	LinkClr	BarBord...	BarFillClr	TextClr	TimeZo...	Lang	Lists	URL	DfltProf...	BckgrTi...	BckgrImg
5	15408	510	1969	384	01/2011	#005555	#AA0000		#AAAAAA		Pacific Ti...	en	3	1	0	1	1

Ilustración 35

Analizando los valores en busca de posibles cambios en alguno de los atributos, vemos que se mantiene en el mismo estado que la ejecución anterior, por lo que podemos concluir que es un modelo muy representativo del análisis vigente.

5.2.4. eClass0

Para la continuación del análisis de este algoritmo mantenemos el parámetro de 80% de entrenamiento y realizamos las dos clasificaciones con los atributos usados anteriormente (Tweets y Default Profile).

- Clasificación por Tweets (5 valores)

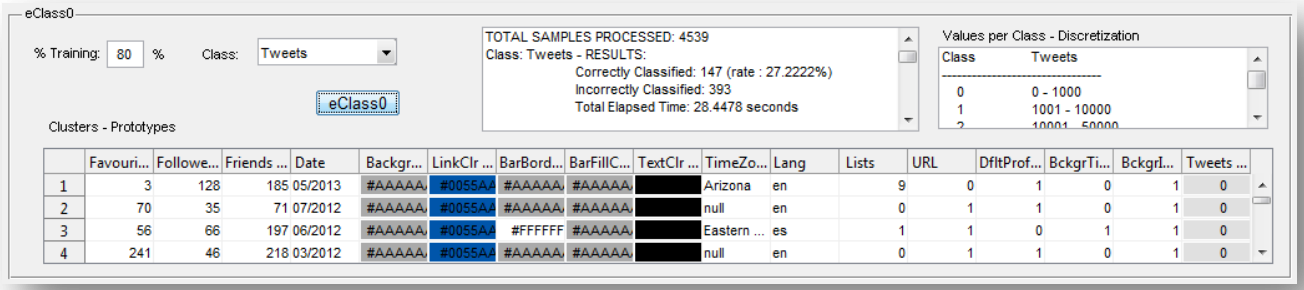


Ilustración 36

El primer dato que nos llama la atención es el porcentaje de acierto. Es levemente inferior al caso anterior. A continuación estudiaremos las posibles causas analizando la matriz de confusión y el número de clústers por clase. El tiempo de procesamiento es inferior, pero es una consecuencia directa del número de ejemplos procesados, ya que en esta ejecución se han extraído algunos datos menos que en la primera.

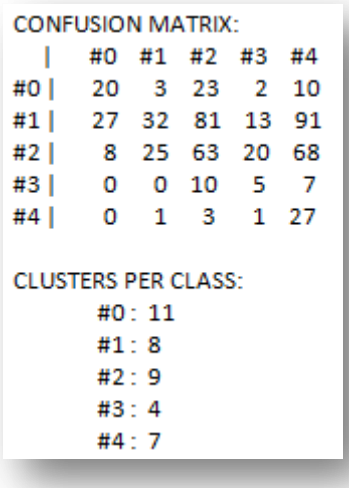


Ilustración 37

Vemos cómo la tendencia del clasificador es similar a la ejecución anterior, tendiendo a asignar clases superiores a las reales a la mayoría de ejemplos testeados. La distribución del número de clústers por clase es similar y no supone grandes cambios respecto al modelo anterior. Únicamente se aumenta en un clúster el modelo de las tres primeras clases. En general, si comparamos con la ejecución previa, la distribución de errores es similar y por lo tanto se puede concluir que la correlación entre los atributos de decisión no es buena en el conjunto de datos utilizado (tanto la primera extracción como la segunda).

– Clasificación por Default Profile (2 valores)

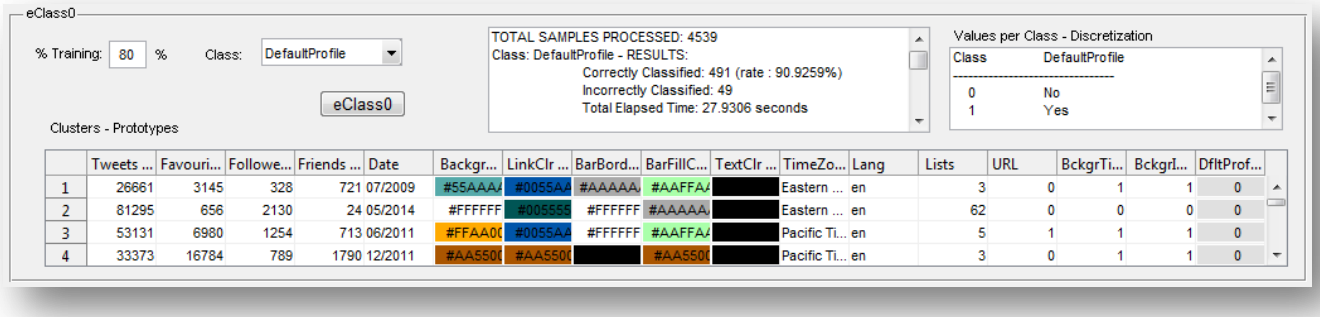


Ilustración 38

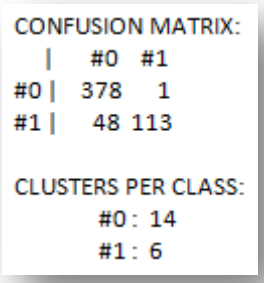


Ilustración 39

En este caso notamos un descenso en el porcentaje de acierto. Comparando la matriz de confusión y el número de clústers por clase con la etapa anterior del análisis vemos cómo el mayor porcentaje de error surge en la clase #1. Antes se clasificaban mal el 30% de ejemplos pertenecientes a la clase #1 (asignándoles la clase #0), mientras que ahora clasifica mal el 42% de los mismos. El desequilibrio que existía entre el número de clústers por clase se ha aumentado, resultando la clase #0 mayoritaria. Esto, sumado a la poca relación entre los datos da como resultado el descenso del porcentaje de acierto.

5.3. Comparativa de modelos según palabra de búsqueda

En este apartado vamos a evaluar la interpretación de resultados ofrecidos en análisis diferentes, en función de su palabra de búsqueda. Para ello vamos a tomar como referencia el primer análisis realizado con la palabra 'music'. Se trata de una palabra en inglés y de tipo muy genérico, por lo que para encontrar un contraste claro de datos vamos a seleccionar una palabra de ámbito más concreto. Una discretización importante es el idioma de la propia palabra de búsqueda, así que si escogemos una palabra en un idioma menos popular,

obtendremos un menor número de resultados. La palabra seleccionada ha sido 'musik', que significa música en varios idiomas, como por ejemplo alemán e indonesio.

