GTH-UPM at TASS 2019: Sentiment Analysis of Tweets for Spanish Variants

Ignacio González Godino and Luis Fernando D'Haro

Grupo de Tecnología del Habla, ETSI de Telecomunicación
Universidad Politécnica de Madrid
Avenida Complutense 30, 28040, Madrid, Spain
ignacio.ggodino@alumnos.upm.es, luisfernando.dharo@upm.es

Abstract. This article describes the system developed by the Grupo de Tecnología del Habla at Universidad Politécnica de Madrid, Spain (GTH-UPM) for the competition on sentiment analysis in tweets: TASS 2019. The developed system consisted of three classifiers: a) a system based on feature vectors extracted from the tweets, b) a neural-based classifier using FastText, and c) a deep neural network classifier using contextual vector embeddings created using BERT. Finally, the averaged probabilities of the three classifiers were calculated to get the final score. The final system obtained an averaged F1 of 48.0% and 48.4% for the dev set on the mono and cross tasks respectively, 46.0% and 45.0% for the mono and cross tasks for the test set.

Keywords: TASS · Multiclassifiers · Natural Language Processing NLP · Twitter · Sentiment Analysis

1 Introduction

Sentiment Analysis (a.k.a opinion mining) is a branch of the Natural Language Processing field whose goal is to automatically determine whether a piece of text can be considered as positive, negative, neutral or none, deriving this way the opinion or attitude of the person writing the text [13]. Sentiment analysis has recently brought a lot of attention since it can be used for companies to understand customers' feelings towards their products [18], for politicians to poll statements and actions (even to predict the results of an election [2]), or it can be also used to monitor and analyze social phenomena and general mood.

For TASS 2019 competition, the organizers proposed to research on sentiment analysis with a special interest on evaluating polarity classification of tweets written in Spanish variants (i.e. Spanish language spoken in Costa Rica, Spain, Peru, Uruguay and Mexico). The main challenges the system must face up were the lack of context (tweets are short, up to 240 characters), presence of informal language such as misspellings, onomatopeias, emojis, hashtags, usernames,

etc., similarities between variants, classes imbalance, but specially restrictions imposed by the organizers on the data used for training [5].

The proposed challenge consisted of two sub-tasks: a) Monolingual: where participants must train and test their systems using only the dataset for the corresponding variant, and b) Cross-lingual: where participants must train their systems on a selection of the complementary given datasets while using the corresponding variant for testing; the goal here was to test the dependency of systems on learning specific characteristics of the text for a given variant. For both tasks, the challenge organizers asked participants that in case submitting a supervised or semi-supervised system, it must be only trained with the provided training data being totally forbidden to use other training sets. However, linguistic resources like lexicons, vectors of word embeddings or knowledge bases could be used by clearly indicating them. The goal here was to have fair comparison between systems but also to furtherance creativity by restricting system to only use the same set of training data.

The paper is distributed as follows. In Section 2 we provided detailed information about the datasets given by the organizers; afterwards, in Section 3 we describe in detail the classifiers and features used in our system; then, in Section 4, we present our results on the monolingual and cross-lingual settings. Finally, in Section 5 we present our conclusions and future work.

2 Corpus description

The organizers provided participants with a corpus including five sets of data, for five different countries where Spanish is spoken, which are: Costa Rica (CR), Spain (ES), Mexico (MX), Peru (PE), and Uruguay (UY). For each variant, training, development and test sets were provided. The data was composed by several tweets, their ID, user, date, variant and, only in training and development sets, the sentiment class, which could be 'P' (positive), 'N' (negative), 'NEU' (neutral) or 'NONE' (no sentiment).

The label distribution for each variant for the training and development sets is shown in Table 1. Table 2 shows the label distribution for the test set. As we can see, the distribution of labels among variants are different showing systems must deal with class imbalance; however, the label distribution between the training, dev and test sets for the same variant are quick similar (except for Peru where there are more Neg, Neu and None for the test set than for training and dev). This posed the challenge to create a robust system, but also explains the difference in performance for this variant as shown in Section 4.

The task was divided in two sub-tasks, monolingual and cross-lingual analysis. In the first sub-task, the systems used tweets from the same variant for both training and testing. In the cross-lingual setting, in order to test the dependency of systems on a variant, they could be trained in a selection of any variant except the one which was used to test. In our case, we just combined the other variants into a single file and evaluate on the corresponding dev or test set for the given variant.

