RBCxImperial

PhD Project TASK-B

August 10, 2020

Contents

1 Introduction

This report is the description of the process design and workflow I implemented in order to help ABC Relations Inc. solve their problems. The solution is designed as a *microservice* with a user friendly interface. The user will have to upload a dataset with text news. The system processes the content of the text according to the required criteria. It filters the inputs and extracts real news and drops fake news. From each articles it outputs relevant people and organization mention in the article as well as a brief summary of the story. In order to address any issues relevant to the client of ABC for each article it is given a broad topic of story so that specified personnel can address the issue in the most efficient way. After outlining the technical details of frameworks used I deeply focus on future works and improvements.

2 Data Description and Preprocesing

The dataset used is downloaded from `https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset`. It consists of a mixed text database containing a total of 44898 entries divided in 21417 real news and 23481 fake news. After deleting duplicate articles I obtain 21192 unique real news and 17454 unique fake news. The original dataset divides the news into the following topics: *News, Politics, Left-News, Government News, US News, World News*. However, for the purpose of this project I will not be taking into consideration these divisions. The preprocess of the raw data consists of cleaning the data. The main steps of such algorithm are explained below:

- Punctuation is stripped away from the articles
- Tokenization splits sentences of an article into individual elements of interest (e.g. words)
- Linguistic Roots – lemmatizing transforms tokens into their root
- Stop-word removal removes words that are not useful to the data analysis.
- Multi-word tokens (bigrams and trigrams) creates a sequence of two or three tokens which individually have less meaning.

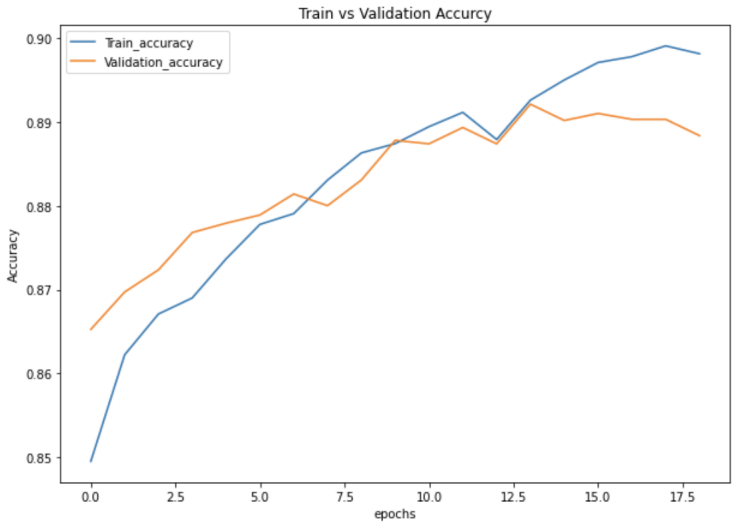
Out of the many Natural Languages Processing packages available I decided to use the *gensim* library. I believe that it is the optimal solution for preparing text which will be used in advanced machine learning algorithms. The stop words list is based on the *Spacy* library which offers a wider list if compared to *NLTK*. I extended the list by using an existing database.

3 Fake News Classifier

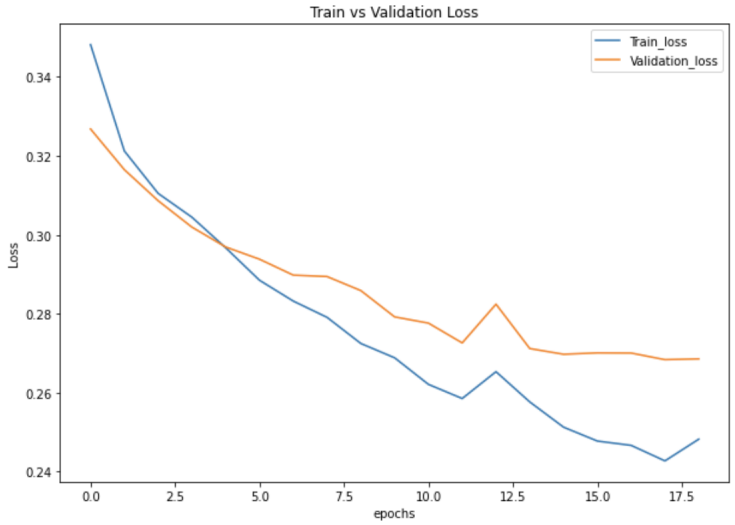
The first step of the project is to detect if a new piece of news is real or fake. In order to tackle this issue I built a *Fake News Classifier*. The embedding matrix I am using for this kind of problem is based on the *Glove* embedding matrix. It is an unsupervised learning algorithm developed by Stanford University that uses a similar approach to Word2Vec. The object of the matrix is to represent each word in our vocabulary (word index dictionary) into a *D-dimensional* vector. Such embedding matrix is used to generated the Embedding layer of the classifier model. This layer transforms each document (list of integers) into a tensor with a fixed dimension using the embedding matrix. Since I am using a pretrained embedding matrix the weights associated with it are frozen and not trainable. The model consists of a 2-layer LSTM architecture. The first layer keeps track of the factor of information which is fed to the second layer. This layer does not return the original sequence but only the last vector, which has embedded in itself all the information from the previous sequences. To code such architecture I use tensorflow functional API. This means that each layer is applied to the tensor as a function. At each step the input tensor is fed to the next layer which itself generates a new tensor used as a new input. The parameters associated with the embedding layer cannot be updated, whereas the LSTM parameters are trained with each iteration. In order to avoid overfitting I am using two callback functions: Early Stopping and Reduce on Plateau. In the *Proof of Concept* notebook I chose the optimal hyperparameters which are used in the application. The model is trained using the dataset mentioned above before deployment. This means that the user is able to get accurate result without re-training the model. In other words, it is not needed a big dataset as the system does only inference on the input text. If the user desires there is the possibility to retrain the model on a specific dataset. The architecture summary and the models performance are attached below:

```
Model: "functional_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 500)]             0
_________________________________________________________________
embedding_1 (Embedding)      (None, 500, 100)          2000000
_________________________________________________________________
lstm_4 (LSTM)                (None, 500, 64)           42240
_________________________________________________________________
lstm_5 (LSTM)                (None, 32)                12416
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 66
=================================================================
Total params: 2,054,722
Trainable params: 54,722
Non-trainable params: 2,000,000
_________________________________________________________________
```