5.3.1. Extracción de datos

Como resultado de la extracción de datos, obtenemos 333 perfiles, un número no muy elevado que es precisamente lo que buscábamos. La fecha del último tweet también es amplia en este caso, 9 días anterior a la realización del análisis. La densidad de publicación es por tanto menor que en el otro caso, ya que es un intervalo de tiempo similar y se han recogido menos datos.

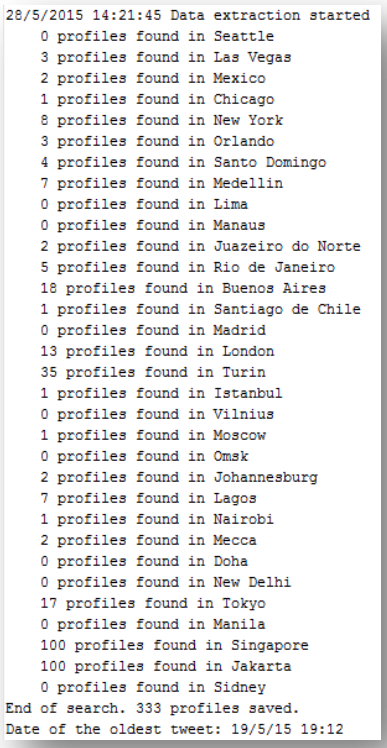


Ilustración 40

Si observamos detalladamente el origen de los datos, apreciamos la influencia del idioma. Hay muchas geolocalizaciones con algunos perfiles, pero las mayoritarias son Singapur y Yakarta, que tienen idioma indonesio.

5.3.2. RDE

Para detectar los *outliers* más significativos y ver cuál es la relación entre el conjunto de datos se ha llegado a fijar un valor de sigma 2.0, lo que significa que los datos recopilados están más relacionados entre sí que en el caso anterior. De igual modo, los *outliers* aparecen en las últimas geolocalizaciones, pero en este caso no podemos establecer similitud con el otro análisis porque la última geolocalización no es la misma (antes era Sidney, mientras que ahora es Yakarta). El resultado es bueno porque cumple con las expectativas: buscamos un conjunto

de datos de ámbito más reducido que guarden más relación entre sí, y éste indicador lo demuestra.

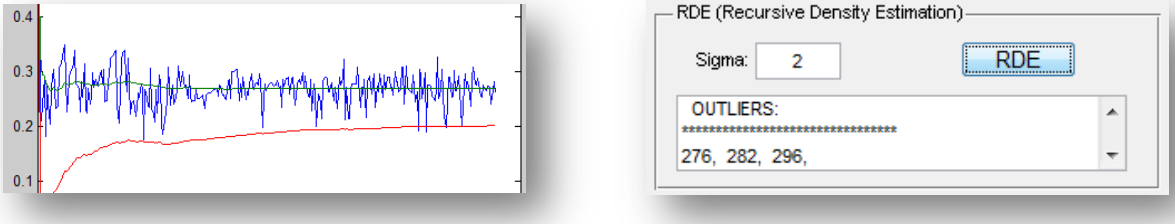
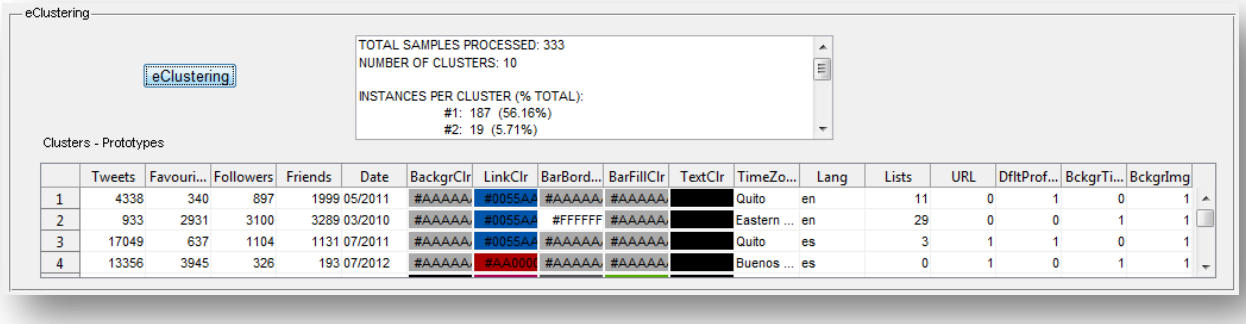


Ilustración 41

5.3.3. eClustering



	Tweets	Favouri...	Followers	Friends	Date	BackgrClr	LinkClr	BarBord...	BarFillClr	TextClr	TimeZo...	Lang	Lists	URL	DfltProf...	BckgrTi...	BckgrImg
1	4338	340	897	1999	05/2011	#AAAAA	#0055AA	#AAAAA	#AAAAA		Quito	en	11	0	1	0	1
2	933	2931	3100	3289	03/2010	#AAAAA	#0055AA	#FFFFFF	#AAAAA		Eastern ...	en	29	0	0	1	1
3	17049	637	1104	1131	07/2011	#AAAAA	#0055AA	#AAAAA	#AAAAA		Quito	es	3	1	1	0	1
4	13356	3945	326	193	07/2012	#AAAAA	#AA0000	#AAAAA	#AAAAA		Buenos ...	es	0	1	0	1	1

Ilustración 42

El procesado de los datos con el algoritmo eClustering genera como resultado 10 clústeres representativos. En el gráfico podemos observar que la tendencia mayoritaria es que las cuentas tengan alrededor de 16000 tweets publicados, como dato más visible.

