UNED at CLEF RepLab 2014: Author Profiling

Jacinto Jesús Mena Lomeña and Fernando López Ostenero
jmena48@alumno.uned.es
flopez@lsi.uned.es

UNED NLP & IR Group
Juan del Rosal, 16
28040 Madrid, Spain
http://nlp.uned.es

Abstract. This paper describes a learning system developed for the RepLab 2014 author profiling task at UNED. The system uses a voting model, which employs a small set of features based mainly on the tweet text information such as POS tags, number of hashtags or number of links. In the unofficial run, the feature set was increased with Twitter metadata such as number of followers or retweet speed. The system achieved good results in author categorisation, although its performance in author ranking was low.

1 Introduction

This paper describes the participation of UNED in RepLab 2014 where we tackled the author profiling task focused on classifying and ranking Twitter profiles using their tweet streams.

Twitter constitutes one of the main sources of data relevant for online reputation management because of the spontaneity and immediacy. Although not all the tweets have the same impact. The way in which a post may affect the reputation of a company often depends on who published it. The author profiling task aims at classifying authors by type of their activity and identifying the influential ones, those whose tweets are more likely to propagate quickly and widely through the network and to produce a greater effect. So the final goal is to build a ranking list of the selected Twitter profiles.

The paper is organised as follows. The applied approach is introduced in Section 2 briefly describing the features considered and the learning process. Sections 3 explains the configurations of the model for author categorisation and author ranking. In Section 4, we report the results obtained for each subtask. Finally, in Section 5, we conclude and outline possible improvements of the system in the future.

2 Proposed Approach

2.1 Features

The model uses the following set of features:

Bag of Words: a feature set based on a Weka filter called StringtoWordVector was built. It contains a vector of occurrences of words in a document. We used the default configuration of this Weka filter.

This feature is important to determine the most important words which decide the classification of Twitter profiles in the Author Categorisation subtask. This feature could be more discriminant if it is used taking in consideration the domain information to divide the classification algorithm.

Number of sentences: The system used GATE [1] with the SentenceSplitter resource to get a feature with the number of sentences. We used a specific SentenceSplitter for each language, one for English and other one for Spanish.

POS information: Seven features were built based on the POS tags. We used the GATE POS Tagger with the OpenNLP framework and different models for each language. Before running the POS tagging, we preprocessed the tweet contents to remove hashtags, mentions, and URLs, using regular expressions. After getting the POS tags, we considered a set of the following features that exploit the number of adverbs, verbs, adjectives, nouns, pronouns, foreign words, and abbreviations. This follows the previous work by [7] where the number of POS elements were considered for measuring polarity. These features, in our opinion, could characterise the author's writing style and could be useful useful in author categorisation.

Number of links: We have built a regular expression method to count the number of links in the tweet.

Similar to the point above, we consider this feature useful for the author categorisation subtask, because it reflects stylistic characteristics of the user's writing.

Number of hashtags: Following [3,4], we included a process based on regular expressions to count the number of hashtags.

The hypothesis is that the number of hashtags could be indicative of the relevancy of a tweet, as the more hashtags there are, the more topics will be involved.

Number of mentions: Again, based on the work in [3], we included the count the number of explicit mentions of users of the form user.

For instance, for the following tweet it would be generated the value of 6 mentions for this feature:

```
   still waiting on @MeganBerry's #fbumpf contribution :)
kevinGEEdavis @MerlinUWard @MimiOrtega @jeremarketer @AmyVernon
@IAmMrSid
```

Number of smileys: The system considered the number of smileys, based on the experience of [2]. In order to count smileys, we manually built a dictionary using information extracted from Wikipedia.

```
   Buenos días :) A por un fin de semana increíble lleno de color
amigs ;) http://ow.ly/i/2EXp7
```

Language: We used the language label provided by the Replab 2014 organisers as a feature of the classifier.

This feature is used mainly to determine the set of words to be considered as Bag Of Words.

In the unofficial run, we included two new features, based on Twitter metadata. For that, we used Twitter4J, a Java Wrapper for Twitter REST API. We built the following new features:

Number of followers: For each profile, we queried Twitter about the number of followers of every profile in the training and test data sets.

The idea was to use this feature in the Author Ranking substask to generate weight values the application of which is described below.

