SPANISH LITERATURE WORK CLASSIFICATION

Sergi Liesegang, Lucas Rodés

{sergi.liesegang, lucas.rodes}@alu-etsetb.upc.edu
BarcelonaTech - Universitat Politcnica de Catalunya (UPC)

ABSTRACT

Classification of literary works according to their author is a field of interest within the text processing area. In particular, this practice becomes relevant for categorizing anonymous works or for author identity verification. Thereby, for this cases, the use of large databases of labelled works, together with the appropriate machine learning algorithms, can help determining the writer of unknown samples.

Index Terms— Literature, Text classification, Natural Language Processing, Machine Learning, Naive Bayes, Support Vector Machine

1. INTRODUCTION

In this work we provide a brief review of the theoretic framework of text classifiers. We aim to implement a software based algorithm capable of classifying spanish literature work fragments to their corresponding authors. In this regard we have built our own training and test sets based on the content provided in [1]. Similar works [2] have been carried out, only considering English language and the use of already prepared training sets. This way, the relevance of our project emerges as it presents a new scenario with a different and interesting challenges.

Our work has focussed on preparing suitable training and test corpus in a Linux environment, which required an exhaustive usage of Bash language. Furthermore, the classification algorithmic was implemented using Python, specifically the Scikit-Learn [3] and NLTK [4] libraries indeed. Thus, all these tools have been crucial for the development of this project. Besides, external computational resources, i.e. TALP servers, have been utilized to achieve this goal rapidly and efficiently.

1.1. Contributions

- Search, selection and preparation (using Bash programming) of Spanish literature work training and test datasets.
- Performance and evaluation comparison between different classification techniques in the context of Spanish literature (Using Python programming).

1.2. Organization

This work is organized as follows: Section 2 gives the sources that have been used as corpus for this project. It further explains how samples have been distributed among the training and test datasets. Section 3 introduces the classifiers that have been employed and details briefly some of its theoretic fundamentals. Section 4 provides some of the results obtained and their interpretation. Finally, Section 5 ends this report with some conclusions and possible future improvements.

2. CORPUS

As aforementioned, we did not find neither prepared training nor prepared test corpus adjusted to our needs. Hence, we had to construct these sets ourselves. To do so, we used the pdf resources available in 2, which content is presented in Table 1. The selected eight authors are well recognized figures of the classical Spanish Literature for their significant and important works. However, given the relation between some authors of the same era, correlation between works can difficult the classification (i.e. separation might result difficult as a result of their similarity). Moreover, note that other subjects can also be used for this objective, regardless of their epoch or genre.

Author	Epoch	Genre
Miguel de Cervantes	C16th	Novel - Poetry - Drama
P. Calderón de la Barca	C17th	Drama - Poetry
Felix Lope de Vega	C17th	Novel - Poetry - Drama
Francisco de Quevedo	C17th	Poetry
Gustavo A. Bécquer	C19th	Poetry - Novel
Emilia Pardo Bazán	C19th	Novel
Benito Pérez Galdós	C19th	Novel - Drama
Federico García Lorca	C20th	Poetry - Drama

Table 1. Authors Set

3. CLASSIFIERS

Two main classification approaches have been followed:

- Machine Learning Classifier, which includes Naive Bayes Classifier and Support Vector Machine (SVM) Classifier.
- Natural Language Classifier

For each of the two approaches, an explanation of the training/test datasets generation process is provided.

3.1. Machine Learning Classifiers

At first we used simple text classifiers such as Naive Bayes and SVM based on word occurrences. Both techniques have been studied and compared in the field of text categorization [5].

3.1.1. Datasets

Given the nature of the dataset, some processing was required to make the data tractable for our software implementation.

1. **Download books**: We downloaded B books per each of the authors. That is, we had a total amount of $8B$ books. For this work, we have used $B = 15$.
2. **Split Training/Test sets**: We decided to use 66% and 33% of the books for the training and test corpus, respectively. Remember that proper trainings need large sets of data and the system evaluation only requires a smaller portion. We chose these percentage values since we did not have too much data. Typically, training corpus tends to have a higher proportion of the total data.
3. **From books to chunks**: For all the books, both in training and test sets, we divided them into blocks of L lines. Note, that due to the nature of the writing style of each author, the number of samples oscillated from author to author.
4. **Labelling**: Once we have the set of samples per author (blocks of L lines), we labelled them accordingly.

