Aspie96 at NEGES (IberLEF 2019): Negation Cues Detection in Spanish with Character-Level Convolutional RNN and Tokenization

Valentino Giudice[0000−0002−8408−8243]

University of Turin, Italy
valentino.giudice@edu.unito.it

Abstract. This paper describes the model used by the Aspie96 team in the subtask A of NEGES 2019 (part of IberLEF 2019), aimed at the automatic detection of negation cues in Spanish. The approach is based on a neural network which uses character-level features to compute token-level features.

Keywords: negation · NEGES · Spanish · natural language processing · neural network

1 Introduction

In the shared task NEGES, described in [8], and part of IberLEF 2019, two subtasks were proposed: subtask A (negation cues detection) and subtask B (role of negation in sentiment analysis). The Aspie96 team took part in subtask A, which required participants to develop a system capable of identifying the negation cues present in a document. The NEGES task had been previously presented in 2018, as described in [9] (the subtask A of NEGES 2019 corresponds to task 1 in NEGES 2018).

Each negation may have been an individual word, or composed of multiple (even non-contiguous) words. In the provided training and testing datasets, documents were already tokenized. Tokens were words non words (for instance, in the case of punctuation). For each document, PoS-tags and lemmas were also provided.

The data used in the task was from the SFU $\text{Review}_{\text{SP}}$-NEG corpus, presented in [10], consisting of reviews, in Spanish, from 8 different domains (cars, hotels, washing machines, books, cell phones, music, computers and movies).

The main characteristic of the presented model, described in the next section, is that it is not based on word-level features, nor on any kind of knowledge which isn't strictly automatically extracted from the provided training dataset.

2 Method

The model used by the Aspie96 team was based upon the system presented in [4] in the context of IronITA (EVALITA 2018), a shared task for irony detection in tweets in Italian described in [2]: a neural network able to classify short texts without making use of word-level features or pretrained layers.

It had been completely developed from scratch by the author of this paper (and of [4]) for his thesis on the classification of tweets in English and Italian. The thesis, in Italian, is not currently publicly available, but the code originally used is[1].

It constitutes the basis of the system used in the subtask A of NEGES, thus it is crucial to understand it first. A representation is provided in Figure 1.

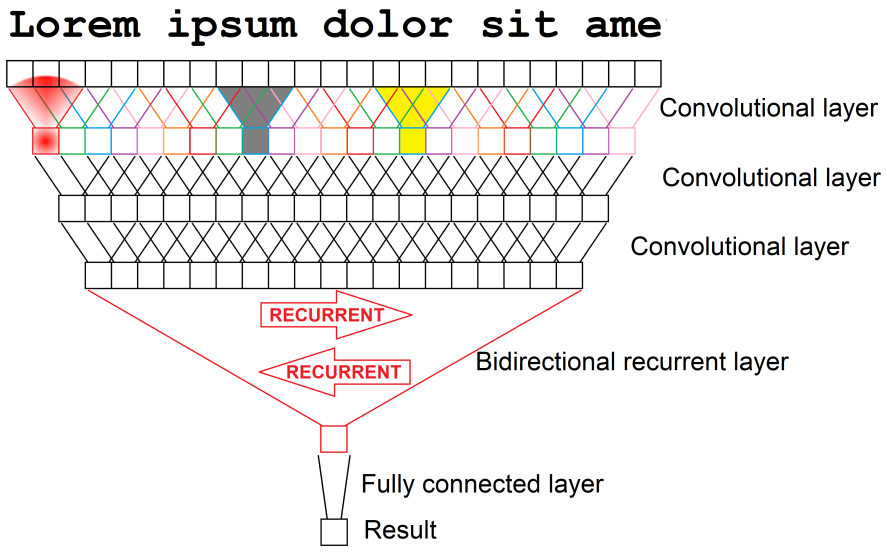

Fig. 1. Tweets classifier. Each character in the input is encoded as a vector of binary flags. Each box represents a vector.

