Capstone Project: Using Deep Learning Models to Perform Natural Language Processing on a Small Dataset

Erica Racine
Udacity Machine Learning Engineer Nanodegree
July 17, 2019

I. Definition

Project Overview

Deep learning has received a lot of attention in the past few years for its impressive performance in many fields, including natural language processing and sentiment analysis.[1] An oft-cited reason for the expansion of the use of deep learning is the increase in the amount of data and computational power available.[2] However, the requirement of more data for methods of deep learning compared to many other machine learning algorithms is stated as a drawback.[3]

For this project I found a dataset of mostly Spanish paper reviews from a computing and informatics conference on which several machine learning algorithms had been used to perform natural language processing, but was reported to be too small for deep learning techniques[4]. I explored several models using deep learning techniques to determine if I could come close to or surpass the accuracy that was achieved using traditional machine learning techniques.

Problem Statement

---

[1] Collobert, Ronan et al. 2011. Natural Language Processing (Almost) From Scratch. *Journal of Machine Learning Research* 12, 2493-2537 http://www.jmlr.org/papers/volume12/collobert11a/collobert11a.pdf

[2] Thompson, Wayne. Deep Learning: The Confluence of Big Data, Big Models, Big Compute. Datanami website https://www.datanami.com/2019/01/10/deep-learning-the-confluence-of-big-data-big-models-big-compute/. January 10, 2019. Accessed July 2, 2019.

[3] Donges, Niklas. Pros and Cons of Neural Networks. Towards Data Science website https://towardsdatascience.com/hype-disadvantages-of-neural-networks-6af04904ba5b. April 17, 2018. Accessed June 26, 2019.

[4] Keith B, Fuentes E, Meneses C. 2017. A Hybrid Approach for Sentiment Analysis Applied to Paper Reviews. In *23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), Halifax 2017*. https://pdfs.semanticscholar.org/3409/6773742015f2d850a3dd778a2499bf3ec3c2.pdf?_ga=2.128911948.1328686511.1562115942-1358169566.1562115942

For this project, I would like to investigate whether deep learning methods could be used to adequately classify this data set. The accuracy scores of the algorithms that were tested in this study are listed in the paper, and I would like to see how deep learning models compare with these results. I plan to look into ways to improve the results of the deep learning models, such as hyperparameter tuning and pre-trained word vectors.

The steps will entail:

- Download json file, upload the dataset into pandas dataframe.
- Clean and preprocess the reviews data to remove punctuation, numbers, stopwords, etc.
- Tokenize and pad reviews data.
- Build a CNN and RNN and train and test both of them on the paper reviews dataset.
- Evaluate accuracy and tune hyperparameters of both models.
- Find and prepare Spanish pre-trained word embeddings.
- Train and evaluate the same CNN and RNN models using pre-trained word embeddings.
- For comparison, train and evaluate the CNN and RNN models using binary and ternary classification.

Metrics

I will be evaluating the performance of my model using accuracy scores. The authors of the paper ("A Hybrid Approach for Sentiment Analysis Applied to Paper Reviews" ["A Hybrid Approach"])[5] that introduced this dataset, evaluate their models by comparing the accuracy scores of their models to the accuracy scores of previously published models attempting sentiment classification.

Accuracy is calculated by the total number of datapoints calculated accurately:

[5] Keith B, Fuentes E, Meneses C. 2017. A Hybrid Approach for Sentiment Analysis Applied to Paper Reviews. In *23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), Halifax 2017*.
https://pdfs.semanticscholar.org/3409/6773742015f2d850a3dd778a2499bf3ec3c2.pdf?_ga=2.128911948.1328686511.1562115942-1358169566.1562115942

Accuracy = (TP+TN)/(TP+TN+FP+FN)

where: TP = True positive; FP = False positive; TN = True negative; FN = False negative[6]

Accuracy is a metric best used for balanced datasets. I calculated the skewness of the reviews data using Pearson's coefficient of skewness, to make sure that the dataset was not too imbalanced to evaluate with accuracy:

$$Sk_2 = \frac{3(\bar{X} - Md)}{s}$$

Pearson's second method for finding skewness.[7]

The Pearson's coefficient was 0.6530, denoting that this data was minimally skewed.

II Analysis

Data Exploration

The paper reviews dataset consists of 405 instances of reviews from an international conference on computing and informatics. The authors of "A Hybrid" paper found what they thought to be many inconsistencies between the text of the reviews and the corresponding score that was given. The authors rescored each review, based on the text of the review alone. The authors then implemented machine learning algorithms with the goal of being able to predict the score based on the text of the review.