Fig. 1 illustrates this four data processing steps to prepare the training and test sets. As aforementioned, for this part we have used Bash programming.
Finally, we trained the classifier using the supervised methods (Naive Bayes and SVM).

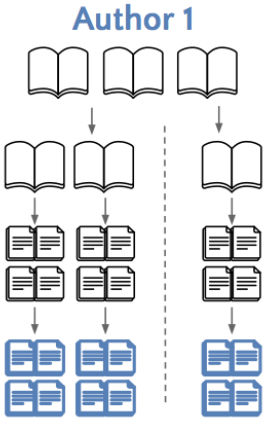

Fig. 1. Machine Learning Classifier: Preparing the training and test data

3.1.2. Naive Bayes Classifier

The Naive Bayes Classifier is the mainstay in text processing given its simplicity when categorizing samples. In particular, it is based on the so-called Bag of Words model [6], which compares the different elements text contain. For instance, a scientific text might include technical vocabulary in comparison with a literary one. Thanks to that, we are able to distinguish them, always relying on the Bayes Formula (1). In fact, the idea here is to evaluate each observation X (fragment of the text converted to occurrences matrix) with respect to the posterior probability of the class C_k (author k in our case) so that the writer leading to its higher value is selected. In other words, a histogram of words from a certain dictionary is created to be later studied and used to decide the most likely class.

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)} \tag{1}$$

3.1.3. Support Vector Machine

Similarly to the Naive Bayes case, here we also use word occurrences as input features. Nevertheless, this well-known classifier tries to separate samples in a space representation, i.e. classify inputs through a decision boundary that ensures maximum class separability. Actually, the type of border can be different depending on the situation. The simplest one is the linear separation, also called linear kernel, used when classes are barely mixed. Another approach can be the use of non-linear kernels, such as the high order polynomial or the Gaussian, which allow a better segmentation of the feature space and usually results in a better separability. The interested reader might refer to [7]. However, once the different possibilities were studied, we realized that no significant improvement was achieved when varying the kernel. Therefore, we selected the linear for its simplicity and cost efficiency. To

optimize the input parameters of the SVM classifier, we also used a tool called *Gridsearch* [8], which is provided by the Scikit-Learn library. The performance is considerably ameliorated, as we will explain in Section 4.

3.2. Natural Language Classifier

Natural Language Modelling (NLM) technique was selected, primarily, as a result of the experience we gained in the last assignment using SRILM toolkit [9] as well as its popularity and widely usage in speech processing field.

3.2.1. Datasets

For this classifier, a modification to the previous database division strategy is required. In particular, the training and test will be still independent text, but now only the test will be divided in blocks. Let us list the steps followed, only rewriting those that differ from the ones in the machine learning classifier approach.

1. **Download books**
2. **Split Training/Test sets**
3. **Prepare test samples**
   - 3.1. **From books to chunks**: For all the books, only in the test sets, we divided them into blocks of L lines. Note, that due to the nature of the writing style of each author, the number of samples oscillated from author to author.
   - 3.2. **Labelling**: Label all the test samples.
4. **Prepare training samples**
   - 4.1. **From many books to one book**: Merge all books into a single one.
   - 4.2. **Language model generation & labelling**: Generate a language model per each author.

Fig. 2 illustrates the data processing steps to prepare the training and test sets. As in the previous approach, we also have used Bash programming. Besides, we will see how perplexity, main output of this technique, is used to classify correctly the authors.

