Infotec + CentroGEO at SemEval-2020 Task 8: Deep Learning and Text Categorization approach for Memes classification

Guillermo Ruiz* and **Eric S. Tellez**** and **Daniela Moctezuma***
and **Sabino Miranda-Jiménez**** and **Tania Ramírez-delReal*** and **Mario Graff****

*CONACyT - CentroGEO, Aguascalientes, México
**CONACyT - INFOTEC, Aguascalientes, México
{lgruiz,dmoctezuma,tramirez}@centrogeo.edu.mx
{eric.tellez,sabino.miranda,mario.graff}@infotec.mx

Abstract

The information shared on social media is increasingly important; both images and text, and maybe the most popular combination of these two kinds of data are the memes. This manuscript describes our participation in Memotion task at SemEval 2020. This task is about to classify the memes in several categories related to the emotional content of them. For the proposed system construction, we used different strategies, and the best ones were based on deep neural networks and a text categorization algorithm. We obtained results analyzing the text and images separately, and also in combination. Our better performance was achieved in Task A, related to polarity classification.

1 Introduction

The kind of information usually shared on social media can be in various modalities such as text, images, audio, or video. In spite that this kind of information is shared in a recreational way, many institutions have paid attention to what the users or citizens shared on the Internet. An aspect that has been analyzed in the past is the polarity of the text (for instance tweets, reviews, news). The polarity reflects if the text has negative, positive, or neutral content. More recently, other kinds of data have gain attention, for instance, images. The most popular and descriptive mix of image and text are the memes. Richard Dawkins proposed the term *meme* referring to *mimeme* in Greek, that is associated with imitation (Dawkins, 2016). He expressed that a meme is a form of social engendering or cultural propagation. Memes are important because they express emotion, joke, humor, or something that only using words could not be possible. The automatic classification of memes has very important challenges. The meaning of a meme could be interpreted in different ways by different people. Furthermore, for analyzing and classifying the meaning of a meme, image and text processing must be tackled and combining them is the best way to reach a good classification result (Iwazaki, 2018; Xia et al., 2020), which is not precisely easy to achieve. Some works have tackled the meme classification problem. Kumar and Garg, (2019) proposed a method for detecting only sarcasm tone in memes. For the sarcastic detection, a set of typo-graphic only memes where used. The authors extracted the text from the images and used standard *TF-IDF* approach over the words. Finally, they reported a result of an accuracy of 88% with a multilayer perceptron classifier from a dataset called MemeBank obtained from the Instagram social network using the hashtags `#sarcasm`, `#sarcastic` and `#irony` as positive examples and `#motivational` and `#inspirational` as negative. Smitha et al., (2018); Peirsoon and Tolunay, (2018), and Kanai, (2016) dealt with meme classification.

Psychological study and commercial marketing are some of the primary motivations for sentiment analysis from an image with text; that is, the classification of memes can be useful in marketing, advertising, trend analysis, so on (Smitha et al., 2018). Also, performing this task using automatic techniques allows the optimization of response times in previous analyzes.

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 1141–1147
Barcelona, Spain (Online), December 12, 2020.

In this paper, we present the proposed system to tackle for Memotion Analysis Task from SemEval 2020. In the following sections is described in more detail our proposal.

The rest of the paper is organized as follows. Section 2 describes the data and the task description. Section 3 describes our approach to solve the problem and our implemented system. Section 4 details our experimental results, and finally, conclusions are given in Section 5.

2 Dataset and task description

The Memotion Analysis Task was divided into three sub-tasks: A, B, and C. The overall task description can be consulted in the task overview (Sharma et al., 2020). Task A is for sentiment classification, that is, given a meme, the system must classify its content as positive, negative, or neutral. Task B is for humor classification where the system has to identify the kind of humor expressed by the meme. The types of humor considered were sarcastic, humorous, offensive and motivational, but a meme can have more than one of these categories. Here, a meme can have one, more or none categories. Task C is for scales of semantic classes where the goal is to quantify the degree of emotion expressed in the meme, these quantifications are divided into not (0), slightly (1), mildly (2), and very (4), all of them for the sarcastic, humorous, and offensive emotions.

The sizes of the datasets provided by the competition organizers were, for the training dataset, a total of 7000 memes, 1000 memes for the trail, and finally, 2000 meme images for the test. The datasets had an associated file, with the contained text in the meme. For a profound description of the datasets see Sharma et al., (2020). The text used to feed all our models was obtained from the concatenation of the fields `text_ocr`, `text_corrected` and `image_name`. This concatenation was done looking for a more amount of text no matters if was duplicated or not.