The reviews are mostly in Spanish, and have the following attributes:

[6] Accuracy. In *Wikipedia*. Retrieved July 11, 2019, from https://en.wikipedia.org/wiki/Accuracy_and_precision

[7] https://www.statisticshowto.datasciencecentral.com/find-pearsons-coefficient-skewness-excel/

Timespan: Date of the conference from which the review was written. (The dataset contains reviews from conferences from four years.)

Paper ID: ID for each individual paper

Review ID: ID as it corresponds to each individual paper (the second review of a paper would have an ID of '2', most papers have two reviews.

Text: Detailed review by the original reviewers of the papers.

Remarks: An optional attribute of additional comments by the original reviewers of the papers

Language Language that the review was written in.

Orientation: The subjective rating of the review by the authors, using the same five point scale.

Evaluation: The original score given to the paper by the review.

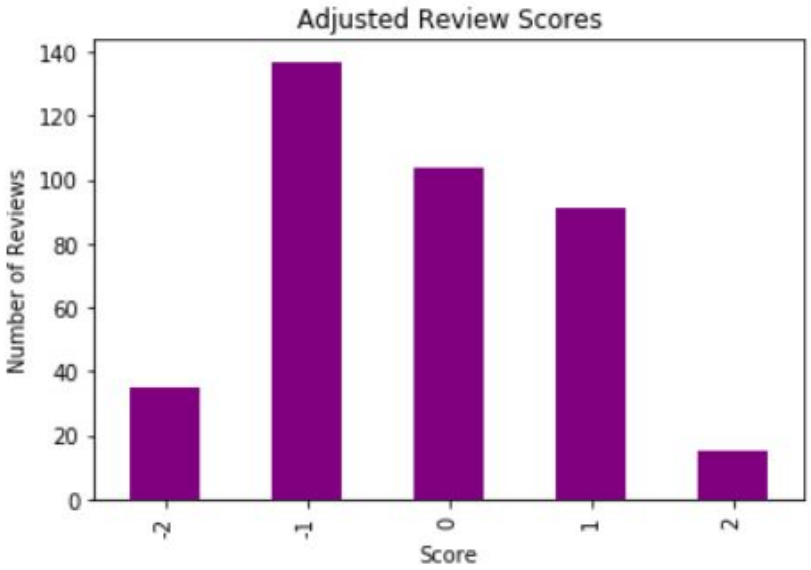
Both the orientation and evaluation attributes contain five classes of scores of values -2, -1, 0, 1, and 1, ranging from the most negative to the most positive.

Summary statistics include:

Minimum number of words:	3
Maximum number of words:	530
Average:	88.6
Standard Deviation:	69.76

Exploratory Visualization

After removing the seventeen instances of English language reviews, and the six instances of empty reviews, here is a visualization of the adjusted scores of the papers by the authors:



There is a notable discrepancy when compared to a visualization of the original scores of the papers (after the English language and empty reviews are removed):

** Since the models that were tested in the "A Hybrid Approach" paper were evaluated based on how well they were able to predict the *Orientation* of the review, the *Orientation* variable will also be the attribute that I will try to predict with my deep learning models.

Attributes and Techniques

I chose to build both a CNN (Convolutional Neural Network) and a RNN (Recurrent Neural Network) with LSTM (Long Short-Term Memory) to determine how well each would classify this dataset. CNNs and RNNs are two main types of deep learning neural networks. Although historically RNNs with LSTM have been a common choice for text classification[8], CNNs have been showing promising results.[9]

The architecture of a CNN is hierarchical, and can generally thought of to recognize patterns across space well. The architecture of RNNs on the other hand is sequential, and uses hidden layers from a previous step as an input to the next step, and can be generally thought of to recognize patterns across time well. Both CNNs and RNNs have been explored and

[8] Sinha, Nimesh Quick Implementation of LSTM for Sentimental Analysis Towards Data Science website https://towardsdatascience.com/understanding-lstm-and-its-quick-implementation-in-keras-for-sentiment-analysis-af410fd85b47 February 19, 2018. Accessed July 17, 2019.
[9] Ghelani, Shreya. Text Classification--RNNs or CNNs. Towards Data Science website https://towardsdatascience.com/text-classification-rnns-or-cnn-s-98c86a0dd361 June 1, 2019. Accessed July 17, 2019.

implemented successfully for a myriad of natural language processing (NLP) problems such as sentiment classification and language modelling, however there is no general consensus on the selection of either for any specific type of NLP tasks.[10] Therefore, I chose to build both models to see if either could effectively classify the paper reviews dataset.

Because the dataset I was using was small[11], I also explored the effectiveness of using pre-trained word embeddings.