Its input was represented as a list with fixed length (leading to padding, added to the left, or truncation, to the right, where needed) of sparse vectors. Each vector of the list represented an individual character of the tweet and contained flags whose values were either 0 or 1. Most of the flags were mutually exclusive and were used to identify a character among a list of known ones. Additional flags were used to represent additional properties of the character (such as being uppercase).

[1] Under Expat License: https://github.com/Aspie96/ThesisValentinoGiudice2018.

A series of three unidimensional convolutional layers, each with a width of 3 and an output depth of 8, convolving along the length of the tweet, was used to reduce the size of such a representation and to encode characters in a way which accounted for the context provided by the neighbouring characters. The output of the last convolutional layer, still a temporal sequence of fixed-size vectors, was used as input to a bidirectional GRU layer, originally described in [1], (similar to the LSTM architecture, originally described in [7] and then improved in [3]) whose role was to produce an individual vector representation of the whole text. The vector obtained in this way could, therefore, be the input of a simple binary classifier: an individual dense layer was used for this purpose.

The described neural network, as presented in [4], had been developed for the English language (although its application on tweets in English was outside the scope of [4]) and the Italian language. Because some characters exist in the Spanish language that don't in Italian or English, this required the neural network to be adapted for the Spanish language, leading to a slightly different input representation, as described in [6].

The full list of known characters in the used representation was the following:

```
Space ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7
8 9 : ; = ? @ [ ] \_ a b c d e f g h i j k l m n o p
q r s t u v w x y z | ~
```

Emojis were represented similarly to their Unicode name (in English), with additional flags.

The full list of additional flags, as described in [6], was:

Uppercase letter Indicating whether the character is an uppercase letter.
Accent Indicating whether the character is an accented wovel, regardless of the accent being acute or grave.
Emoji Indicating whether the character is part of the Unicode name of an emoji.
Emoji start Indicating whether the character is the first in the unicode name of an emoji.
Letter Indicating whether the character is a letter.
Number Indicating whether the character is a numerical digit.
Inverted Indicating if the character is an opening inverted question mark or exclamation mark.
Tilde Indicating whether the character is an N with virgulilla.

The neural network presented in [4] and [6], except for its last layer, the classifier, will, from now on, be referred to as `networkA` and constitutes the basis for the model used for the subtask A of NEGES. Thus, the last layer of `networkA` is the recurrent one and its output constitutes an individual vector representation of the text given in input. It must be noted that it has been slightly tweaked between different tasks (not always out of necessity, but resulting in several slightly different versions): the differences, however, are minor.

The subtask A of NEGES was much different from tweets classification: the texts were much longer than individual tweets and they were not to be classified: instead, elements within the text were to be recognized.

The problem was considered by the Aspie96 team as a problem of classification of words within the text: each word had to be classified as either:

- Not part of a negation cue, at all (118441 instances in the training set).
- The first word of a new negation cue (2490 instances in the training set).
- Part of the latest started negation cue (597 instances in the training set).

This assumed a negation cue containing multiple words, although possibly not contiguous, could not contain, between its first and last words, any part of other negation cues. This was mostly the case in the training set: only 26 words constituted exceptions to this rule as they were non beginning parts of negations cues and they were not part of the last began negation cue.

In the input texts, already tokenized, each token was marked as either a word or not a word (according to a simple match with a regular expression): words were the main focus of the task as they were the ones which had to be classified.

The representation of the text was built as follows.

In the text, spaces (which were not provided) were inserted back, simply by putting one between every two tokens. This might not have been accurate (it often is not in the case of punctuation as, for instance, commas are not usually preceded by a space), but, because the network was not pretrained and all the knowledge about the language had to be gained from scratch, this was irrelevant.

Then each word was represented as a fixed-size (of length 50) list of vectors, each of which representing an individual character. The word being represented was centered in its representation and, because of the length of the representation of each word being fixed, the neighbouring characters, on the left and on the right, whether belonging to other words or not (in the case of spaces or non-word tokens) were used as padding.

To each vector representing an individual character a flag was added, the *token flag* whose value specified if the character was part of the word being represented or the padding (its name comes from the fact that its value is 1 if the character is part of the token being represented).