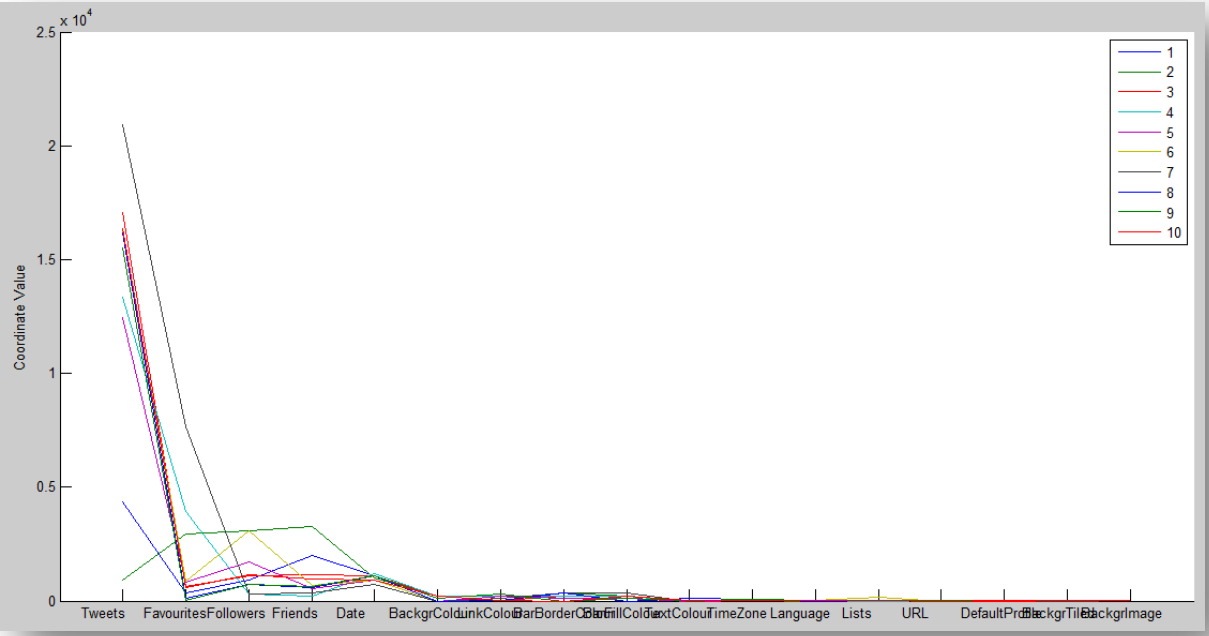


Ilustración 43

```
INSTANCES PER CLUSTER (% TOTAL):
        #1: 187  (56.16%)
        #2: 19  (5.71%)
        #3: 43  (12.91%)
        #4: 5  (1.50%)
        #5: 34  (10.21%)
        #6: 2  (0.60%)
        #7: 39  (11.71%)
        #8: 1  (0.30%)
        #9: 2  (0.60%)
        #10:1  (0.30%)
```

Ilustración 44

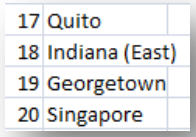
Sin embargo la deducción anterior no es del todo correcta, ya que si nos fijamos en el porcentaje de ejemplos que abarca cada clúster, el número #1 es muy alto. Examinando sus valores concretos vemos cómo contiene únicamente 4338 tweets, nada que ver con la suposición anterior. Por lo tanto podemos concluir que la condensación visual de clústers en el atributo 'Tweets' se debe a la variedad de combinaciones del resto de atributos entre perfiles que comparten el mismo número de tweets publicados.

	Tweets	Favouri...	Followers	Friends	Date	BackgrClr	LinkClr	BarBord...	BarFillClr	TextClr	TimeZo...	Lang	Lists	URL	DfltProf...	BckgrTi...	BckgrImg
1	4338	340	897	1999	05/2011	#AAAAA	#0055A4	#AAAAA	#AAAAA		Quito	en	11	0	1	0	1

Ilustración 45

Por otra parte, vemos cómo la zona horaria del perfil mayoritario es Quito (Ecuador), y el idioma es inglés. Nada que ver con lo que refleja la extracción de datos (200 perfiles de los 333 totales pertenecen a Singapur y Yakarta) y el idioma de la palabra de búsqueda. Esto nos lleva

a una conclusión más avanzada, fuera de los límites de nuestro estudio. La información que ofrece Twitter de cada cuenta no siempre tiene sentido a primera vista. Por una parte, puede que el usuario establezca el idioma de uso en inglés, aunque twitee en otro idioma. Respecto a la zona horaria es más extraño, no es lógico que haya un número mayoritario de perfiles de Ecuador cuando la extracción completa de América del Sur no es tan numerosa. Vamos a analizar la codificación enumerada de éste campo:

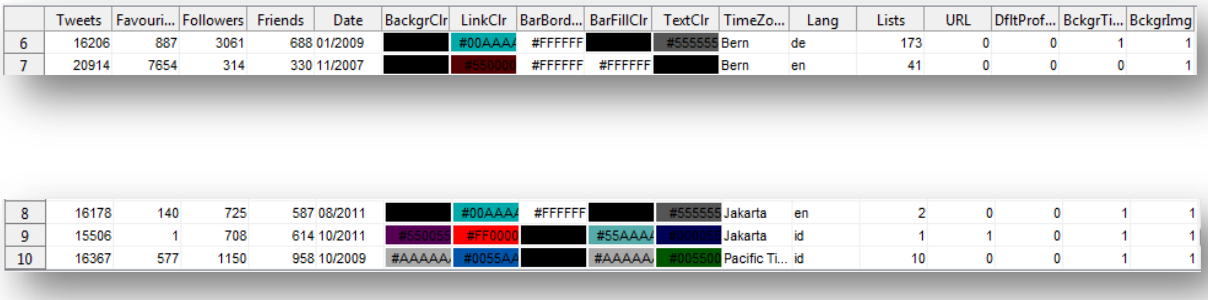


17	Quito
18	Indiana (East)
19	Georgetown
20	Singapore

Ilustración 46

Curiosamente, los valores de Singapur y Quito están muy próximos, por lo que es muy probable que la zona horaria de Quito correspondiente al clúster #1 provenga de un ajuste del algoritmo, que en nuestro caso no es útil porque distorsiona la información real al no ser un campo puramente numérico.

Analizamos la zona horaria del resto de clústeres, y observamos que el número #4 es de Buenos Aires, y agrupa 5 perfiles (mientras que en la extracción de datos se recogieron 18 perfiles). Por otro lado, retomando la elección de la palabra de búsqueda 'musik', apuntamos que entre otros idiomas existía en alemán e indonesio. Efectivamente vemos cómo los clústers #6 y #7 tienen zona horaria de Berna (Suiza) por lo que podemos asociarlo a Alemania. Los idiomas de ambos clústers son el alemán y el inglés, que si lo enfocamos con la reflexión anterior tiene sentido porque Alemania es un país muy bilingüe. Los ejemplos que agrupan éstos clústers son el 12.31% del total, siendo un número de representación aceptable.