Benchmark

For a benchmark, I used the model championed in the "A Hybrid Approach" paper, which is a hybrid model of supervised and unsupervised machine learning techniques. The accuracy scores of the hybrid model were:

5-point scale:	0.46 (+/- 0.5)
Ternary:	0.56 (+/- 0.04)
Binary	0.79 (+/- 0.05)

III Methodology

Data Preprocessing

Data preprocessing was a substantial part of the workflow for this project:

- The dataset was uploaded from a json file into a pandas dataframe and unpacked.
- The English language reviews and empty reviews were removed as well as columns with unnecessary attributes.
- The reviews data text was cleaned by removing stopwords, numbers, punctuation and other unwanted symbols.
- The reviews data was then tokenized and padded. The labels were red hot encoded.

---

[10] Yin, Kann, Yu & Schutze. 2017. Comparative Study of CNN and RNN for Natural Language Processing. https://arxiv.org/pdf/1702.01923.pdf .

[11] Mnasri, Maali. How to train word embeddings using small datasets? Towards Medium website https://medium.com/opla/how-to-train-word-embeddings-using-small-datasets-9ced58b58fde May 20,, 2019. Accessed July 17, 2019.

To implement the pre-trained word embeddings I downloaded a text file posted by Rachael Tatman[12] of ~ 1 million word embeddings that were trained on the Spanish Billion Words Corpus using word2vec. I used a tutorial [13] to create an embedding matrix to use in the embedding layer of both my CNN and RNN models.

Implementation

Guided by a Keras tutorial on constructing convolutional neural networks to be used with word embeddings[14], I constructed a simple CNN. The embedding layer had an input dimension of my vocabulary size, output dimension of 50 and an input length of 506 (the maximum length of my padded sequences). The embedding layer is followed by a three alternating layers of both a 1D convolutional layer and a max pooling layer. These six layers are followed by a dense layer activated by softmax. The model is compiled with categorical crossentropy loss for multi-class classification and optimized with rmsprop.

Using several tutorials as inspiration [151617] I created a RNN with LSTM. The embedding layer was the same as used in the CNN, but followed by Spatial Dropout to perform variational dropout, an LSTM layer with dropout and recurrent dropout to help with overfitting with the small dataset[18]. These layers are followed by a dense layer with soft max activation. The model is compiled with categorical crossentropy loss and the adam optimizer.

After evaluating these two models were tuned and evaluated, I replaced the word embeddings with the pre-trained word2vec word embeddings and tuned and evaluated again. Finally, I adapted both models to accept binary and ternary classes, adjusted the hyperparameters and evaluated again.

---

[12] https://www.kaggle.com/rtatman/pretrained-word-vectors-for-spanish
[13] https://realpython.com/python-keras-text-classification/
[14] https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html
[15] https://medium.com/@shivambansal36/language-modelling-text-generation-using-lstms-deep-learning-for-nlp-ed36b224b275
[16] https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/
[17] https://towardsdatascience.com/multi-class-text-classification-with-lstm-1590bee1bd17
[18] https://machinelearningmastery.com/how-to-reduce-overfitting-with-dropout-regularization-in-keras/

Refinement

There was a lot of refinement that took place in both the preprocessing and implementation stages of this project.

In preprocessing, I went through several approaches to cleaning the text. Originally I had more complex review cleaning function, but found the filters hyperparameter in Tokenizer() to be just as effective. I added punctuation marks unique to the Spanish language and played with transforming the text to lower case. Leaving the upper case letters increased the number of unique tokens found in the reviews text from 6,237 to 6,987. However the increased number of unique tokens did not seem to increase my accuracy in my original CNN and RNN models. However leave the upper case words in the models using embedding matrices did increase the number of words matched in my data to the embedding matrix from around 80% to about 95%, so I left the upper case words in the dataset for those models.

I did a lot of hyperparameter tuning in the models, especially with the five-point classification models. With the CNN model I tried different numbers of filters, kernel, types of pooling layers, activation functions, loss functions and optimizers. I was stuck with an accuracy score of 32.5% for a long time, but after trying many different hyperparameter tunings was able to consistently achieve accuracy scores of over 35%.

With the RNN model I tried different levels of dropout, different numbers of LTSM memory cells, different activation functions and loss functions and different ordering of the layers. My RNN continued to overfit however, and did not outperform my CNN.

IV Results

Model Evaluation and Validation

The final model chosen was the CNN without pre-trained word embeddings, because it surpassed the other models.