Retweet speed: We examined the last retweet of each author. The retweet speed was calculated as follows using the creation date, number of retweets and the creation date of the last retweet:

$$avgTime = \frac{(LastRTCreationTime - TweetCreationTime)}{NumberOfRT} \tag{1}$$

In order to sort elements, we built a weight measure which was calculated using the following formula:

$$weight = \frac{NumberofFollowers}{AverageRTspeed} \tag{2}$$

This formula tries to relate the retweet speed with the number of followers. The aim is to capture those cases when, given two profiles, for instance, one with 1,500 followers and the other with 1,600, the former has more activity in terms of tweets propagation and retweet speed than the latter. So the underlying hypothesis is that it is more relevant a profile with a smaller number of followers and higher speed, than a profile with a bigger number of followers and lower speed. One run was configured with this weight parameter. Regarding this feature, the bigger the weight value is, the more important is a profile.

Due to Rate limiting, we only managed to obtain retweet speed information for about 50% of profiles. In order to use it as a feature, an empty value for the feature was taken to build the classifier for the Author Category subtask. For Author Ranking, an average speed was assigned, multiplied by the number of followers.

2.2 Learning Process and Confidence Methods

The learning process of our system is composed of a voting system, a set of classifiers and a method to resolve the ties by means of confidence scores.

We divided the training data set into 5 subsets, each containing 20% of data. 601 tweets provided by the organisers with each profile were also split in five parts. The classifiers were trained considering each tweet as an instance instead of grouping all the data related to one profile in one instance. Four of the subsets were used to train the system employing the following Weka algorithms:

- ZeroR Algorithm
- RandomTree Algorithm [5]
- RandomForests Algorithm [8,6]
- Nave Bayes Algorithm

These four algorithms allowed covering 80% of the data set. The remaining 20% was used to create a confidence score table.

That training set partition had nearly 300,000 tweets. We iterate tweet per tweet and stored (in a relational database) 4 rows per each tweet as confidence information. As result of that we had a table with close to 1,200,000 (per each Replab 2014 subtask) rows to query information about confidence. The following formula was used to solve those cases when at least three classifiers decided the same:

$$confidence(cat, algs) = \sum\nolimits_{alg \in algs} \frac{nRightClassification(cat, alg)}{nClassifications(cat, alg)} \tag{3}$$

Where *cat* is the category for which the confidence value has to be calculated and *algs* is a set of algorithms the result of which was the category *cat*. $nRighClassification$ is a function with the number of correct classifications for this category produced by this algorithm, and $nClassifications$ is a function which counts the number of classifications for that category.

The confidence scores are used to decide which category is more plausible after training. Figure 1 reproduces the architecture of the confidence score component. This figure shows how the confidence scores table is populated with the outcomes of the algorithms, based on the training data.

Figure 2 illustrates how the confidence score information is used to disambiguate the results and decide which class value should be assigned to a profile.

3 Algorithms

In this section, we describe the algorithm configurations. Table 1 provides an overview of the Author Categorisation algorithms, specifying the kind of data used in each of them. "_AC" in the runs identifiers indicates the "Author Categorisation task", while "_AR" stands for "Author Ranking".

