RoMa at HAHA-2021: Deep Reinforcement Learning to Improve a Transformed-based Model for Humor Detection

Mariano Rodriguez[1], Reynier Ortega-Bueno[2], and Paolo Rosso[2]

[1] Universidad de Oriente, Cuba
mjasoncuba@gmail.com
[2] PRHLT Research Center, Universitat Politècnica de València, Valencia Spain
{rortega,prosso}@prhlt.upv.es

Abstract. In this paper, we describe our system we participated in the shared task *"Humor Analysis based on Human Annotation (HAHA) at IberLEF-2021* with. Our system relies on data representations learned through fine-tuned neural language models. The representations are used to train a Siamese Neural Network (SNN) which learns to verify whether or not a pair of tweets belong to the same or distinct classes. A key point in our model is the heuristic used to create the pair of messages in the training and test phases. For that, we used a Deep Reinforcement Learning (DRL) strategy that aims at identifying a set of optimal prototypes in each class. In general, the results achieved are encouraging and give us a starting point for further improvements.

Keywords: Humor recognition · Deep Reinforcement Learning · Siamese Neural Networks

1 Introduction

Humor is an important part of human communication. Time ago a philosopher had a conception establishing that humor, deep down, is a type of catharsis that makes existence more bearable, like art. He said:

> Perhaps I know best why it is man alone who laughs; he alone suffers so deeply that he had to invent laughter. (Nietzsche, 1888)

Humor comes from a variety of sources, making it a real challenge to design a computational model for addressing its automatic recognition on texts. Sometimes humorous texts use wordplays as an engine to provoke laughter, in other cases they appeal to social, cultural, and commonsense backgrounds to produce

IberLEF 2021, September 2021, Málaga, Spain.

funny response. In other cases it makes use of irony, satire, hyperbole and other figurative devices to achieve its goals. The difficulty of the task increases when language is short and informal like the one used in Twitter. All this raises interest in humor recognition tasks within Natural Language Processing (NLP) and Human-Computer Interaction (HCI). On this line, the HAHA Task: Humor Analysis based on Human Annotation at IberLEF-2021 aims at computationally recognizing humor in Spanish tweets [4].

In this paper, we adapt and re-evaluate the RoMa system [14] that we employed int Task 7 at SemEval21 [10] for addressing the task of humor recognition in Spanish tweets. This architecture combines learned representations with an SNN [1] to learn a metric for discriminating whether or not a pair of tweets belong to the same class (i.e., humorous tweets and not humorous tweets). Also, we considered applying a variation by introducing Deep Reinforcement Learning [9] within its structure. We bring empirical evidence through experiments on the human-annotated datasets that the DRL-based strategy outperforms the original version of the RoMa system. The source code of the work is public available on GitHub: `https://github.com/mjason98/haha21`

The paper is organized as follows: in Section 2 we briefly introduce the proposed task and the main ideas that motivated our proposal. Section 3 presents our proposed architecture and gives details about their modules. The experiments and results are described in Section 4. Finally, in Section 5 we present our conclusions and interesting directions for future work.

2 Background

The 2021 edition of the HAHA shared task, as part of IberLEF-2021, aims at classifying humorous tweets written in Spanish. The first subtask is about Humor Detection and proposes the problem of determining whether a tweet is funny or not. An annotated corpus of tweets in Spanish was provided to carry out this task. The dataset is composed of 24000 tweets, 9253 labeled as humor, and 14747 as not humor. In this work, we focused only on this subtask.

In previous works, specifically in HaHakathon at Semeval21, with the team named RoMa, we presented a system to address the problem of humor detection, based on SNN. The Siamese model (SiaNet) required a pair of messages as input in both training and test phases. In that work, we transformed the classification problem into a verification one, in which data is classified by comparisons with two reference sets employing the SiaNet model. This algorithm can be split into three main steps:

1) A Pretrained Language Model (PLM) was used to represent the tweets as vectors. We used a transformer model [15] as PLM and fine-tune it on the humor dataset to achieve a target-dependent representation of the data.
2) Vectors are separated into two classes: the first one contains the vectors corresponding to tweets annotated as humorous, and the second contains the opposite set. Then, a clustering algorithm is applied and a prototype set is extracted for each class.

3) Finally, each element from the dataset is paired with a prototype generating negative and positive examples (i.e., pairs from the same and opposite classes respectively). These pairs are used for training the SiaNet model. When this training process ends, the system is used to classify unlabeled tweets by giving them the label of the closest prototype. The closeness of two messages is measured by a distance function, in this case, the SiaNet.