As an example, let us consider the following sentence:

```
Lorem ipsum dolor sit amet, consectetur
adipisci elit, sed do eiusmod tempor
incidunt ut labore et dolore magna
aliqua.
```

And let's assume the length of the representation of each word to be 14.
The 5^{th} word (`amet`) has a length of 4 and will therefore need $14-4=10$ characters of padding: $10/2=5$ on the left and $10/2=5$ on the right.
It will, therefore, be represented as:

sit amet, con

The underlining means that the token flag for the marked characters has value 1.
The whole text has 19 words and will thus be represented as 19 lists of 14 vectors: each list representing a word (and its neighbouring characters) and each vector representing an individual character.

Because the representations of each token already encodes the neighboring characters as well there is no need to represent non-word tokens as individual elements.

This results in a representation in which each element corresponds to a word and is a fixed-size list of vectors of binary flags, where each individual vector represents a character.

Because the representations of each token already encoded the neighboring characters as well there was no need to consider anything other than a word as a token.

Based on this representation and on `networkA` (which, given a short text as input, represented as a list of vectors of flags, outputs an individual vector representation of such text), a model to solve the task could be built.

`networkA` was used to convert the representation of each word into an individual vector: in the model presented by the Aspie96 team for the subtask A of NEGES, `networkA` convolves trough the text, word by word.

The result is a list of as many fixed-size vectors as there are words in the original text and such that each one of them represents the corresponding word in the original text, with its context provided by the neighbouring characters, thus taking into account the different meanings a word could have depending on its context.

The representation obtained in this way is the input to the following layers of the neural network: a recurrent layer. All outputs of the recurrent layer, each of which being a vector, are considered (also as many as the words in the text), not just the last one. The role of the additional recurrent layer is to consider each word not one by one, but according to its context, provided by the other words, reading the text fully.

A simple classifier (a simple neural network, using the softmax function as output) is then applied to each such vectors individually to get, for each word (corresponding, in position, to the vector), its classification in one of the three classes.

The resulting model is represented in Figure 2.

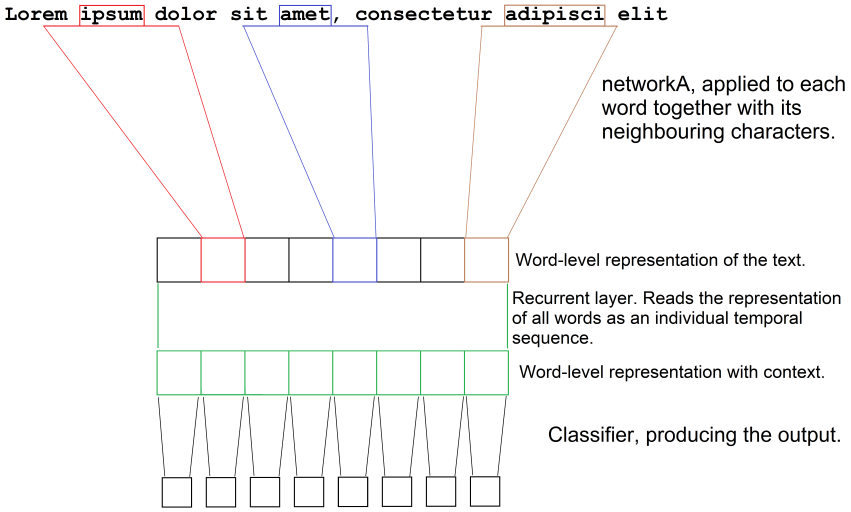

Fig. 2. Model used in the subtask A of NEGES. Each word is represented as a fixed-size list of vectors, each of which representing an individual character, using neighbouring characters as padding. This sparse representation of each word is then separately fed trough `networkA`, to get a vector representation of each word (in the context provided by the padding). The output vectors, as many as the words in the text, are then all fed, as an individual temporal series, trough a recurrent layer, whose all outputs are considered. Then, the classifier is used to classify each vector, corresponding to a word.