	Tweets	Favouri...	Followers	Friends	Date	BackgrClr	LinkClr	BarBord...	BarFillClr	TextClr	TimeZo...	Lang	Lists	URL	DfltProf...	BckgrTi...	BckgrImg
6	16206	887	3061	688	01/2009		#00AAA	#FFFFFF		#555555	Bern	de	173	0	0	1	1
7	20914	7654	314	330	11/2007		#550000	#FFFFFF	#FFFFFF		Bern	en	41	0	0	0	1
8	16178	140	725	587	08/2011		#00AAA	#FFFFFF		#555555	Jakarta	en	2	0	0	1	1
9	15506	1	708	614	10/2011	#55005	#FF0000		#55AAA		Jakarta	id	1	1	0	1	1
10	16367	577	1150	958	10/2009	#AAAAA	#0055AA		#AAAAA	#005500	Pacific Ti...	id	10	0	0	1	1

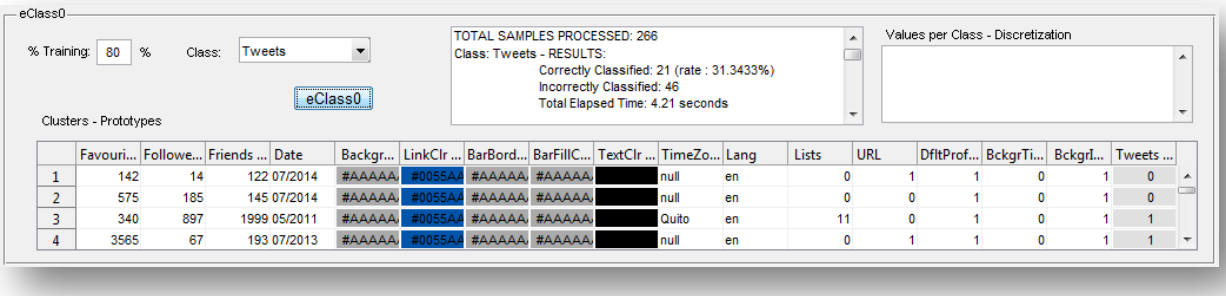
Ilustración 47

Si buscamos la representación indonesia encontramos que los clústers #8, #9 y #10 tienen zona horaria Jakarta con idiomas inglés e indonesio. Pero el porcentaje de representación entre los tres es del 1.2%, un indicador muy bajo dadas las circunstancias del análisis.

5.3.4. eClass0

Procedemos a ejecutar el algoritmo eClass0. Se busca comparar los porcentajes de acierto con el análisis anterior, que derivan de la correlación que exista entre los datos de cada uno. Un dato significativo es el tiempo de ejecución, que en este caso es mucho menor dado el reducido número de ejemplos procesados, en comparación con el análisis anterior.

- Clasificación por Tweets (5 valores)



	Favouri...	Followe...	Friends ...	Date	Backgr...	LinkClr ...	BarBord...	BarFillC...	TextClr ...	TimeZo...	Lang	Lists	URL	DfltProf...	BckgrTi...	BckgrI...	Tweets ...
1	142	14	122	07/2014	#AAAAA	#0055AA	#AAAAA	#AAAAA		null	en	0	1	1	0	1	0
2	575	185	145	07/2014	#AAAAA	#0055AA	#AAAAA	#AAAAA		null	en	0	0	1	0	1	0
3	340	897	1999	05/2011	#AAAAA	#0055AA	#AAAAA	#AAAAA		Quito	en	11	0	1	0	1	1
4	3565	67	193	07/2013	#AAAAA	#0055AA	#AAAAA	#AAAAA		null	en	0	1	1	0	1	1

Ilustración 48

En este caso el porcentaje de acierto es algo mayor. Puede ser debido tanto al menor número de datos como a la mejor relación que guardan entre ellos. En el análisis con el algoritmo RDE vimos cómo en este caso los datos están más relacionados entre sí que en el análisis anterior.

CONFUSION MATRIX:

	#0	#1	#2	#3	#4
#0	6	2	2	0	0
#1	9	7	12	2	0
#2	3	12	8	1	0
#3	0	2	1	0	0
#4	0	0	0	0	0

CLUSTERS PER CLASS:

#0: 2
#1: 5
#2: 4
#3: 3
#4: 3

Ilustración 49

Analizando la matriz de confusión nos encontramos una situación muy diferente de las analizadas hasta ahora. Tomando como referencia la diagonal correspondiente a los aciertos, vemos cómo los errores se distribuyen alrededor de la misma, estando equilibrados ambos lados de la diagonal. Si contrastamos con la lectura del número de clústeres por clase, se aprecia un equilibrio entre todas las clases por lo que deducimos que el entrenamiento va por buen camino, pero le faltan ciclos de ejecución. Para comprobarlo, exponemos al algoritmo a una nueva ejecución con una nueva extracción de datos. Dicha extracción se realiza el 29/05/2015 a las 12:47 y los resultados son los siguientes:

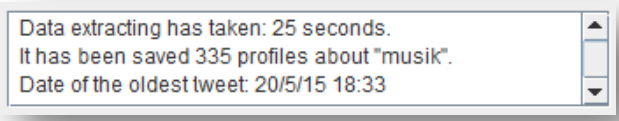


Ilustración 50

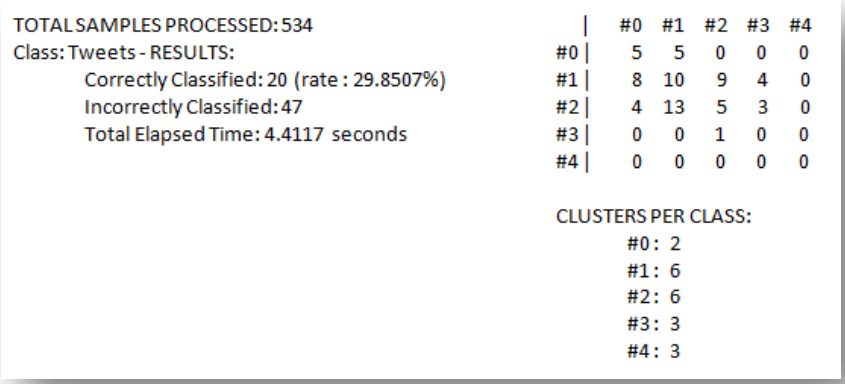


Ilustración 51

Los resultados empeoran y se deteriora el equilibrio de clústers por clase. No se han cumplido las predicciones establecidas y se plantea una vez más la poca relación entre datos como causa del fracaso.