3.2.2. Classifier

Once we have a model for each author (with their corresponding set of texts from the training database as main input), the differences between them can be easily obtained. In particular, by means of the perplexity we are able to estimate which writer does a certain work belong to. In fact, perplexity provides us a measure of similarity between texts, regarding

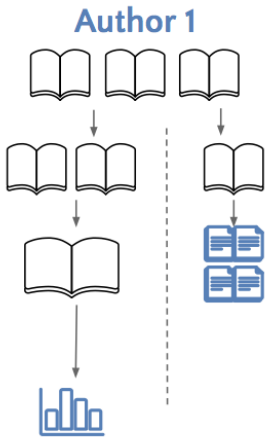

Fig. 2. Natural Language Classifier: Preparing the training and test data

their words and structure. Therefore, when two texts are sufficiently close to each other, the perplexity measure tends to be low (and vice-versa: different works lead to a high values). Note, then, that the NLM are not actually classifying samples but we are using its output to do it (so-called *perplexity based classifier*). In other words, since we create each model out from several texts from the same author, when evaluating such model with a work from the same writer, we will obtain a low perplexity and a higher otherwise. Fig. 3 illustrates the evaluation of the trained classifier using some samples from the test set.

Moreover, in order to create the models, we rely on the N-gram model [10], only considering the cases: unigram, bigram and trigram. Besides, a smoothing method [11] is also introduced to avoid combination of words with a zero probability. Particularly, we have used the *Add-one* smoothing method due to its simplicity and computational low cost. Moreover, using the NLTK library it is very simple to work with this smoothing method, contrarily to other smoothing methods.

Fig. 3. Testing the classifier

4. RESULTS

In order to evaluate the performance of the whole system, the measures of Precision (2) and Recall (3) were used. Partic-

ularly, they reflect the accuracy of each approach by means of the relation between true positives with false positives and with false negatives (respectively). However, so as to provide a more compact measure, the F-Score (4) is computed, which in turn takes into account the previous values. Actually, we will use this score to show the response of each classifier and its variants.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (2)$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (3)$$

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

Thereby, for the classifiers seen before, we will represent their F-Score with respect to the size of the chunk (number of lines N). Note that for the case of NLM, only the test is segmented and that the results for the unigram, bigram and trigram approaches are shown. In addition, for the case of SVM, the behaviour with and without *GridSearch* is provided too. To gain more insights, we have evaluated the classifiers for different chunk sizes in the test set $L = \{10, 25, 50\}$. All these results can be seen in Fig.4.

As we can see, the worst performance is achieved when using the Naive Bayes Classifier, fact that could be expected given its simplicity. It is followed by the SVM approach, which is considerably improved when parameter optimization (*GridSearch*) is carried out. Finally, we can observe that the best option is the use of NLM to categorize correctly the texts. In particular, both unigram and bigram present a similar and satisfactory curve. Nevertheless, the trigram approach results in a notorious degradation of the classifier performance. This might be due to the fact that the training corpus are not long enough and might not contain many trigrams. Thus, the smoothing technique might be overused, dealing to bad estimations. Overall, we can appreciate that the longer the chunk is, the better the performance of the system is. This is due to the fact that larger pieces enable classes to be more separated as they contain more differential traits of each author. For instance, Cervantes and Garcia Lorca might be quite similar when having analysing a single sentence but completely distinct when studying a longer fragment. Thanks to that, we have achieved an accuracy of 87.6% with the bigram model and a length of 50 lines per chunk.

Nonetheless, note that even with the poorest strategy (Bayesian), a performance of 67% is reached. Also, the SVM approach using *GridSearch* leads to an score of 80 %. Hence, we can see that simple methods can have acceptable results. However, there is still a noticeable performance gap between them and NLM technique, as a consequence of the correlation between authors (e.g. Lope de Vega and Calderón de la Barca).