It must be noted that in the presented model PoS-tags and lemmas are not unused.

In order to reduce the amount of false positives, a copy of the model was used to distinguish, simply, between words that were not part of negation cues and words that were (whether as the first word or not), resulting in two copies of the model, trained separately (one for classification among all the three considered

classes and one for classification among the first and the conjunction of the other two). In the final classification, words were marked as not part of negation cues if that was the result of either one of the copies of the model, otherwise one of the remaining two classes was selected.

3 Results

Models have been evaluated by the task organizers using precision, recall and F1-score, using the same script developed for [11]. Only word tokens are considered and:

- A true positive requires all parts of a negation cue to be correctly classified (essentially, a negation cue is correctly classified if all and only the tokens it contains are classified as part of the same negation).
- To avoid penalizing partial matches more than missed matches, they are counted as false negatives, but not false positives.

The results of the Aspie96 team were quite modest and are presented in Table 1.

Table 1. Results of the Aspie96 team, compared to those of the best system according to F1-score for each domain.

Domain	Precision	Recall	F1-score	Best system	Best F1-score
cars	0.1942	0.2941	0.2339	CLiC	0.8819
hotels	0.1059	0.1525	0.1250	NLP_UNED	0.8302
washing machines	0.2424	0.3478	0.2857	CLiC	0.8413
books	0.1600	0.2857	0.2051	NLP_UNED	0.8266
cell phones	0.1807	0.2632	0.2153	CLiC	0.8113
music	0.2417	0.3333	0.2802	CLiC	0.8553
computers	0.1736	0.2593	0.2080	CLiC	0.9212
movies	0.2053	0.3313	0.2535	NLP_UNED	0.8553
average	0.1880	0.2834	0.2258	CLiC	0.8409

The average precision, recall, and F1-score for the Aspie96 team were, respectively, 0.1880, 0.2834 and 0.2258. As a comparison, the best average precision, recall and F1-score were, respectively, 0.9182 (NLP_UNED team), 0.7940 (CLiC team) and 0.8409 (CLiC team).

The results do suggest the unsuitability of the model for the task.

The reasons for this needs to be investigated. The copy of the model which simply classifies words as part of a negation cue or not, although a seemingly unnecessary source of complexity (as the problem is of classification among three classes, which is dealt with by the other copy of the model) is not the source of the problem as it causes very few false negatives on its own. In a test, it resulted in 9480 true negatives, 365 false positives (which can be dealt with by the main

model, which classifies among all three classes), 560 true positives and just 54 false negatives. As its purpose was simply to prevent at least some false positives, without causing too many false negatives, this was its intended behavior.

Another apparent cause of the poor results could seem to be the main assumption of the model: that three classes are enough to solve the problem, although it not being a problem of classification by itself. However, it must be noted that in the training set only 26 words could not possibly have been properly handled by the system as a result of this assumption: in every other case, no negation cue contained, between its starting and ending word, any part of other negation cues.

The root of the problem must, therefore, be the structure of the model itself, perhaps inapplicable due to the very high unbalance of the classes.

4 Discussion

The poor results obtained with the proposed model suggest the unsuitability of the approach for the task at hand, while the good results obtained by other teams suggest much room for improvement.

The main disadvantages of the proposed model are, arguably, its most basic characteristics: it only works on what can be strictly and automatically learned from the text contained in the documents, but a lot more could be used, such as, to name just a few, PoS-tags and lemmas, which were provided in the datasets, pre-computed word-level features, such as word embeddings and common knowledge about negations and the Spanish language.

An extremely similar model was employed in the FACT (Factuality Analysis and Classification Task) task for factuality classification, also part of IberLEF 2019 as described in [5], resulting in much better results: the task consisted in classifying marked words, within a text, among three different categories. The Aspie96 team ranked 2nd (out of a total of 5 teams), with results very close to that of the first ranking team.

This means this kind of model is indeed suitable for the classification of words within a text, while the results got in [4] show, in general, the suitability of character-level models for natural language processing.