- Clasificación por Default Profile (2 valores)

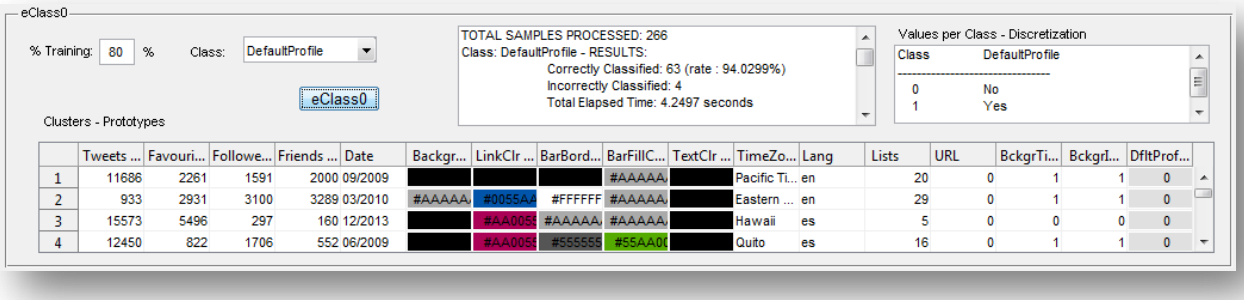


Ilustración 52

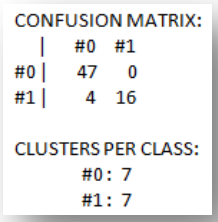


Ilustración 53

Partiendo de la base de que la clasificación con este atributo es mejor al ser binario, en este caso se ha mejorado el porcentaje de acierto conseguido en el análisis anterior. El número de clústers por clase está perfectamente equilibrado entre ambas y la matriz de confusión tampoco aporta demasiada información adicional.

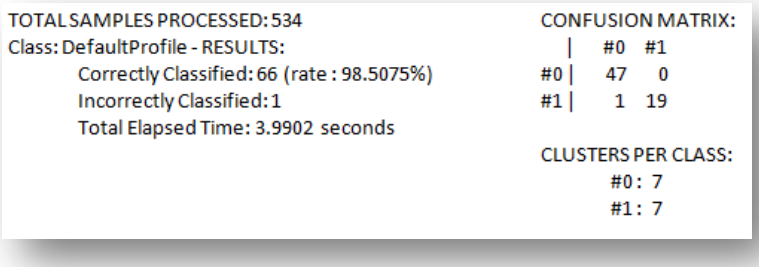


Ilustración 54

Si volvemos a entrenar el modelo con los datos de la nueva ejecución como en el atributo 'Tweets', conseguimos aumentar el porcentaje de acierto, resultando únicamente un ejemplo mal clasificado. El número de clústers por clase se mantiene equilibrado e igual en número, por lo que el modelo establecido es bueno y apenas necesita readaptarse.

5.4. Influencia del orden de extracción de las geolocalizaciones

Los algoritmos que utilizamos tienen aprendizaje de tipo incremental. Esto significa que el conocimiento que acumulan se conforma a lo largo de las sucesivas ejecuciones. En nuestro caso esto tiene repercusiones negativas ya que los datos procesados en cada ejecución tienen el sesgo del orden por geolocalizaciones. A priori puede no parecer ser un gran inconveniente, pero teniendo en cuenta la dinámica de funcionamiento del eClustering y eClass0 es razonable que sí lo es. Para comprobarlo vamos a realizar una prueba a raíz del análisis de 'musik' anterior, modificando el orden de las geolocalizaciones situando las tres últimas del fichero original al principio (Singapur, Yakarta y Sídney). Veamos cómo influye este cambio:

```
29/5/2015 13:5:36 Data extraction started
    100 profiles found in Singapore
    100 profiles found in Jakarta
    0 profiles found in Sidney
    0 profiles found in Seattle
    2 profiles found in Las Vegas
    2 profiles found in Mexico
    1 profiles found in Chicago
    8 profiles found in New York
    5 profiles found in Orlando
    4 profiles found in Santo Domingo
    15 profiles found in Medellin
    0 profiles found in Lima
    0 profiles found in Manaus
    2 profiles found in Juazeiro do Norte
    4 profiles found in Rio de Janeiro
    17 profiles found in Buenos Aires
    2 profiles found in Santiago de Chile
    0 profiles found in Madrid
    15 profiles found in London
    32 profiles found in Turin
    1 profiles found in Istanbul
    0 profiles found in Vilnius
    1 profiles found in Moscow
    0 profiles found in Omsk
    2 profiles found in Johannesburg
    5 profiles found in Lagos
    1 profiles found in Nairobi
    0 profiles found in Mecca
    0 profiles found in Doha
    0 profiles found in New Delhi
    16 profiles found in Tokyo
    0 profiles found in Manila
End of search. 335 profiles saved.
Date of the oldest tweet: 20/5/15 18:33
```

Ilustración 55

En el algoritmo RDE obtenemos un valor de sigma de 1.6, mucho menor de los obtenidos hasta ahora. Tiene sentido que al procesarse primero todos los perfiles de Singapur y Yakarta (que representan más de la mitad del conjunto de datos) se establezca una densidad más adaptada desde el inicio a este análisis concreto.

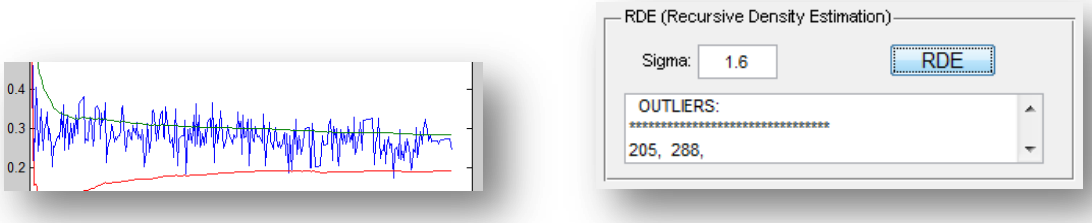


Ilustración 56

La mejora más notable la apreciamos en el algoritmo eClustering. El principal motivo que nos ha llevado a realizar ésta prueba es la poca coherencia que tenían los clústers generados en el análisis anterior. Veamos qué resultados ofrece el programa en este caso.