The interest in Siamese architectures remains strong, so we decided to test the performance of this algorithm (RoMa) in the HAHA humor task introducing a variation in the second step.

The clustering algorithm used in RoMa was a graph-based method, and one of the challenges with this approach was the threshold tuning in the graph construction. In the RoMa system, we build a graph of β-distance, analogous to the β-similarity graphs proposed in [6], for the humor and non-humor classes. The nodes in the graphs represent the tweets from the training set and the edges joining two nodes are weighted with the distance between them. In the β-distance's graph the edges with weights greater than the threshold β are removed, allowing only the closest representations to be in the same connected subgraph. Afterwards, communities are detected on the β-distance's graphs employing the InfoMap algorithm [5] which is based on the map equation [12]. This algorithm reports a set of subgraphs whose nodes are paired with a flow value. For each subgraph, the k nodes with the highest flow value are selected as prototypes.

The threshold is adjusted until the number of extracted prototypes lies within a range. This interval is defined by two integers. In this work, as in RoMa, 200 and 300 were used. It is a fact that the performance of SiaNet in the task will be linked to the quality and quantity of the extracted prototypes. A variation in the threshold carries a different graph structure and, therefore, different prototypes. We found out that the SiaNet model is very sensitive to these variations.

In this work we propose, to replace the clustering algorithm by a Deep Q-Learning approach, where the number of prototypes is controlled by an upper limit value set in advance. One advantage of this approach is that the relationships among tweets are learned, in contrast to the graph method where the Euclidean distance was used to weigh the edges.

3 System Overview

We keep the first and third steps as well the modular schema of the RoMa architecture. This is the composition of an encoder module (*Encoder*) and a prediction module (*Classifier*), which are trained independently. We replace the clustering-based approach from the prototype selection phase by a Deep Q-Network (DQN) method. In the following subsections we provide the most relevant details of our system.

3.1 Encoder Module

The Encoder plays an important role because it is concerned with learning an abstract representation that vanishes the colinearity between its features and

compresses the textual information on a single dense vector. The core of our Encoder's design is based on a Transformer model (TM). Our architecture differs from RoMa system in the pretrained TM. Particularly, in this work we used BETO [3] since HaHakathon was a challenge based on English tweets.

For fine-tuning the TM-based encoder we add an intermediate layer that receives the vectors from the output sequence of the TM. On this sequence of vectors, a normalized sum operation is applied. Then, an output layer makes the final prediction for the targeted task. For each layer of the TM, a different learning rate is set up, increasing it using a multiplier while the neural network gets deeper. This multiplier increases 0.1 points from a layer L_i to another L_{i+1}. We use this dynamic learning rate to keep most information from pre-training at shallow layers and biasing the deeper ones to learn about the specific tasks.

3.2 Prototype Selection through Deep Q-Learning

The task of prototype selection is addressed using Deep Q-Learning [11], which is a model-free reinforcement learning technique [13]. The reinforcement learning algorithm, which is called the agent, learns by maximizing rewards in some environment. At each time step $t = (0, 1, 2, ..., n)$, the agent receives as input data the state s_t, which is a snapshot of the environment. Then, the agent evaluates that data and takes the action a_t, from a set of possible actions given its current state. At the next time step, the environment gives a reward, r_{t+1}, to the agent and change itself to a new state s_{t+1}. The rewards are the only learning signals the agent is given. Maximizing the total reward that the agent receives is its goal.

Environment and States Our environment works with the vector representation of the tweets produced by the Transformer model. For every time step, it gives to the agent the k-th vector v_k and the currently candidate prototypes p_{t_i} in a list as state s_t:

$$s_t = (p_{t_1}, p_{t_2}, ..., p_{t_M}, v_k) \tag{1}$$

$$p_{0_i} = 0$$

where M is the total number of prototypes allowed, and $k = 1 + (t \; mod \; T)$ with T the length of the training set.

Agent and Actions The agent was designed using two layers of Multi-head Attention mechanism [15] and a feed-forward layer on top with the output size of the total number of actions. We find the use of Multi-head Attention, in this case, convenient since the state s_t is given by a list of vectors where the agent must be related with all the current prototypes in order to learn relations allowing to perform the best-rewarding action. We provide to the agent a total of $M + 1$ actions. In the first M actions a_i we replace p_{t_i} with the vector v_t, and in the last action a_{M+1} we ignore that vector.

