**Noname manuscript No.**
(will be inserted by the editor)

# Insert your title here

## Do you have a subtitle? If so, write it here

**Bobak Farzin**[1], **Nombre Apellidos2**[2]
[1]**bfarzin@gmail.com**
[2]**Universidad o lugar de trabajo**

Received: 15 June 2019 / Accepted: date

**Abstract** Insert your abstract here. Include keywords, PACS and mathematical subject classification numbers as needed.

**Keywords** First keyword · Second keyword · More

### 1 Introduction

Included citations as necessary: [Jr(2006)]

*Contribution* Our contribution with this work is..

### 2 Task and Dataset Description

The *Humor Analysis based on Humor Annotation (HAHA)* competition asked for analysis of two tasks in the Spanish language based on a coprpus of publicly collected data [Castro et al.(2018)Castro, Chiruzzo, Rosá, Garat, and Moncecchi]:

- **Task1: Humor Detection**:Detemine if a tweet is humorous. System ranking is based on F1 score which will balance precision and accuracy.
- **Task2: Funniness Score**:If humorous, what is the average humor rating of the tweet. System ranking is based on RMSE.

HAHA dataset includes labeled data for 24,000 tweets and a test set of 6,000 tweets (80%/20% train/test split.) Each record includes the raw tweet text (including accents and emoticons) and a True/False labels as well as a "Funninness Score" that is the average of the 1 to 5 start votes cast. Examples and data can be found on the CodaLab competition webpage[1].

Address(es) of author(s) should be given

[1] http://competitions.codalab.org/competitions/22194/

## 3 System Description

We generally follow the method of ULMFiT [**?**]

1. Train a language model (LM) on a large corpus of data
2. Fine-tune the LM based on the target task language data
3. Replace the final layer of the LM with a softmax or linear output layer and then fine-tune on the particular task at hand (classification or regression)

Below we will give more detail on each step and the parameters used to generate our system.

### 3.1 Data, Cleanup & Tokenization

### 3.2 Additional Data

For our initial training, we collected 475,143 tweets in the Spanish language using tweepy [Twe(2019)]. The frequency of terms, punctuation and vocabulary can be quite different from the standard Wikipedia corpus that is used to train.

We combined the labeled and un-labeled text data so that we have the largest corpus of language for our fine-tuning step.

### 3.3 Cleaning

We applied a list of default cleanup functions included in Fastai and added an additional one for this Twitter dataset.

- Add spaces between special chars (ie. `!!!` to `! ! !`)
- Remove useless spaces (remove more than 2 spaces in sequence)
- Replace repetition at the character level (ie. `grrrreat` becomes `g xxrep r 3 eat`)
- Replace repition at the word level (similar to above)
- Deal with ALL CAPS words replacing with a token and converting to lower case.
- **Addition:** Move all text onto a single line by replacing new-lines inside a tweet with a reserved word (ie. `\n` to `xxnl`)

The following example shows the application of this data cleaning to a single tweet:

```
Saber, entender y estar convencides que la frase \
#LaESILaDefendemosEntreTodes es nuestra linea es nuestro eje.\
#AlertaESI!!!!
Vamos por mas!!! e invitamos a todas aquellas personas que quieran \
se parte.
```

```
xxbos saber , entender y estar convencides que la frase \
# laesiladefendemosentretodes es nuestra linea es nuestro eje.\
xxnl  # alertaesi xxrep 4 ! xxnl vamos por mas ! ! ! e invitamos a \
todas aquellas personas que quieran se parte.
```

### 3.4 Tokenization

We used sentencepiece [Kudo and Richardson(2018)] to parse into sub-word units and reduce the possible out-of-vocabulary (OOV) terms in the data set. We selected a vocab size of 30,000 and used the byte-pair encoding (bpe) model.

## 4 Training and Results

### 4.1 LM Training and Fine-tuning

With our data cleaned and tokenized we move to training the LM using a 90/10 training/validation split. For the LM, we selected a AWD_LSTM [] model included in Fastai with QRNN units, 2304 hidden-states, 3 layers and a softmax layer on the end to predict the next-word. We tied the embedding weights on the encoder and decoder for training. We performed some simple tests with LSTM units and with a Transformer Language model and found all the models were similar in performance during LM training. We chose the QRNN units to speed up training. The model has about 60 million trainable parameters.

Both the LM and the Classifer models used label smoothing [**?**] of the flattened cross-entropy loss. With the classifier, this allowed us to train without gradual unfreezing of the layers as discussed in [Howard and Ruder(2018)] Parameters used for training all networks can be found in 1. For all the networks we applied a dropout multiplier that scales the dropout used throughout the network. We used the Adam optimizer with weight decay as indicated in the table.

Following the work of [**?**] we found the largest learning-rate that we could apply and then ran a one-cycle policy [] for a single epoch. We then ran subsequent training with lower learning rates, again indicated in 1.