	5-Class	Ternary	Binary
CNN	~ 39.0 %	~40.3%	~75.0%
RNN	~ 31.2 %	~35.1%	~66.1%
CNN/W2V	~ 32.5 %	~37.7%	~62.2%
RNN/W2V	~ 33.8 %	~37.7%	~66.1%

The final architecture of the model was:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 506, 50)           2500000
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 502, 64)           16064
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 251, 64)           0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 247, 64)           20544
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 123, 64)           0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 119, 64)           20544
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 325
=================================================================
Total params: 2,557,477
Trainable params: 2,557,477
Non-trainable params: 0
_________________________________________________________________
```

A validation set was used during testing to evaluate the models. Minor adjustments were made to the 5-class classification models to make them compatible with binary and ternary datasets.

Justification

Although the CNN came fairly close to the benchmark results in some instances, particularly the binary classification, the final model did not beat any of the benchmark results:

	5-Class	Ternary	Binary
Benchmark	~46.0%	~56.0%	~79.0%

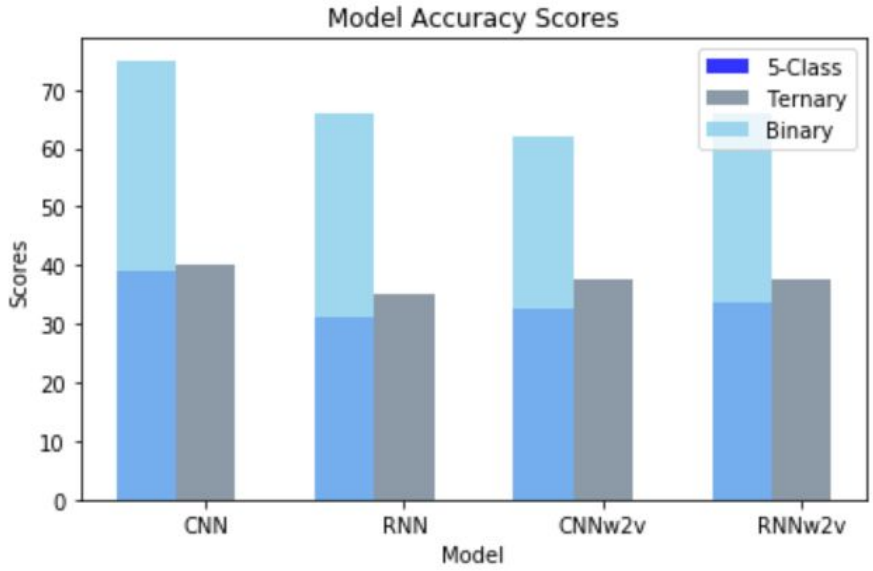
The RNN with the word2vec pre-trained word embeddings had a higher accuracy score than the CNN in two of the classifications, and effectively tied it in the third. Interestingly though, although I was hoping the pre-trained word embeddings would improve accuracy, the CNN without the pre-trained word embeddings beat all the other models.

Although the classification problem here wasn't solved, the performance of the CNN indicates strong potential for successful sentiment analysis on smaller datasets with further refinement in the future.

V. Conclusion

Free-Form Visualization

The accuracy scores of the four models for each number of classes:



Reflection

In this project I attempted to use deep learning models to perform natural language processing on a dataset that was previously declared too small to do so. I downloaded the json file and uploaded the dataset to a dataframe. I experimented with different preprocessing techniques, including ways to clean the data and remove stopwords and punctuation. I tokenized and padded the reviews data to prepare it for embedding. I built a CNN and an RNN, and trained, tuned and evaluated both models. I downloaded a text file of pre-trained word2vec word embeddings, created an embedding matrix, and used it in the embedding layers of both the CNN and RNN. Finally, I adjusted the CNN and RNN models to accept binary and ternary classes and trained and evaluated those models. Although I neither matched nor surpassed the benchmark accuracy scores, I came significantly close in some instances and have laid a foundation on which with more refinement the CNN model in the future will be able to surpass the benchmark scores.

This project was extremely interesting. I performed countless hours of research, took a few additional online courses in natural language processing, and spent weeks exploring CNNs and RNNs. Being new to the NLP field, I am amazed at how vast it is. There are many different approaches that I have yet to take with this project, and I will continue working on it, but since there is a deadline I will end it here.

Improvement

There are many additional methods I could and plan to try to improve the results of this classification problem. For example, I would like to try the BERT language model[19] with this dataset. I would also like to experiment with different pre-trained word embeddings, like those available through spacy[20]. Finding Spanish language pre-trained word embeddings was a challenge. Several of the links that I did find were broken or were difficult to download and there was not a lot of support available. I would love to see expanded options for languages other than English in the future.

[19] https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
[20] https://spacy.io/