3 Proposed system

3.1 Neural networks

A neural network is a machine learning method that has gained a lot of popularity since (Krizhevsky et al., 2012) showed the potential of deep learning by winning the Large Scale Visual Recognition Challenge 2012 (ILSVRC12) competition with the AlexNet. Following this trend, we decided to apply them for this competition. Particularly, we used the Inception architecture for our experimentation process, all details are explained below.

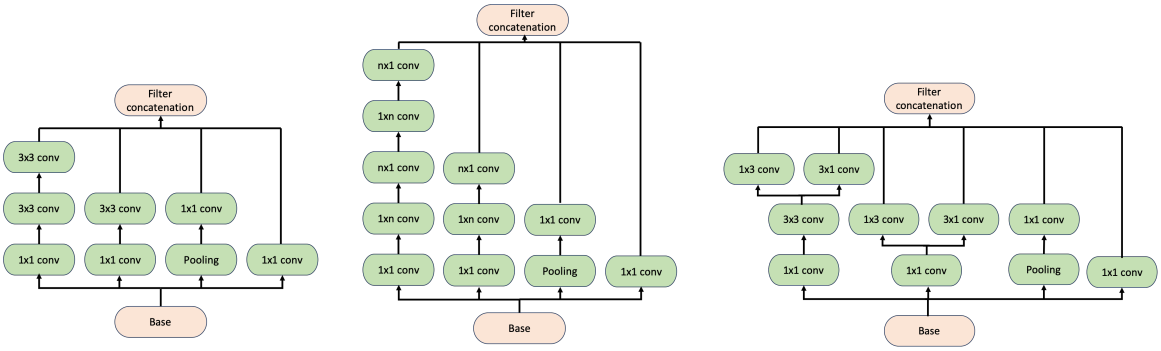
3.1.1 Inception modules

In recent years, one of the most notable concepts on the topology of a neural network is the inception module, a new architecture producing efficient and accurate predictions. The inception module was first showed by Szegedy et al., (2014) as building blocks used to create neural networks by placing one on top of another.

A later version, the Inception-V3 (Szegedy et al., 2016b) has improvements in the way the computations are done. The number of operations is reduced by changing 5×5 filters by two 3×3 filters followed one after the other. With this small change, we get an inception module as shown in Figure 1a which is used for early layers. In a similar way, the authors change $n \times n$ filters by two asymmetric convolutions: one $1 \times n$ and one $n \times 1$ (see Figure 1b). With that, they change n^2 parameters for $2n$ which is a big saving on the computations. A third type of module is used for the final layers with the configuration showed on Figure1c. With all that, the proposed architecture uses three types of inception modules. This architecture will be the base of the proposed system.

3.1.2 Residual connections

Theoretically, deeper networks should be more powerful than shallow ones. Suppose we have two neural networks A and B, where A has n layers and B has $n + m$ layers. Then, the first n layers on B can compute the same as A and then the rest of the m layers simply compute the identity. So, the network B should be able to do everything A can. Unfortunately, in practice this usually does not happen. One problem with neural networks is that the deeper they get, the more difficult it is to train them.



(a) Inception module A. (b) Inception module B. (c) Inception module C.

He et al., (2015) proposed a solution to this problem with the help of skip connections between the layers called residual connections (Figure 2). They got very convincing practical evidence for the usefulness of the connections on training deep neural networks. The rationale is that if you want to compute the output of some layers $\mathcal{H}(x)$ you can decompose it in $\mathcal{H}(x) = \mathcal{F}(x) + x$ and computing $\mathcal{F}$ is easier than computing $\mathcal{H}$.

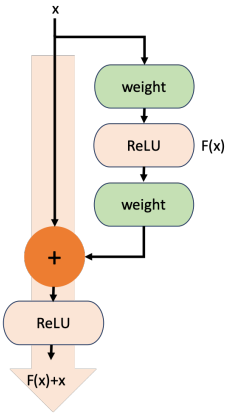
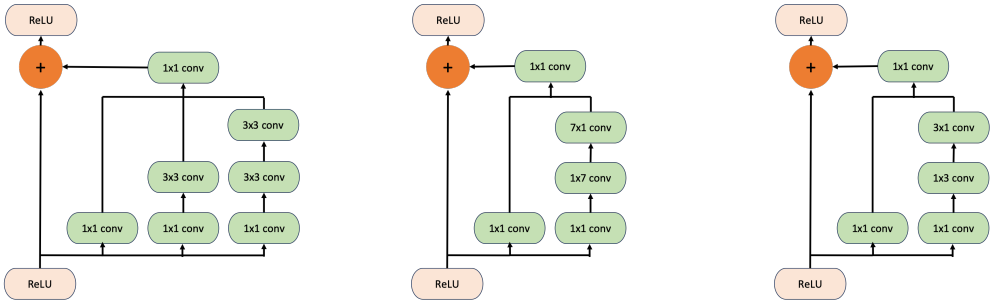


Figure 2: Residual connection.