### 4.2 Classification and Regression Fitting

The process and training parameters for both classification and regression are very similar. We build a model with either a classifer or regression head and the appropriate output and initialize those head weights randomly. We load the LM weights for the layers below and then train the head first before unfreezing the rest

**Table 1** LM Training Parameters

| Param | LM | Fine-Tune LM |
|---|---|---|
| Weight Decay | 0.1 | 0.1 |
| Dropout Mult | 1.0 | 1.0 |
| Learning Rate | 1 epoch at $5 * 10^{-3}$ | 5 epochs at $3 * 10^{-3}$ |
| Cont. Training | 15 epochs at $1 * 10^{-3}$ | 10 epochs at $1 * 10^{-4}$ |

of the layers and training further. We used differential learning rates as explained in [Howard and Ruder(2018)] for the layers so as not to lose the pre-training already done.

With the same learning rate and weight decay we apply a 5-fold cross-validation on the outputs and take the mean across the folds. We sample 20 random seeds 4.3 to find the best init for our gradient descent search and select the best validation F1 metric or MSE for use in our test submission.

#### *4.2.1 Classifier setup*

For the classifier, we have a softmax head and label smoothing loss. We over-sample the minority class to 50% 0 and 1 for better training using SMOTE [Chawla et al.(2002)Chawla, Bowyer, Hall, and Kegelmeyer].

#### *4.2.2 Regression setup*

For the regression, we fill all N/A labels with scores of 0. We add a linear head output and mean-squared-error (MSE) loss function.

**Table 2** Classificaiton and Regression Training Parameters

| Param | Value |
|---|---|
| Weight Decay | 0.1 |
| Dropout Mult | 0.7 |
| Learning Rate (Head) | 2 epochs at $1 * 10^{-2}$ |
| Cont. Training | 15 epochs with diff lr:$(1 * 10^{-3}/(2.6^4), 5 * 10^{-3})$ |

### 4.3 Random Seed as a Hyperparamter

The random seed sets the initial random weights of the head layer. This initialization determines where on the optimization surface you start your gradient descent and effects the final F1 metric you can achieve with your model. Across each of the 20 random seeds, we average the 5-folds and get a single F1 metric on the validation set. The histogram of outcomes is shown in **??** and covers a range from 0.820 to 0.825 in the validation set. We selected our single best random seed for the test submission. With more exploration, a better seed could likely be found. We only tried a single seed for the LM training but one could do a similar search with random seeds there and select the best down-stream seed similar to [**?**]

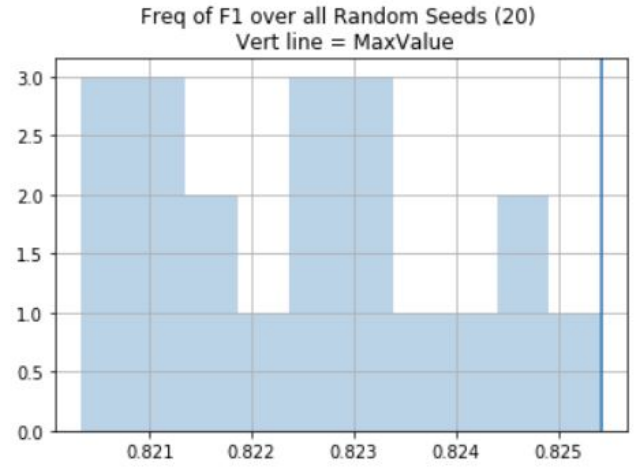


**Fig. 1** Histogram of F1 metric averaged across 5-fold metric

## 5 Conclusion

**Acknowledgements** If you'd like to thank anyone, place your comments here and remove the percent signs.

## References

[Twe(2019)] (2019) Tweepy: Twitter for python! URL http://github.com/tweepy/tweepy

[Castro et al.(2018)Castro, Chiruzzo, Rosá, Garat, and Moncecchi] Castro S, Chiruzzo L, Rosá A, Garat D, Moncecchi G (2018) A crowd-annotated spanish corpus for humor analysis. In: Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media, pp 7–11

[Chawla et al.(2002)Chawla, Bowyer, Hall, and Kegelmeyer] Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: Synthetic minority over-sampling technique. J Artif Int Res 16(1):321–357, URL http://dl.acm.org/citation.cfm?id=1622407.1622416

[Howard and Ruder(2018)] Howard J, Ruder S (2018) Fine-tuned language models for text classification. CoRR abs/1801.06146, URL http://arxiv.org/abs/1801.06146, 1801.06146

[Jr(2006)] Jr N (2006) My article

[Kudo and Richardson(2018)] Kudo T, Richardson J (2018) Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. CoRR abs/1808.06226, URL http://arxiv.org/abs/1808.06226, 1808.06226