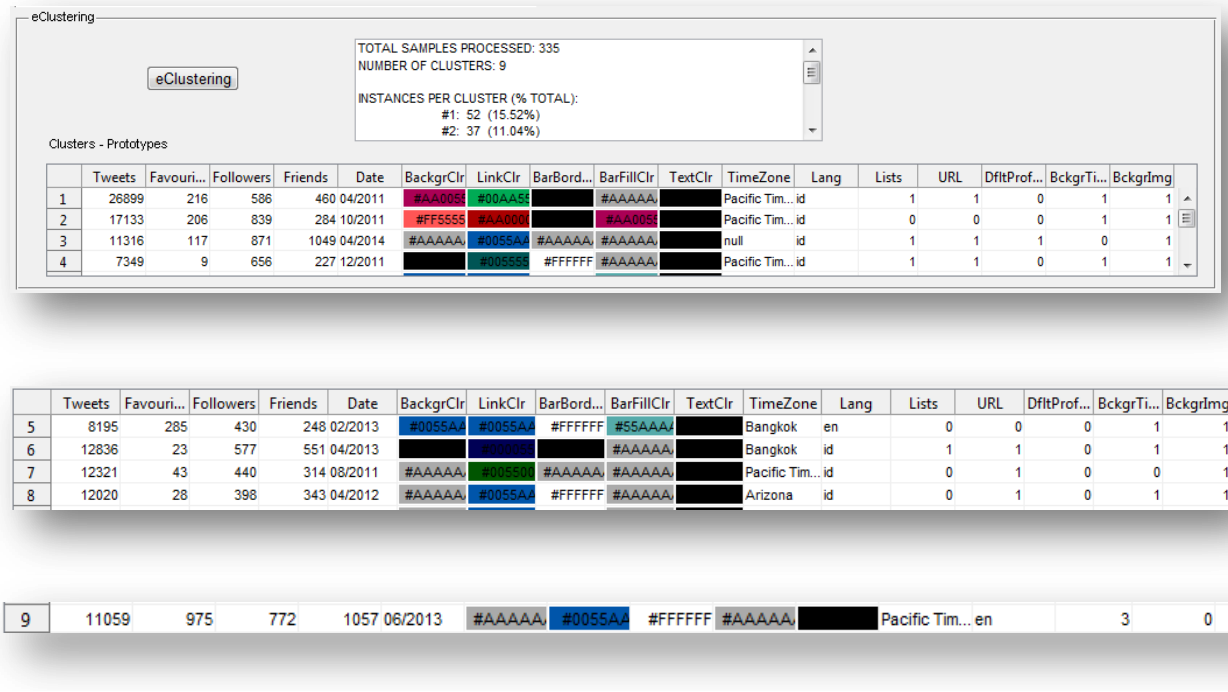


Ilustración 57

En primer lugar se reduce el número de clústers, por lo que la reorganización mejora la creación del modelo inicial tomado como base en las sucesivas iteraciones. Efectivamente se cumplen nuestras suposiciones y los clústers ahora son más coherentes. La mayoría tienen el indonesio como idioma, y sólo dos de ellos tienen el inglés. Respecto a las zonas horarias también notamos mejoría, ya que la mayoría son Pacific Time e incluso hay dos cuya zona horaria es Bangkok (capital de Tailandia, muy cercana a Singapur y Yakarta).

La conclusión clara de esta prueba es que al tratarse de un algoritmo de aprendizaje incremental, influye mucho el orden de entrenamiento para la obtención del modelo final. Las modificaciones que pueden suponer en un clúster el procesado de un ejemplo nuevo no son tan significativas como se puede suponer.

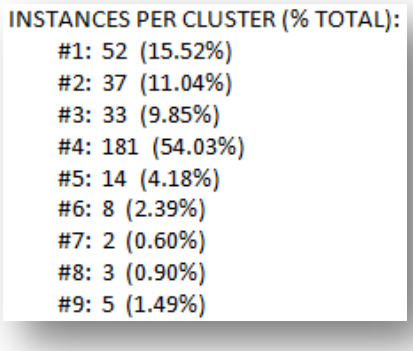


Ilustración 58

El clúster más representativo es el número #4, que sin nos fijamos en sus atributos detallados tiene zona horaria Pacific Time y como idioma el indonesio.

Se ha realizado una evaluación con el algoritmo eClass0 al igual que en el resto de análisis, pero los resultados empeoraban notablemente. Siguiendo con la reflexión anterior, éste algoritmo también sufre gravemente las consecuencias del orden de extracción de datos según las geolocalizaciones porque existe una separación entre conjunto de datos de entrenamiento y test. De este modo los datos empleados para test tienen más probabilidad de ser diferentes al resto por pertenecer a geolocalizaciones radicalmente distintas. En este caso se ha producido el efecto contrario que en eClustering, ya que el eClass0 se ha entrenado muy bien a los perfiles de Singapur y Yakarta, pero a la hora de evaluar el resto pertenecientes a muy diversas geolocalizaciones, el modelo generado no está adaptado y ofrece un mal resultado de clasificación.

5.5. Valoración de resultados obtenidos

Después de las pruebas realizadas hay dos aspectos a evaluar: la eficacia de los algoritmos y los datos que componen el dominio (en este caso la actividad de Twitter). La valoración depende de la combinación de ambos, ya que el objetivo del programa es extraer un conocimiento útil, derivado de unas relaciones y tendencias claras del conjunto de datos. Pero si en los datos tomados como base no existen tales relaciones no se podrán sacar conclusiones válidas. Vamos a analizar individualmente cada algoritmo para ver cómo interactúa con los datos y cómo de bueno puede ser el análisis ofrecido por cada uno desde una visión global.

5.5.1. RDE

En el caso del algoritmo RDE se analiza el conjunto de datos vigente sin ninguna otra relación con extracciones anteriores. De manera local nos permite medir una densidad media entre los ejemplos procesados, indicador que marca la correlación existente entre ellos. De todas las pruebas realizadas podemos concluir que es una buena funcionalidad del programa que cumple las expectativas y ayuda a la interpretación de resultados de los demás algoritmos.

5.5.2. eClustering

En general es un buen algoritmo pero la bondad de la clusterización depende mucho del orden de procesado de los datos. Como ya hemos visto, el eClustering ofrece buenos resultados pero en ocasiones es necesario hacer varias pruebas para poder sacar conclusiones coherentes. Como estudios futuros pueden variarse las constantes de modificación y adaptación de los clústeres en cada iteración, ya que por los resultados observados es muy decisivo el modelo inicial que crea el algoritmo y los valores de los clústers se modifican muy pocas veces. En el procesado de un nuevo ejemplo, es más frecuente que se asigne como tal a un clúster existente o que cree uno nuevo por sí mismo, pero rara vez se modifican los clústers existentes por lo que el aprendizaje no es del todo efectivo. Esta restricción afecta en la adaptación a diferentes conjuntos de datos y por eso no siempre es posible realizar una interpretación certera de los resultados.