Table 1. Data and labels distribution for the training and development sets

Label	NEG		NEU		NONE		POS		Total	
Variant	**Train**	**Dev**	**Train**	**Dev**	**Train**	**Dev**	**Train**	**Dev**	**Train**	**Dev**
CR	310	143	91	55	155	72	221	120	777	390
ES	474	266	140	83	157	64	354	168	1125	581
MX	505	252	79	51	93	48	312	159	989	510
PE	228	107	170	56	352	230	216	105	966	498
UY	367	192	192	90	94	51	290	153	943	486

Table 2. Data and labels distribution for the test set

Variant	NEG	NEU	NONE	POS	TOTAL
CR	459	151	220	336	1166
ES	663	195	254	594	1706
MX	745	119	111	525	1500
PE	485	368	176	435	1464
UY	587	290	82	469	1428

After reading the data, each tweet was pre-processed as follows:

- Leading and trailing spaces were removed
- Words starting by the symbol "#" were replaced by just keeping the word and removing the "#". If camel case was found, the word was separated. For instance, "#thisBeautifulDay" was replaced by "this Beautiful Day".
- Url references ('http://...') were replaced by the word 'http'.
- User references ('@username') were replaced by the word USER_NAME.
- Sequences of three or more equal characters were replaced by a single occurrence of that character. For instance, "siiii" was replaced by "si".
- References to human laughter, as "jajja" or "jajajajaj" were replaced by "jajaja".
- Numbers were removed.
- Other punctuation symbols were removed.

Then, we performed lemmatization and tokenization using the large Spanish model included in SpaCy[1]. Finally, tweets were converted to lowercase.

During this pre-processing phase, we analyzed the tweets and discovered a remarkably high percentage of Out-Of-Vocabulary words (OOV's) by comparing the vocabularies from the training and development sets, with values ranging between 53-55%. We managed to reduce it up to 48-51% by using lemmatization and character vector embeddings (see Section 3.2), but it was still a surprisingly high value, which reduced the performance of our final system leaving it as a future work to find additional solutions to this problem.

The resulting pre-processed tweets were then used as input for the three classifiers mentioned in the following section.

[1] https://spacy.io/

3 Classifiers

The final system was based on three different and independent classifiers followed by an ensembling method, which are explained below.

3.1 Feature-based classifier

The first classifier was based on features extracted from the training tweets[2]. Then, we concatenated them and trained a classifier for each variant. The extracted features were:

- Number of words in the tweet.
- The number of words with all characters in upper case.
- Number of "hashtags" found in the tweet (i.e. words starting with symbol "#").
- Whether the tweet has an exclamation mark.
- Whether the tweet has a question mark.
- Presence or absence of words with one character repeated more than two times, as "holaaa".

These features were selected based on features commonly used in sentiment analysis [1] or [3], and our intuition. For instance, we can intuitively expect that longer tweets tend to be more negative in order to explain the sorrow situation, or that upper case words usually have more importance and tend to be used in highly sentimental tweets. On the other hand, when analyzing the corpus, we noticed than tweets containing exclamation marks, "hashtags" or words like "holaaa" are more likely to be positive, and tweets containing exclamation marks tend to be non-emotional; therefore many of the proposed features were extracted based on our initial analysis and intuition.

Besides, a negative and positive vocabulary was automatically created from the training data extracting the 25 most discriminating words between the classes 'P' and 'N' using the algorithm proposed in [11]. Four features were extracted from this vocabulary (checking if these words were in the tweet or not) and normalized by the number of words in the tweet:

- Number of negative words.
- Number of positive words.
- Number of positive words minus the number of negative words.
- Total count of both negative and positive words.

This set of ten features was used for training eight different classifiers: Logistic Regression, Multinomial Naïve Bayes, Decision Tree, Support Vector Machines, Random Forest, Extra Trees, AdaBoost and Gradient Boost. Each one was computed following three different strategies, Normal, One-Vs-Rest and One-Vs-One. All these classifiers were implemented with the scikit-learn[3] tools for Python, and finally we kept the one that obtained the best performance for the corresponding variant on the development set.

[2] Some features were extracted before applying the pre-processing methods.
[3] https://scikit-learn.org

3.2 Fasttext classifier

FastText [10] is an efficient library, created by Facebook's AI Research (FAIR) lab[4] that allows learning n-gram word and sub-word representations (i.e. vector embeddings) using a supervised or unsupervised learning algorithm on a standard multicore-CPU. The library also allows training a multi-class sentence classifier using a simple linear model (multinomial logistic regression) with rank constraint.

In more detail, for the sentence classifier, the library implements a shallow neural network that uses as input features the averaged vector embeddings of the input sentence and a Softmax layer to obtain a probability distribution over the pre-defined classes. Several tricks are implemented such as Hierarchical Softmax [7] and Huffman coding tree [12] to reduce the computational complexity when the number of labels is large; use of bag of n-grams and hashing trick [19] are also implemented to maintain a fast and memory efficient mapping of the learned n-grams. Finally, FastText also deals with the problem of OOVs (Out-of-Vocabulary words) by training bag of n-grams of characters; this capability was one of the main motivations for using FastText as we discovered there was a huge proportion of OOVs between training and dev data during our data analysis (see Section 2).