Reward The reward was designed as follow:

$$r_{t+1} = \begin{cases} 0, & t \; mod \; W \neq 0 \\ ACC(p_{t+1_1}, p_{t+1_2}, ..., p_{t+1_M}), & t \; mod \; W = 0 \; or \; t \; mod \; T = 0 \end{cases} \quad (2)$$

where W is a hyperparameter and ACC a metric that measures the accuracy of predicting the humor class on the validation set using the prototypes $\{p_{t+1_i}\}$. The prediction operation can be described by equation 3, using the Euclidean distance as D_f instead of the SiaNet model.

3.3 Training the DQN Model

It is well known, that the training phase of deep reinforcement learning algorithms often is strongly time-consuming. For that reason, we introduce the parameter W in the reward design. The algorithm starts with W equal to 10% of the training data size. We model a reward schedule such that for a set of iterations index, W is increased by itself unless is greater than T. When the environment produces a reward different than 0, is conducted an environment reset, that is all the candidate prototypes are erased.

The increasing behavior of W represents an increment in the learning difficulty. In this way, we induce the agent to archive good results with few examples instead of waiting until all vectors are processed to get a reward. Therefore, fewer training iterations are needed.

Nevertheless, the number of zero rewards is still huge, which motivates us to introduce an Intrinsic Curiosity Module (ICM) [2] in our system at training time. This imbues the agent with a sense of curiosity, facing the sparse reward problem since the rewards in the environment are sparsely distributed. The process for training the agent uses the strategies proposed in [16] with initially ϵ-greedy policy and then softmax function.

After the agent finishes its training and the environment reset, the schedule of W is ignored and k takes zero value. Then, interactions between the agent and the environment start over until all vectors from the training set are processed. Finally, the prototypes within the last environment state are used by SiaNet in the third step. Remark that in contrast to RoMa, we do not run our prototype selection method separately in the humor and not humor classes. In this case, it accepts both classes as input, letting it to decide the number of prototypes for each group from the total number.

3.4 Classification Module

The classification module architecture lies on SiaNet. This network consists of two input messages and one output that indicate how distant they are according to their representation [1]. Both messages are encoded by using the fine-tuned Transformers model (see Section. 3.1). Later, each input is passed through two dense layers with 32 and 16 hidden neurons each. Then, the representations

of the messages are compared to each other through a distance metric. The specific features the model learns to extract, make that message representations corresponding to opposite classes have a distance greater than the threshold defined in the loss function used. Particularly, we used the Contrastive Loss [7] with an empirical value threshold of 0.85.

For training SiaNet, the dataset needs to be processed for constructing pairs of messages from the same class and pairs of messages from distinct classes. The process to compound the pairs used remains equal to the one in [14]. During the test phase, given an unlabeled tweet, we obtain the encoding of z by using the Encoder. After that, we predict its label using the next equation:

$$\hat{y} = arg\min_{i}\{D_f(z, x_{i,j})\} \tag{3}$$

where $x_{i,j}$ is the j^{th} prototype in the class i, $i \in \{no\_humor, humor\}$, and D_f our SiaNet.

4 Experiments and Results

In this edition of HAHA, the contest had development and evaluation phases. Submissions of system predictions were allowed in both phases, but the official results of the competition were only those from the second one. We use a 10-fold validation strategy for hyperparameters tuning during the whole contest.

An epoch in the training process of the agent finishes once an environment reset arrives. In this work, we use 2000 epochs. All learnable parameters for the PLM, the agent, the ICM and SiaNet were trained using Adam [8]. As part of the experiments, our system was tested on the classification tasks in previous editions of HAHA. In Table 1 can be observed the performance of our system in the 2018 and 2019 competitions, where F_1 was used as evaluation metric. Columns *prototypes* and *icm* represent the maximum number of prototypes and the use of the ICM strategy. The last row shows the results using the graph-based clustering algorithm (GBC) used by the RoMa system.

Table 1. Results of our systems in HAHA'18 and HAHA'19

system	prototypes	icm	F_1-(2018)	F_1-(2019)
DQN	100	yes	0.8256	0.8007
	100	no	0.8256	0.8005
	52	yes	0.8615	0.8201
	52	no	0.8614	0.8201
GBC	276	-	0.8478	0.7982

Looking at Table 1, the results using ICM are similar as when it is omitted. We hypothesize that W and the schedule over this parameter mitigates the malicious consequences of the sparse reward in this particular environment-reward

pair design. Another hypothesis lies in the similarity of the states, produced by the vector representation provided by the fine-tuned PLM. Both ideas need a deeper analysis which we plan to explore in future works.