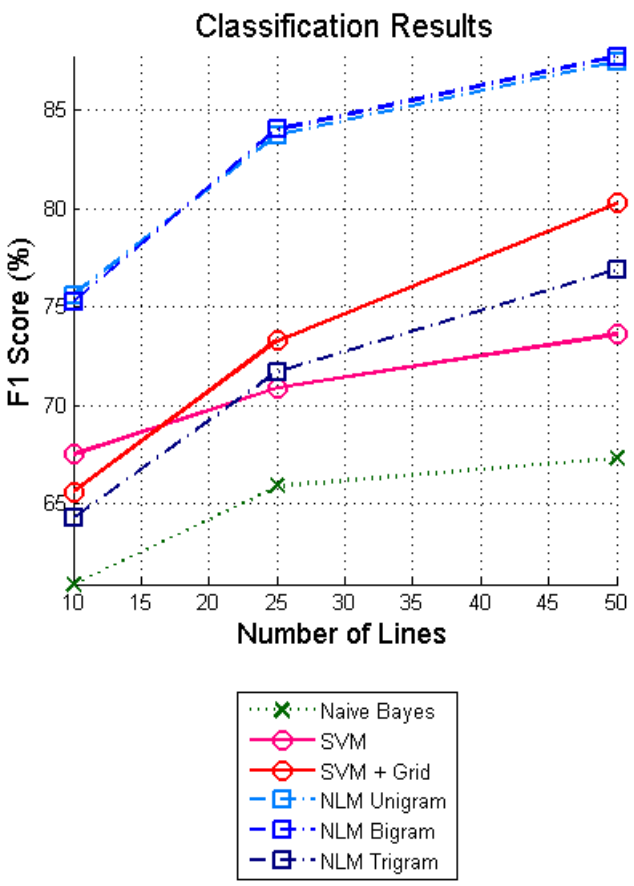

Fig. 4. Classification Results

Actually, only through NLM based classifier this correlation problem is correctly overcome, since models are created independently (not jointly like in SVM where classes can be mixed) and thus, robustness against input samples correlation is achieved.

5. CONCLUSION AND FUTURE WORK

We can see that NLM is by far the best approach to classify texts to their authors. Perplexity measure seems to be quite a good metric when categorizing text sources. Furthermore, longer blocks of text lead to better results, since the writer divergence tends to increase when larger works are compared. Finally, correlation represents an issue in simple approaches, like Bayes or SVM.

On the other hand, there are some possible aspects that could be done in the future to improve this work. In particular, the database for all authors could be increased, both for training and test sets. Also, a deeper analysis of the relation between chunk size and accuracy would be a good practice too, such

as a performance convergence. Besides, given the differences between the writers genres, the length of their works are not equal, which leads to different authors having more samples than others. Thus, it could be of interest to equilibrate the number of samples per author. In addition, other and more sophisticated smoothing methods for the NLM. Last but not least, the use of neural networks could have a prominent impact.

6. ACKNOWLEDGEMENT

We want to thank Prof. Jose A. Fonollosa for allowing us to have access to the servers of TALP Research Group.

7. REFERENCES

[1] `http://www.edu.mec.gub.uy/`, "Lectura solidaria," [Online accesed on 19-May-2016].

[2] Caitlin Colgrove, Sheldon Chang, and Phumchanit Yiam Watanaprakornkul, "Literary period classification," 2010.

[3] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al., "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] C Joakim, "Explore python, machine learning, and the nltk library," *IBM Developer Works*, 2012.

[5] Sundus Hassan, Muhammad Rafi, and Muhammad Shahid Shaikh, "Comparing svm and naive bayes classifiers for text categorization with wikitology as knowledge enrichment," in *Multitopic Conference (INMIC), 2011 IEEE 14th International*. IEEE, 2011, pp. 31–34.

[6] Yin Zhang, Rong Jin, and Zhi-Hua Zhou, "Understanding bag-of-words model: a statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1-4, pp. 43–52, 2010.

[7] Andrew Ng, "Cs229 lecture notes," *CS229 Lecture notes*, vol. 1, no. 1, pp. 1–3, 2000.

[8] `http://scikit-learn.org/stable/modules/grid_search.html`, "3.2. grid search: Searching for estimator parameters," [Online accesed on 4-June-2016].

[9] Andreas Stolcke et al., "Srilm-an extensible language modeling toolkit.," in *INTERSPEECH*, 2002, vol. 2002, p. 2002.

[10] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[11] Stanley F Chen and Joshua Goodman, "An empirical study of smoothing techniques for language modeling," in *Proceedings of the 34th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1996, pp. 310–318.