For our classifier, we first created a set of vector embeddings with dimension 100 using a supervised method trained with the labeled data from previous TASS challenge [6]. The idea was to use those pre-trained vector embeddings as a linguistic resource for the following steps. Initially, instead we tried using the available pre-trained vectors [8] released by FAIR for Spanish[5] but our results were worse probably due to differences in pre-processing, nature of the text (tweets vs formal text), and the reduce number of training data to correctly adapt the 300-dimensional pre-trained vector embeddings. The hyper-parameters we used for this pre-training phase were: learning rate: 1.0, epochs: 5, wordNgrams: 2, dimension: 100. The rest of parameters were the default ones provided by FastText.

Next, we trained 5 independent supervised models for each variant in the competition using the pre-trained vector embeddings as input and the corresponding training data for that variant complying to the established rules by the organizers. Finally, we fine-tuned the model hyper-parameters (learning rate, number of iterations, and size of word n-grams) using the corresponding development set. In general, the only parameter we fine-tuned was the number of epochs ranging from 5 to 10 depending on the amount of training data for each variant.

3.3 BERT classifier

BERT stands for Bidirectional Encoder Representations from Transformers. Proposed by [4], this is a new method of pre-training contextual vector representa-

[4] https://fasttext.cc/
[5] https://fasttext.cc/docs/en/crawl-vectors.html

tions which obtains state-of-the-art results on several Natural Language Processing (NLP) tasks such as text classification, question-answering, labeling tagging and language model prediction.

One of the main advantages of BERT is that their creators have publicly released pre-trained English and Multilingual models, which have been trained on massive corpora of unlabeled data with a new pre-training objective: the "masked language mode" (MLM), inspired by the Cloze task [17]. This change allowed authors to use bidirectional networks instead of the left-to-right networks used in the earlier OpenAi GPT model [15]. Finally, the advantage of using BERT is that the pre-trained models are ready to be fine-tuned for downstream tasks with limited amount of data by using transfer learning approaches [16]. This is done by fine-tuning BERTs final layers while taking advantage of the rich representations of language learned during pre-training.

For our classifier, we used the BERT-Base pre-trained Multilingual Cased model, which was trained on 104 languages and consists of 12 layers (Transformer blocks), 768 hidden units, 12 multi-head attentions, which sum up to 110M parameters. Then, we created 5 different models for each variant by fine-tuning the model using only the training data for the corresponding variant and checking the progress along up to 10 different iterations on the development set with a batch size of 32 samples.

3.4 Averaging Ensemble

We used the three former classifiers to get a distribution of probabilities for each class (i.e. multi-label classification) given the tweet. Then, we implemented a soft-voting ensemble by averaging the three probabilities and classified the tweet with the most likely class. As the feature-based classifier was trained in several different classifiers, we had 24 different results for each variant. Therefore, we selected the classifier that obtained the best performance on the dev set, as shown in Tables 3 and 4.

Table 3. Selected classifiers per variant for the Cross-lingual setting

Variant	Selected Classifier
CR	OneVsRest Logistic Regression
ES	OneVsOne Logistic Regression
MX	Normal Ada Boost
PE	OneVsRest Gradient Boost
UY	Normal Naïve Bayes

4 Results

From the Ensemble explained in the section before, we decided to submit the best performance for each classifier in the development set and the same approach

Table 4. Selected classifiers per variant for the Mono-lingual setting

Variant	Selected Classifier
CR	Normal Ada Boost
ES	Normal Naïve Bayes
MX	OneVsRest Ada Boost
PE	Normal Ada Boost
UY	OneVsRest Ada Boost

for the test set. Our results, for each classifier, are shown in Tables 5 (feature-based), 6 (Fasttext), and 7 (BERT); results for the final ensemble are presented in Table 8.

It is important to mention that, when evaluating on the test set, we used the best hyper-parameters found using the development set and then combined the training and dev sets for training the final classification models; then we evaluated the resulting models on the test set. For the cross-lingual setting we took care, when combining the training and development data, to exclude the corresponding development set for the variant to be tested.

As we can see in the results, the ensemble outperformed most of the times the individual classifiers on the test set for both cross and mono lingual settings. We also found that the feature classifier performed the worst in most of the cases (except for BERT for the Peruvian variant), however we think it provides complementary information as we found when performing the optimizations on the development set.