3.1.3 The base model

We used the **Keras** (Chollet and others, 2015) implementation of the **inception-resnetV2** (Szegedy et al., 2016a) network as our base model. Basically, this version uses the inception modules with residual connections which we present on figures 3a, 3b, and 3c. The authors showed that the residual connections help to improve the training speed on their architecture.



(a) Inception module A with residual connection. (b) Inception module B with residual connection. (c) Inception module C with residual connection.

3.2 Using text and images

Once we chose the base model for images, we tried to incorporate the text data into the network. We started by using a network with two inputs, one for the image (224 by 224 pixels on the channels RGB), and one for the vectorized text. For the image input we used the **inception-resnetV2** without the top layers and no trained weights, we then flatten the output to get a vector; for the text input, we first vectorized the text using the **CountVectorizer** function of **sklearn** (Pedregosa et al., 2011) (producing a vocabulary of 11205 words after the removal of stopwords), and then apply a sequence of dense layers. The **CountVectorizer** method converts a set of text documents to a matrix of token counts, in the most simple way a token is a word of the vocabulary. After that, we had two vectors of the same length that we concatenate and apply some extra dense layer to finally get a three-way output using a softmax activation to predict the negative, neutral and positive label for the meme.

Unfortunately, the results for the images and the text were not better than just using the images. Nevertheless, the results of using only a text vectorization method, ignoring the images, were better than using both. So the strategy was to use a neural network for the images, a vector model for the text and combine them together with a neural network. In order to combine the two models, we used stacked generalization (Wolpert, 1992). For using this method, we split the dataset into 5 parts. Then trained a neural network model on 4 of the 5 partitions. With that neural network, we predicted the labels of the missing part which were stored on a file. We repeated this process in a similar way as the cross-validation. In the end, after 5 models trained, we had the unbiased predictions of the full dataset. Finally, we trained again a neural network, with the same configuration, but this time on the full data. The model was stored to be used along with the text model. The model for the images we used was an **inception-resnetV2** without the top layers and no trained weights, followed by a Global Average Pooling layer and a Dropout. During the training we used the Adam optimization with a learning rate of 0.0001 and 50 epochs. We also tried using the pre-trained version of the **inception-resnetV2** without the top layers, plus a Global Average Pooling layer, and some extra dense layers. Since the changes did not improve the results, we focus on the untrained version.

For the text model we used our text classification algorithm based on Tellez et al., (2018) called TextCategorization.jl[1]. TextCategorization.jl is a Julia package inspired by μTC. The main difference with μTC is that it performs a full model selection, and this means the combinatorial problem represents the entire text-classification pipeline. That is, each configuration (model) describes all preprocessing functions, different tokenization schemes as μTC, several term weighting schemes, and several parts of the classifier used (a kernel-based and prototype-based classifier). The selection of both kernel and prototyping schemes are part of the combinatorial problem. The combinatorial problem is tackled using a local search algorithm (Beam Search); the configuration space is sampled randomly for the initial population, and then it is explored with a mutation and crossover strategy. This algorithm was used for text representation, some efforts were the concatenation of the text representation generated by it, and the image extracted by the deep neural network approach. The neutral network for combining the image and text inputs is depicted on Figure 4. It had two inputs, one branch for the image model, and one for the text model. The outputs of the image and text models were followed by two dense layer, then a concatenation and a series of more dense layers. The activation functions used were ReLU except for the last one where we used softmax. Sadly, the results were not better than just using images of just using text.

For our final attempt to combine the images and text, we used **XGBoost** on the predictions (with the default configuration) without better results (see Table 1).

3.3 The final model

Up to this point, we were testing our image methods on Task A only and chose our best ranked neural network: the **inception-resnetV2** without the top layers and no trained weights, then a Global Average Pooling layer and a Dropout training using Adam optimization with a learning rate of 0.0001 and 50 epochs. So from now on, the base model used for Task B and Task C, was this with minor changes on the top layers to predict the more complex labels.