Further research may include the creation of a unique structure of `networkA`, without alterations or multiple versions, to be applied to different tasks, resulting in a more stable neural network, and the exact same structure, based on `networkA`, to be used for both the FACT task and the NEGES task, in order to allow for a better comparison between the results.

Any persistent difference in the result will be due, strictly, to the differences between the tasks.

The main difference is that the subtask A of NEGES isn't of classification of specific words within a text but, rather, it requires the recognition of elements, some of which composed of multiple, even non contiguous, words. It is very possible that, although this can be indeed converted to a task of classification of every word within the text (which already is different than only trying to

classify some, already marked, ones), the two kinds of task are not to be seen as similar ones. Alternatively, the problem might have been caused by the highly unbalanced classes: the negative class is much bigger than the other ones, as it includes every word in the text which isn't part of a negation cue. In FACT, the thee classes had 2918, 1171 and 255 words each: most words did not belong to any class and they were not to be classified.

The main point of the model presented in [4] was to be able to work across different tasks, in different languages (originally English and Italian) and any model based upon it should inherit such properties.

It is, therefore, needed to create a more general model than the one used in the FACT task, able to work properly in different tasks. Because the results obtained in this paper are so poor, the NEGES dataset can still be used for this purpose, with the aim of producing a model still able to work on FACT, but also on NEGES.

Whether this is possible will require more research and a more stable structure for `networkA`, upon which to build a model for classification of words within a text.

References

1. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (Oct 2014), https://doi.org/10.3115/v1/D14-1179
2. Cignarella, A.T., Frenda, S., Basile, V., Bosco, C., Patti, V., Rosso, P., et al.: Overview of the EVALITA 2018 task on irony detection in italian tweets (IronITA). In: Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18). pp. 26–34 (2018), http://ceur-ws.org/Vol-2263/paper005.pdf
3. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to Forget: Continual Prediction with LSTM. Neural Computation **12**(10), 2451–2471 (2000), https://doi.org/10.1162/089976600300015015
4. Giudice, V.: Aspie96 at IronITA (EVALITA 2018): Irony Detection in Italian Tweets with Character-Level Convolutional RNN. In: Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18). pp. 160–165 (2018), http://ceur-ws.org/Vol-2263/paper026.pdf
5. Giudice, V.: Aspie96 at FACT (IberLEF 2019): Factuality Classification in Spanish Texts with Character-Level Convolutional RNN and Tokenization. In: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2019). CEUR Workshop Proceedings, CEUR-WS, Bilbao, Spain (2019)
6. Giudice, V.: Aspie96 at HAHA (IberLEF 2019): Humor Detection in Spanish Tweets with Character-Level Convolutional RNN. In: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2019). CEUR Workshop Proceedings, CEUR-WS, Bilbao, Spain (2019)
7. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation **9**(8), 1735–1780 (1997), https://doi.org/10.1162/neco.1997.9.8.1735

8. Jiménez-Zafra, S.M., Cruz Díaz, N.P., Morante, R., Martín-Valdivia, M.T.: NEGES 2019 Task: Negation in Spanish. In: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2019). CEUR Workshop Proceedings, CEUR-WS, Bilbao, Spain (2019)
9. Jiménez-Zafra, S.M., Díaz, N.P.C., Morante, R., Martín-Valdivia, M.T.: NEGES 2018: Workshop on Negation in Spanish. Procesamiento del Lenguaje Natural **62**, 21–28 (2019)
10. Jiménez-Zafra, S.M., Taulé, M., Martín-Valdivia, M.T., Ureña-López, L.A., Martí, M.A.: SFU ReviewSP-NEG: a Spanish corpus annotated with negation for sentiment analysis. A typology of negation patterns. Language Resources and Evaluation **52**(2), 533–569 (2018), https://doi.org/10.1007/s10579-017-9391-x
11. Morante, R., Blanco, E.: *SEM 2012 Shared Task: Resolving the Scope and Focus of Negation. In: Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation. pp. 265–274. SemEval '12, Association for Computational Linguistics, Stroudsburg, PA, USA (2012), https://aclweb.org/anthology/papers/S/S12/S12-1035/