3.1 Author Categorisation

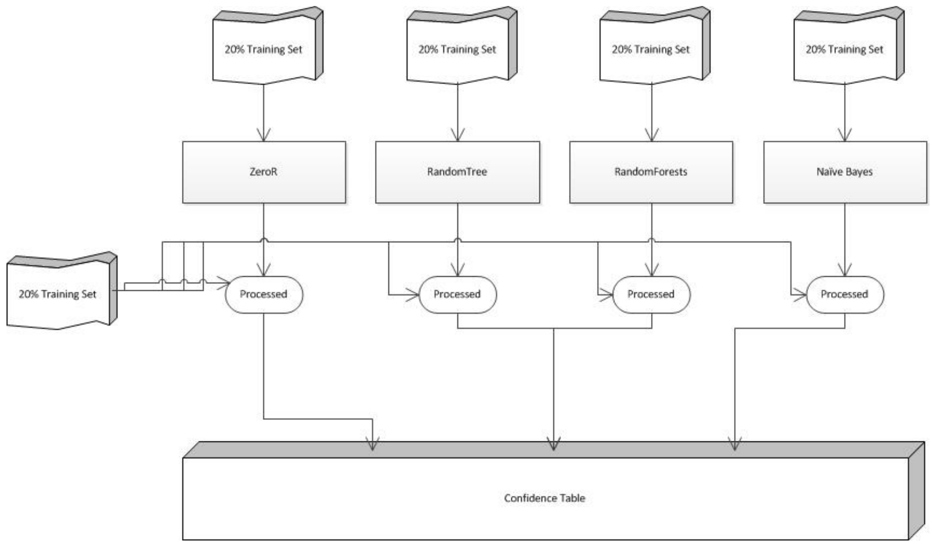

Fig. 1. Confidence Score Component

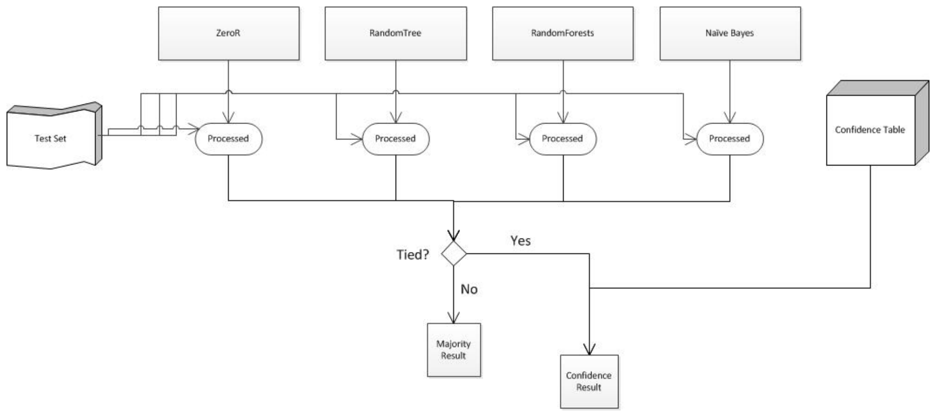

Fig. 2. Using the Confidence Scores

Table 1. Runs submitted for the Author Profiling task.

System ID	# Classifiers	Characteristics
ORM_UNED_AC_1	4	basic configuration
ORM_UNED_AC_2	8	domain information
ORM_UNED_AC_3	8	confidence scores
ORM_UNED_AC_4	8	confidence scores + Twitter info
ORM_UNED_AR_3	1	followers + retweet speed

Basic configuration This is the first and the simplest system configuration (ORM_UNED_AC_1) for author categorisation that consists only of classification algorithms without taking into account information about the domain. The 4 classifiers were fed with a small set of features which included BoW, POS, hashtags, mentions, links, smileys, and language. The classification result was obtained by applying a basic voting algorithm using majority rule.

In order to avoid the bias towards the most frequent class (*Undecidable*), a threshold was applied. The majority class label (*Undecidable*) was assigned only if it was supported by 80% or more votes. Below that threshold (80%), we classified the profile as another majority class, distinct from *Undecidable*.

We used 4 classifiers which classified a profile tweet by tweet. For each profile, we generated 4 class values per tweet, producing near 2400 class values per profile. This information was used to obtain the majority result of the voting algorithm.

Basic configuration with domain features This configuration (ORM_UNED_AC_2) includes information about the profile domain. Algorithms were defined to consider the domain element and decide which algorithm should be used. The same set of features as in the basic configuration, although choosing different classifiers depending on the domain.

As before, we used a threshold to avoid the bias towards the most frequent class (*Undecidable*), setting it at the same value. This configuration produced 8 classifiers.

Confidence scores model This configuration used information about confidence of classifiers algorithms when their results are close to a tie. We submitted the results of this configuration as ORM_UNED_AC_3.

The confidence information was used to decide the outcome of the classification. In case of a tie, we calculated confidence scores using the equation 3. We used the same feature set and threshold as in the basic configuration.

Confidence scores and social information model We built a last configuration using a new kind of information, social information (ORM_UNED_AC_4) after the official deadline for submitting results.

This configuration, for which we can report an unofficial result, is similar to the simple confidence score model described above, but using two new features: number of followers and retweet speed.

We applied to the annotations with the *Undecidable* class the same threshold as in the basic configuration.

3.2 Author Ranking

For the Author Ranking subtask, we submitted one official run: ORM_UNED_AR_3 (see Table 1). The developed algorithm is described below.

Basic configuration We used the following features:

– Class value of opinion_maker/non_opinion_maker
– Number of followers
– Retweet speed

The weight function defined in Equation 2 was used to sort the ranking results.

4 Results

Table 2 reports the scores obtained in the author categorisation subtask *Average Accuracy* and $F_1(R,S)$. The results of the SVM baseline provided by the organisers are also included for reference.

Table 2. Results of the runs submitted to the Author Categorisation subtask.