5.5.3. eClass0

Tal y como se ha constatado en todas las pruebas, el hecho de que eClass0 sea un clasificador basado en clústeres es un factor que reduce su efectividad dentro del ámbito en que nos encontramos. Hemos comprobado que en el dominio de datos tratado en el programa el algoritmo no tiene capacidad de adaptación y el hecho de que los datos recolectados no sigan un patrón estrictamente constante no ayuda a mejorar los resultados. El tiempo de procesado es aceptable, pero la obtención de nuevos datos con los que entrenar y testear el algoritmo no es una tarea trivial en nuestro programa. Si precisamos de un gran número de ciclos de entrenamiento para mejorar resultados tenemos que realizar bastantes extracciones de datos lo que supone un coste en tiempo y en gestión por parte del usuario de la ejecución del algoritmo en cada una de ellas.

Además, como el medidor del algoritmo es el porcentaje de acierto en el testeo del conjunto de datos de test, la naturaleza de los datos que suponen el dominio del análisis no es adecuada para la separación entre conjuntos de entrenamiento y test. Se debería plantear otro tipo de separación, por ejemplo extraer ejemplos de test de forma aleatoria entre todos los del conjunto de datos, excluyéndolos del conjunto de entrenamiento.

En una visión más amplia, se podría haber realizado un estudio comparativo de la efectividad de nuestro clasificador con otros clasificadores incrementales (sometiéndolos a un entrenamiento y testeo con los mismos conjuntos de datos), pero los resultados obtenidos de manera individual en nuestro caso son malos y no tiene sentido compararlo con otros dentro del dominio de nuestro trabajo.

6. PLANIFICACIÓN Y PRESUPUESTO

6.1. Planificación

La planificación final que se ha seguido a lo largo del desarrollo del proyecto se ha visto muy influenciada por la paralelización de tareas con el resto de asignaturas universitarias. En el comienzo del proyecto se comenzó a trabajar desde la primera reunión con el tutor. Se hizo una distribución inicial de tareas significativas del proyecto, la cual se ha modificado a lo largo de su desarrollo dependiendo de las necesidades documentales que han ido surgiendo.

Respecto a la distribución general de etapas, se optó por comenzar con la parte técnica, implementando los programas en Java y MATLAB, y después continuar con la documentación y experimentación. A continuación se muestra la planificación seguida:

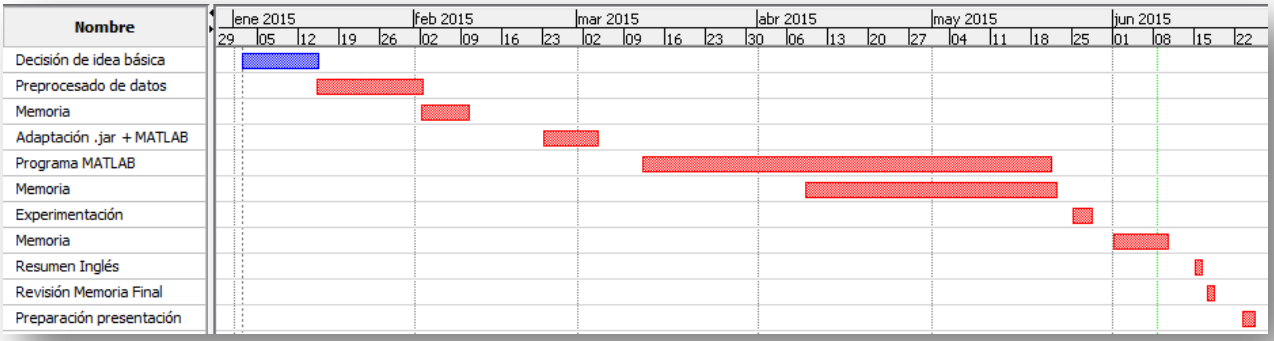


Ilustración 59

En total, se han empleado 140 días a tiempo parcial (una media aproximada de 2'5 horas diarias). Se puede ver cómo la mayor parte de tiempo la consume la implementación de la herramienta en MATLAB, así como su documentación. Por el contrario, las etapas finales de realización de Memoria son mucho más cortas. Esto es debido a la finalización del resto de asignaturas, pudiendo dedicar más tiempo diario al proyecto.

Por otro lado, las tareas existentes ofrecen poco paralelismo entre ellas, por lo que no podría configurarse un hipotético equipo de trabajo cuyos trabajadores desarrollasen etapas simultáneamente. La naturaleza del proyecto fuerza a que casi todas las fases se realicen de forma secuencial.

6.2. Presupuesto

Para el presupuesto del proyecto hay que contar con el tiempo de programación dedicado por el alumno, el gasto y amortización del equipo informático y la licencia de MATLAB ya que el resto de software utilizado es gratuito. Vamos a detallar cada una de las partes para justificar el presupuesto final:

- Capital humano: Se ha fijado un coste de 25€/h. Por lo tanto, procedemos a calcular el presupuesto relativo a este apartado:
    - 140 días * 2'5 h/día * 25 €/h = 350 horas * 25 €/h = **8750 €**
- Equipo informático: Se ha utilizado un ordenador portátil Acer Aspire 5820TZG, con procesador Intel Pentium P6000 a 1'86 GHz (doble núcleo), 8 GB DDR3 de RAM y Windows 7 Home Premium 64 bits, con 5 años de antigüedad. Ya que todo el trabajo se ha desarrollado digitalmente, el coste de material será el resultado de sumar las horas de encendido (consumo eléctrico) más la amortización relativa al equipo [16].
    - 350 horas de encendido * 0'0085 €/h = **2'98 €** consumo eléctrico
    - 700 € (Precio del equipo) * 26% / año * 140/360 días/año = **70'7 €** amortización
- MATLAB Student: Se aplica el coste de la licencia adquirida para la ejecución del proyecto [17]:
    - **35 €** (licencia individual)

Por lo tanto, el presupuesto final del proyecto es el siguiente:

Concepto	Precio
Capital humano	8750 €
Equipo informático	73'68 €
MATLAB	35 €
TOTAL	**8858'68 €**

7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1. Conclusiones

Podemos decir que se ha cumplido el objetivo principal del proyecto, que era desarrollar una herramienta con la que analizar perfiles de Twitter a través de sistemas capaces de reflejar los cambios en los perfiles, trabajar *on-line* de forma rápida, sin almacenar mucha información: Sistemas Auto-Adaptativos. La parte más decisiva sin duda ha sido el diseño del sistema. Por medio de las decisiones de diseño tomadas y la adaptación técnica en la programación de las mismas se ha logrado conseguir una herramienta útil que nos ha permitido posteriormente un análisis satisfactorio.

La documentación del proyecto ha sido trivial, pues el desarrollo del mismo no puede llevarse a cabo sin un apoyo por escrito de cada una de las fases realizadas.