[1] `https://github.com/sadit/TextClassification.jl`

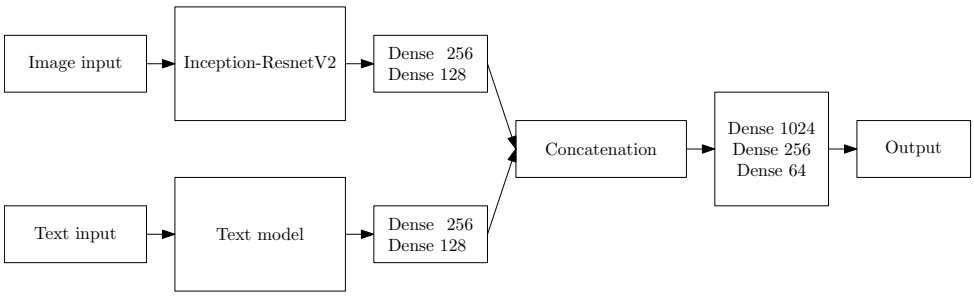


Figure 4: Neural network used for combining the image and text predictions.

The decision of applying similar models for the other tasks based on Task A was backed up after seeing this method having better scores[2] on the three tasks.

3.4 Proposal for task B

For the Task B, we tried the same as the described for Task A with the same results, so the final model was very similar than that of the previous section. The first change is that, after the Global Average Pooling layer, we added a Dense layer of size 512 and for the final layer, an output of four dimensions with the sigmoid activation. We trained the network with the same parameters as on Task A.

3.5 Proposal for task C

For the Task C, we had to change a bit more the model. After the Global Average Pooling layer, we split the network in four branches, so the output will have the form (`humour, sarcasm, offensive, motivation`). For each branch, we added a Dense layer of size 512 and finally, a Dense layer with four neurons for the first three and with 2 neurons for the motivation one. These layers used a softmax activation. So the output was the probability of getting certain level of sentiment. We obtained better results training for 20 epochs.

4 Experimental results

We have two types of experimental results, one without the test and one evaluated on it. The evaluation without the test set was made using the predicted labels from the stacked generalization. With that, we obtained unbiased predictions of the whole training set. We show the results of the **XGBoost** and our final method over the training set on Table 1. Note that the results of the **XGBoost** combines both, the images and text, to make the prediction but the results show a lower Macro-F1 score on every task, although scoring high on Micro-F1.

	Task A		**Task B**		**Task C**	
Approach	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**
XGBoost with images and text	0.3113	0.5420	0.4611	0.6812	0.2833	0.4435
Inception-resnetV2 on images only	0.3161	0.5120	0.4920	0.5848	0.3039	0.3885

Table 1: Internal results on the training set of our system construction

Next, we show the results provided by the organizers on the test set on Table 2. The bold scores are the best on each task. Note that our final pick was the best on Task A and B, and the score in Task C was below the baseline.

The Figure 5 shows the confusion matrix for the Task A on the test set, here, we can see that the model has problems classifying the negative and neutral sentiment. On Task B, the confusion matrices on Figure 6 show that our solution struggles to identify the type of humor. For Task C, the prediction is more complicated. Figure 7a shows that the model fails to detect very humorous memes. Something similar

[2]According to the score provided by the prior script (not final) used by the organizers.

	Task A		Task B		Task C	
Approach	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**
Baseline	0.2176	0.3078	**0.5002**	0.5687	0.3009	0.3328
Inception-resnetV2 with images and text (CountVectorizer)	0.3339	0.4638	-	-	-	-
Inception-resnetV2 with images and text (TextClassification.js)	0.2877	0.5021	0.4979	0.6105	**0.3040**	0.4029
XGBoost on Inception-resnetV2 and TextClassification.js	0.3038	**0.5431**	0.4627	**0.6717**	0.2761	**0.4386**
Inception-resnetV2 on images only (**final pick**)	**0.3469**	0.5181	0.4773	0.6121	0.2758	0.3807

Table 2: Official results of our methods

happens on Figure 7b with problems also on not sarcastic memes. Figure 7c follows the same trend for offensive samples. The motivational detection on Figure 7d shows that most of the errors occur assigning the negative label to positive examples.