Run	*AverageAccuracy*	$F_1(R,S)$	**Rank**
SVM baseline	0.461	0.368	2
ORM_UNED_AC_1	0.392	0.323	6
ORM_UNED_AC_3	0.391	0.325	8
ORM_UNED_AC_2	0.371	0.356	11

The test set contained three domains. We employed two domains and in order to assign a value to the third class, we selected one of the classifiers built using the training dataset. Tables 3, 4, 5 report the scores obtained for the evaluation metrics used in the author category subtask: *Reliability (R)*, *Sensitivity (R)* and $F_1(R,S)$ for each domain. For the automotive and banking domains we also include scores of the baselines for reference.

Table 3. Algorithms submitted for the author category subtask: automotive domain.

	automotive		
System	Reliability	Sensitivity	F-measure
`Baseline-SVM`	0.275	0.531	0.362
`ORM_UNED_AC_3`	0.248	0.557	0.343
`ORM_UNED_AC_2`	0.241	0.582	0.341
`ORM_UNED_AC_1`	0.256	0.409	0.314

Table 4. Algorithms submitted for the author category subtask: banking domain.

	banking		
System	Reliability	Sensitivity	F-measure
Baseline-SVM	0.295	0.508	0.373
ORM_UNED_AC_3	0.256	0.658	0.369
ORM_UNED_AC_2	0.282	0.340	0.308
ORM_UNED_AC_1	0.289	0.389	0.332

Table 5. Algorithms submitted for the author category subtask: miscellaneous domain.

	miscellaneous		
System	Reliability	Sensitivity	F-measure
ORM_UNED_AC_3	0.465	0.361	0.407
ORM_UNED_AC_2	0.437	0.417	0.427
ORM_UNED_AC_1	0.423	0.527	0.469

4.1 Author Ranking Results

Table 6 reports the *Average Accuracy* obtained in the Author Ranking subtask evaluation, the position in the official RepLab 2014 ranking. For reference, it also includes the baseline provided by the organisers that uses the number of followers.

Table 6. Results of the system submitted for the Author Ranking subtask.

System	**Average Accuracy**	**Rank**
Followers	0.378	14
ORM_UNED_AR_3	0.349	15

5 Conclusions and Future Work

We described the algorithms submitted to the RepLab 2014 Author Profiling task, where we tackled both author categorisation and author ranking.

Author categorisation was our main focus at RepLab 2014. We submitted three official and one unofficial run. Our proposal was based on a voting system featuring a method to calculate confidence scores to solve ties in votes. However, the results obtained with the confidence method were not as good as we expected, as they were surpassed by the basic configuration. Nevertheless, although the confidence method got the worst results in *Average Accuracy*, it turned out the best in *F-measure* not only among our runs, but also considering the rest of the Author Categorisation task participants.

Future work in author categorisation is going to focus on selecting new features and improving on the whole system in order to make processing more efficient. Furthermore, we will have to refine the confidence formula to avoid setting a threshold for the majority "Undecidable" class.

Regarding author ranking, the bad results can be partly explained by the lack of information for building the ranking. Due to the Twitter Rate Limit, we failed in getting necessary information about the followers and retweet speed for all the profiles. So in case of profiles without this information, they were assigned an average value. This distortion might have affected the system's outcome.

For author ranking, future work will focus on getting more information from Twitter, although the first step, of course, will be to improve the query process to cope with the Twitter Rate Limit.

References

1. Cunningham, H., Maynard, D., Bontcheva, K.: Text processing with gate. Gateway Press CA (2011)

2. Filgueiras, J., Amir, S.: Popstar at replab 2013: Polarity for reputation classification
3. Greenwood, M.A., Aswani, N., Bontcheva, K.: Reputation profiling with gate. In: CLEF (Online Working Notes/Labs/Workshop) (2012)
4. Martın, T., Spina, D., Amigó, E., Gonzalo, J.: Uned at replab 2012: Monitoring task
5. Meina, M., Brodzinska, K., Celmer, B., Czoków, M., Patera, M., Pezacki, J., Wilk, M.: Ensemble-based classification for author profiling using various features
6. Mosquera, A., Fernández, J., Gómez, J.M., Martınez-Barco, P., Moreda, P.: Dlsi-volvam at replab 2013: Polarity classification on twitter data. In: Working Notes of CLEF 2013 Evaluation Labs and Workshop (2013)
7. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Foundations and trends in information retrieval 2(1-2), 1–135 (2008)
8. Saleiro, P., Rei, L., Pasquali, A., Soares, C., Teixeira, J., Pinto, F., Nozari, M., Félix, C., Strecht, P.: Popstar at replab 2013: Name ambiguity resolution on twitter