Aparte de esto, si lo vemos desde un punto de vista más externo, los resultados de las pruebas nos han demostrado que el dominio de datos de perfiles de Twitter no sigue un patrón tan identificativo como para poder realizar una clasificación genérica y contundente. La clusterización individual, por su parte, sí permite obtener una imagen rápida de los perfiles analizados, pero también se ve influida por la propia naturaleza del dominio de los datos.

El fruto del proyecto es bueno, pues se ha conseguido alcanzar los objetivos y mostrar el funcionamiento conjunto de los datos de Twitter y los Sistemas Auto-Adaptativos.

7.2. Trabajos futuros

A partir del desarrollo vigente es posible añadir múltiples funcionalidades a la herramienta para facilitar su uso. En su mayoría son aspectos técnicos que persiguen por un lado hacer más versátil su uso, y por otro aprovechar ésta versatilidad para mejorar la calidad del análisis obtenido. En cualquier caso serían adaptaciones sencillas sobre la herramienta implementada dada la modularidad que ofrece y la documentación existente.

Las funcionalidades adicionales que pueden surgir del estado actual de la herramienta son, entre otras, las siguientes:

- Facilitar la elección de las geolocalizaciones parametrizables en la propia interfaz, prescindiendo de la modificación directa del fichero que las contiene.
- Equilibrar automáticamente las búsquedas realizadas en cada geolocalización para llegar a conseguir un número de perfiles extraídos concreto. Principalmente

consistiría en realizar una cantidad de búsquedas proporcional a la densidad de publicación de cada geolocalización, teniendo en cuenta la palabra de búsqueda seleccionada y la densidad de publicación del resto de geolocalizaciones.

- Parametrizar las constantes (o al menos sus valores iniciales) de los algoritmos utilizados. Actualmente dichos valores están fijados dentro del propio código del algoritmo, por lo que por medio de la herramienta no podemos saber cómo afectaría al resultado del análisis la modificación de los mismos.

En el ámbito de experimentación, también se puede realizar una comparación empírica del eClass0 con otros clasificadores incrementales. El objetivo es medir la efectividad de nuestro clasificador respecto a otros, utilizando los mismos datos en la comparación. De esta manera conseguimos saber cómo de bueno sería nuestro clasificador sin tener en cuenta la influencia de la correlación entre los datos del dominio.

Para adaptar la herramienta a unos requerimientos específicos de análisis, también es posible modificar los datos extraídos de Twitter. Actualmente se toman datos directamente del perfil, pero un análisis que persiga otros factores específicos podría eliminar algunos de los atributos que ya considera la herramienta y añadir otros. La variedad de datos que se pueden recolectar es amplia ya que el API de Twitter ofrece gran cantidad de información sobre perfiles, tweets,…. Por ejemplo, se podría extraer la hora de los últimos 100 tweets que haya publicado el usuario, y así analizar las franjas horarias en las que twittea (el análisis consistiría en detectar las tendencias mayoritarias en franjas horarias de publicación).

Por último, una de las funcionalidades que se planteó al principio del proyecto y que no se ha llegado a implementar es la inclusión de un perfil introducido por el usuario en los modelos generados con los algoritmos de la herramienta. Consistiría en obtener y preprocesar en tiempo real los datos del perfil introducido de igual forma que el resto de datos analizados, y contrastarlo con los resultados del RDE, eClustering y eClass0. De esta manera el usuario puede ver la afinidad del perfil introducido con el análisis vigente. El principal aliciente para el desarrollo de esta funcionalidad es la interacción con el usuario, dando a la herramienta un aspecto más amigable.

8. BIBLIOGRAFÍA

[1] A. J. Sánchez, «Hipertextual,» 21 Febrero 2015. [En línea]. Available: http://hipertextual.com/2015/02/estado-de-las-redes-sociales-2015.

[2] E. Ballesteros, «Multiplicalia Online S.L.,» 23 Febrero 2015. [En línea]. Available: http://www.multiplicalia.com/las-redes-sociales-mas-usadas/.

[3] C. G. Akcora, B. Carminati, E. Ferrari y M. Kantarcioglu, «Detecting anomalies in social network data consumption,» *Social Network Analysis and Mining,* vol. IV, nº 231, August 2014.

[4] Z. Miller, B. Dickinson, W. Deitrick, W. Hu y A. H. Wang, «Twitter spammer detection using data stream clustering,» *Information Sciences: an International Journal,* vol. 260, pp. 64-73, March, 2014.

[5] C. Yang, R. Harkreader y G. Gu, «Empirical Evaluation and New Design for Fighting,» *Information Forensics and Security, IEEE Transactions on,* vol. 8, nº 8, Aug. 2013.

[6] «Twitalyzer,» [En línea]. Available: http://www.twitalyzer.com/5/index.asp.

[7] «Xefer,» [En línea]. Available: http://xefer.com/twitter/.

[8] P. Angelov, Evolving Rule-based Models: A Tool for Design of Flexible Adaptive Systems, New York: Springer-Verlag, 2002.

[9] P. Angelov y X. Zhou, «Evolving Fuzzy-Rule-Based Classifiers From Data Streams,» *Fuzzy Systems, IEEE Transactions on,* vol. 16, nº 6, pp. 1462 - 1475, Dec. 2008.

[10] R. D. Baruah, P. Angelov y D. Baruah, «Dynamically evolving clustering for data streams,» de *Evolving and Adaptive Intelligent Systems (EAIS), 2014 IEEE Conference on*, Linz, Austria, 2-4 June 2014.

[11] «Twitter4j,» [En línea]. Available: http://twitter4j.org/en/index.html.

[12] «Twitter API v1.1,» [En línea]. Available: https://dev.twitter.com/rest/public.

[13] «Tweetping,» [En línea]. Available: http://tweetping.net/.

[14] «Eclipse 4.4 Luna,» [En línea]. Available: https://eclipse.org/.

[15] «MATLAB R2010a,» [En línea]. Available: http://es.mathworks.com/.

[16] «Agencia Tributaria,» [En línea]. Available: http://www.agenciatributaria.es/AEAT.internet/Inicio_es_ES/_Segmentos_/Empresas_y

_profesionales/Empresarios_individuales_y_profesionales/Rendimientos_de_actividades_economicas_en_el_IRPF/Regimenes_para_determinar_el_rendimiento_de_las_actividades_economic.

[17] T. MathWorks, «New License for MATLAB Student,» [En línea]. Available: https://es.mathworks.com/store/link/products/student/new?s_iid=htb_buy_gtwy_cta3.