During the experiments, increased stability in SiaNet's training was observed. This is, small changes in the hyperparameters did not impact the model's learning curve. This stability effect only occurred when prototypes produced by the DQN agent were used instead of those generated by the RoMa method. Also, better results were found using fewer prototypes. For the official submission, 52 prototypes with ICM was the best setting for our system.

4.1 Official Results

For the evaluation phase we submitted predictions made by our system as well as predictions from the RoMa system. The one based on Deep Q-Network achieved the best results among our submissions, ranking us in 7^th place out of 17 teams with F_1-Score of 0.8583, whereas the best system reached 0.8850.

An interesting fact about the performance of our system was the negative effect of increasing the number of prototypes. This effect should be the opposite since having more references to compare an unseen message must yield more steady predictions. We hypothesize that one cause is the agent-actions design. In our model, the number of actions is equal to the number of prototypes plus one, which implies the use of considerably large action space and induces the agent to learn a more complex strategy.

5 Conclusions

In this work, we presented a model for addressing the humor recognition in Spanish tweets. The model employs the deep representations learned by Transformers models for encoding the tweets. These representations are used by a Siamese Neural Network combined with a Deep Q-Network prototype selection method. A key point of our system was the schedule creation within the reward design to reduce training time. The achieved results show that our system outperformed the original version of RoMa, based on graph clustering.

As future work, we will analyze why ICM did not provide the expected improvement during our training process since lots of zero rewards were present. Also, we propose through experimentation with a non-fixed size action space, to prove the hypothesis of the agent-actions design problem presented in Section 4.1. Another direction we plan to address is exploring the use of more robust Deep Reinforcement Learning algorithms and reward policies.

6 Acknowledgments

The work of the second and third authors was in the framework of the research project MISMIS-FAKEnHATE on MISinformation and MIScommunication in

social media: FAKE news and HATE speech (PGC2018-096212-B-C31), funded by Spanish Ministry of Science and Innovation, and DeepPattern (PROMETEO/2019/121), funded by the Generalitat Valenciana.

References

1. Bromley, J., Bentz, J.W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., Shah, R.: Signature Verification Using a "siamese" time delay Neural Network. International Journal of Pattern Recognition and Artificial Intelligence **7**(04), 669–688 (1993)
2. Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., Efros, A.A.: Large-scale Study of Curiosity-Driven Learning. arXiv preprint arXiv:1808.04355 (2018)
3. Cañete, J., Chaperon, G., Fuentes, R., Ho, J.H., Kang, H., Pérez, J.: Spanish Pre-Trained BERT Model and Evaluation Data. In: PML4DC at ICLR 2020 (2020)
4. Chiruzzo, L., Castro, S., Góngora, S., Rosá, A., Meaney, J.A., Mihalcea, R.: Overview of HAHA at IberLEF 2021: Detecting, Rating and Analyzing Humor in Spanish. Procesamiento del Lenguaje Natural **67**(0) (2021)
5. Edler, D., Anton, E., Rosvall, M.: The MapEquation Software Package. URL: https://mapequation. org (2020)
6. Garcia, R.J.G.: Algoritmos de Agrupamiento sobre Grafos y su Paralelización. Ph.D. thesis, Universidad Jaume I (2005)
7. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality Reduction by Learning an Invariant Mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 2, pp. 1735–1742. IEEE (2006)
8. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), `http://arxiv.org/abs/1412.6980`
9. Li, Y.: Deep Reinforcement Learning: An Overview. arXiv preprint arXiv:1701.07274 (2017)
10. Meaney, J., Wilson, S., Chiruzzo, L., Lopez, A., Magdy, W.: SemEval 2021 Task 7: Hahackathon, Detecting and Rating Humor and Offense. In: 15th International Workshop on Semantic Evaluation (2021)
11. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing Atari with Deep Reinforcement Learning. arXiv preprint arXiv:1312.5602 (2013)
12. Rosvall, M., Axelsson, D., Bergstrom, C.T.: The Map Equation. The European Physical Journal Special Topics **178**(1), 13–23 (2009)
13. Sutton, R.S.: AGB: Reinforcement Learning: An Introduction. A Bradford Book (1998)
14. Tamayo, R.L., Rodriguez, M.J., Bueno, R.O., Rosso, P.: Roma at Semeval-2021 Task 7:A Transformer-based Approach for Detecting and Rating Humor and Offense. In: Proceedings of the 15th International Workshop on Semantic Evaluation (2021)
15. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is All you Need. arXiv preprint arXiv:1706.03762 (2017)
16. Zai, A., Brown, B.: In: Deep Reinforcement Learning in Action, pp. 223–234. Manning Publications (2020)