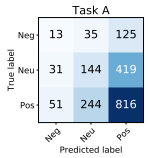
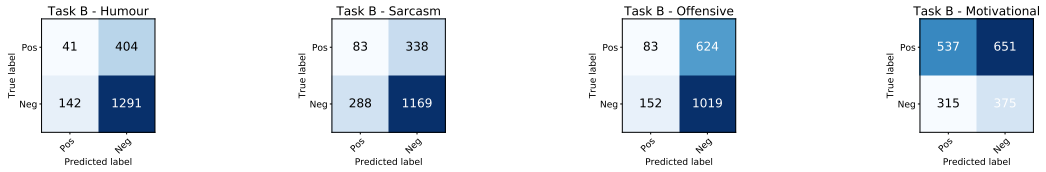
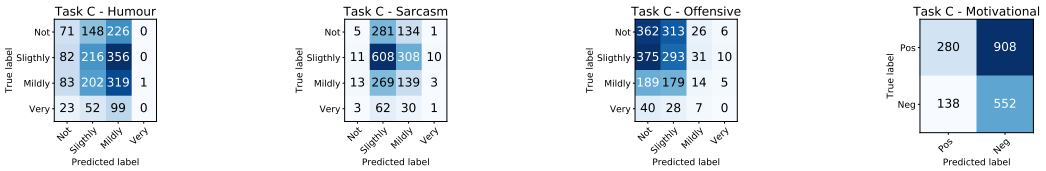


Figure 5: Confusion matrix for Task A



(a) (b) (c) (d)

Figure 6: Confusion matrices for Task B



(a) (b) (c) (d)

Figure 7: Confusion matrices for Task C

Finally, the official results for our participation are in Table 3. The official results are in Macro-F1 and Micro-F1, the best results in each task are in bold. In summary, our system got position six in Task A, position 24 in Task B, and position 25 in Task C. As can be seen, our best performance was in Task A, this is because we mainly design our system based on the internal results obtained with Task A, but we wanted to also submit our system to the other tasks.

	Task A		Task B		Task C	
TEAM	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**
vkeswani_IITK	**0.3546**	**0.4872**	0.4889	0.6235	0.3145	0.3651
guoym_guoym	0.3468	0.5181	0.5146	0.6231	**0.3224**	**0.3779**
INGEOTEC	0.3468	0.5181	0.4773	0.6120	0.2758	0.3807
george.vlad_eduardgzaharia_UPB	0.3453	0.5218	**0.5183**	**0.6144**	0.3171	0.4054

Table 3: Official results: Best ones and our results by tasks

5 Conclusions

This paper describes our system submitted to Memotion Analysis SemEval 2020 Task. The proposed system was based on deep learning, specifically using the Inception-resnetV2 as a base model. Unfortunately, we reached the best results using in a separate way the text and the images instead of using both. We design our system based on Task A in which we obtained six general position (over 35 results) which is a good result for our team, nevertheless, in Task B and C our results were in 24 and 25 positions.

Acknowledgements

This research is supported by the project A1-S-34811 of Basic Science grant by the National Council of Science and Technology (CONACyT) from Mexico.

References

François Chollet et al. 2015. Keras. `https://keras.io`.

Richard Dawkins. 2016. *The selfish gene*. Oxford university press.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Yuki Iwazaki. 2018. Deep ctr prediction in facebook ads. In *Proc. Annual Conference of JSAI*.

Akane Kanai. 2016. Sociality and classification: Reading gender, race, and class in a humorous meme. *Social Media + Society*, 2(4):2056305116672884.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Akshi Kumar and Geetanjali Garg. 2019. Sarc-M: Sarcasm Detection in Typo-graphic Memes. In *International Conference on Advances in Engineering Science Management & Technology (ICAESMT) - 2019, Uttaranchal University*, Dehradun, India, March.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Abel L. Peirson and E. Meltem Tolunay. 2018. Dank learning: Generating memes using deep neural networks. *CoRR*, abs/1806.04510.

Chhavi Sharma, Deepesh Bhageria, William Paka, Scott, Srinivas P Y K L, Amitava Das, Tanmoy Chakraborty, Viswanath Pulabaigari, and Björn Gambäck. 2020. SemEval-2020 Task 8: Memotion Analysis-The Visuo-Lingual Metaphor! In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain, Sep. Association for Computational Linguistics.

E. S. Smitha, S. Sendhilkumar, and G. S. Mahalaksmi. 2018. Meme classification using textual and visual features. In D. Jude Hemanth and S. Smys, editors, *Computational Vision and Bio Inspired Computing*, pages 1015–1031, Cham. Springer International Publishing.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2014. Going deeper with convolutions. *CoRR*, abs/1409.4842.

Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. 2016a. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016b. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Eric S. Tellez, Daniela Moctezuma, Sabino Miranda-Jiménez, and Mario Graff. 2018. An automated text categorization framework based on hyperparameter optimization. *Knowledge-Based Systems*, 149:110–123, 6.

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259.

Bohui Xia, Hiroyuki Seshime, Xueting Wang, and Toshihiko Yamasaki. 2020. Click-through rate prediction of online banners featuring multimodal analysis. *International Journal of Semantic Computing*, 14(01):71–91.