Table 5. Macro F-1 results for the Feature-based classifier only

	Cross-F1-Score		**Mono-F1-Score**	
	Dev	Test	Dev	Test
CR	0.2979	0.2968	0.3698	0.3823
ES	0.333	0.3670	0.2889	0.2873
MX	0.361	0.3369	0.3697	0.3990
PE	0.317	0.2741	0.349	0.2925
UY	0.296	0.3305	0.3879	0.3846

Table 6. Results for the Fasttext classifier

	Cross-F1-Score		Mono-F1-Score	
	Dev	Test	Dev	Test
CR	0.5431	0.4518	0.4872	0.4595
ES	0.4415	0.4048	0.442	0.4206
MX	0.4455	0.4542	0.4265	0.4557
PE	0.4598	0.4621	0.4839	0.4226
UY	0.448	0.4520	0.489	0.4793

Table 7. Results for the BERT classifier

	Cross-F1-Score		Mono-F1-Score	
	Dev	Test	Dev	Test
CR	0.471	0.4608	0.4362	0.4713
ES	0.4417	0.4680	0.4616	0.4604
MX	0.4374	0.4471	0.296	0.4856
PE	0.4286	0.4641	0.4134	0.0536
UY	0.4826	0.4735	0.4755	0.4555

Table 8. Ensembling classifier results

	Cross-F1-Score		Mono-F1-Score	
	Dev	Test	Dev	Test
CR	0.5156	0.4639	0.4923	0.4678
ES	0.4686	0.3772	0.4849	0.4552
MX	0.4732	0.4706	0.4143	0.4867
PE	0.4555	0.4565	0.4868	0.3987
UY	0.5059	0.4811	0.5193	0.4921

5 Conclusions and Future Work

In this paper we have described the first participation of GTH-UPM for the "Sentiment Analysis at SEPLN" (TASS 2019) at Tweet level. Our final system consisted of an ensemble using average voting of three different multi-label text classifiers: a) feature-based using Scikit-Learn, a shallow neural network using FastText, and a transfer-learning approach using BERT. Our results on the development set provided an averaged F1 score of 45.0% and 46.0% on the test set for the cross- and mono-lingual settings respectively. Our system performed very well when compared with other submitted systems across the two settings showing that our proposal was robust enough.

As future work we are planning to perform a more exhaustive analysis of the results given by the three classifiers, fine tuning models hyper-parameters, and testing new features as the ones proposed in [3] and new pre-trained vector embeddings such ElMO [14] or UlmFit [9].

Acknowledgments

The work leading to these results has been supported by the following projects: AMIC (MINECO, TIN2017-85854-C4-4-R) and CAVIAR (MINECO, TEC2017-84593-C2-1-R). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research.

References

1. Agarwal, A., Xie, B., Vovsha, I., Rambow, O., Passonneau, R.: Sentiment analysis of twitter data. In: Proceedings of the Workshop on Language in Social Media (LSM 2011). pp. 30–38 (2011)
2. Bermingham, A., Smeaton, A.: On using twitter to monitor political sentiment and predict election results. In: Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology. pp. 2–10. SAAIP 2011 (April 2011)
3. Chiruzzo, L., Rosá, A.: Retuyt-inco at tass 2018: Sentiment analysis in spanish variants using neural networks and svm. Proceedings of TASS **2172** (2018)
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
5. Díaz-Galiano, M.C., et al.: Overview of tass 2019. CEUR-WS, Bilbao, Spain (2019)
6. García Cumbreras, M.Á., Martínez Cámara, E., Villena Román, J., García Morera, J.: Tass 2015–the evolution of the spanish opinion mining systems (2016)
7. Goodman, J.: Classes for fast maximum entropy training. arXiv preprint cs/0108006 (2001)
8. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)
9. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146 (2018)
10. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers. pp. 427–431. Association for Computational Linguistics (April 2017)
11. King, G., Lam, P., Roberts, M.E.: Computer-assisted keyword and document set discovery from unstructured text. American Journal of Political Science **61**(4), 971–988 (2017)
12. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
13. Pang, B., Lee, L., et al.: Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval **2**(1–2), 1–135 (2008)
14. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. arXiv preprint arXiv:1802.05365 (2018)
15. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding with unsupervised learning. Tech. rep., Technical report, OpenAI (2018)
16. Ruder, S.: Neural Transfer Learning for Natural Language Processing. Ph.D. thesis, National University of Ireland, Galway (2019)

17. Taylor, W.L.: cloze procedure: A new tool for measuring readability. Journalism Bulletin **30**(4), 415–433 (1953)
18. Vinodhini, G., Chandrasekaran., R.M.: Sentiment analysis and opinion mining: a survey. International Journal **2**(6), 282–292 (2012)
19. Weinberger, K., Dasgupta, A., Attenberg, J., Langford, J., Smola, A.: Feature hashing for large scale multitask learning. arXiv preprint arXiv:0902.2206 (2009)