(a) Model Summary



(b) Accuracy Performance



(c) Loss Performance

Figure 1: Functional Model

4 Topic Model

In order to give background knowledge about the article I estimate a topic model using a Latent Dirichlet Allocation (LDA). The topics are then used as labels for a topic classifier. This model is trained on the articles from the original True dataset. The *LDA* implementation I am using is MALLET, a java application developed by University of Massachusetts Amherst. Such application can be simply implemented in Python thanks to a *Gensim* wrapper. Gibbs Sampling implemented in MALLET makes the model more precised with respect to Gensim Mallet. From the same *Gensim* library I am using 'corpora.Dictionary' and 'doc2bow' to generate the dictionary and the corpus respectively. Topic Modelling Algorithm are unpervised learning models which attempt to cluster together texts of the same topic. Important hyperparameter in this kind of problems is the number of topics. In order to find the optimal number of topics I generate several models, I then compute their coherence value and compare in order to find the right model. Ideally, we want to choose the number of topics before the line flattens. After that point, an extra topic is not adding any more useful information. This step is not implemented in the final application as the number of topics is specified based on the results of the *Proof of Concept.*

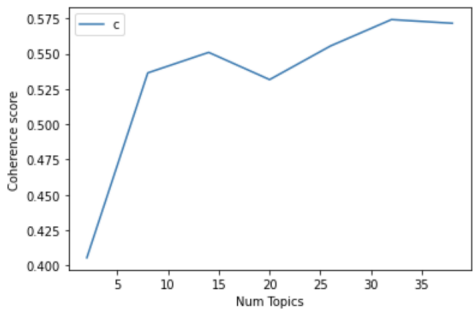


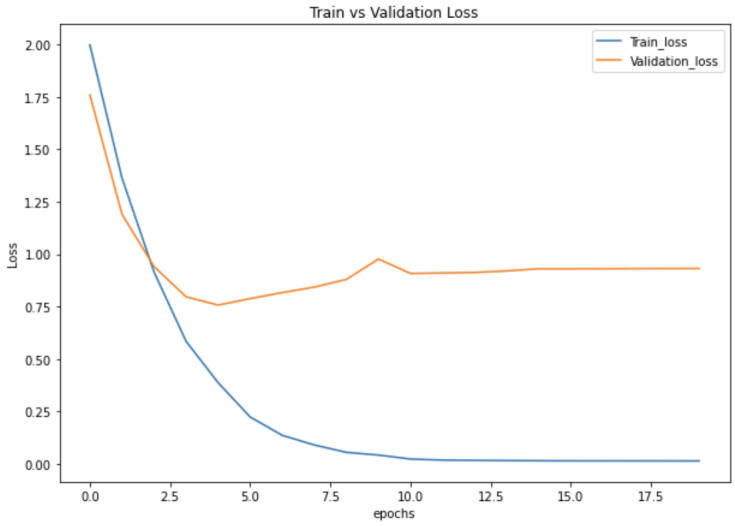
Figure 2: Coherence Value vs Number of Topics

The results of the *LDA* are used as the target variable for the *Topic Classifier*. The topic classifier is trained to predict the most relevant topic in each article. For this model I am using a similar algorithm by adjusting the loss function and activation function to the multiclass nature of the problem. However, I am changing the architecture. In this case the *LSTM* layers are going to process the data sequentially. The input is fed to a trainable embedding layer which means that the embedding matrix is not going to be frozen and it can be updated during the training through gradient descent. In this case I am using a bidirectional *LSTM* architecture. The input sequences generated by the input layer through two difference *LSTM* layers in two directions. It is going to process the sequence from the first word to the last words and from the last word to the first. This layer outputs two

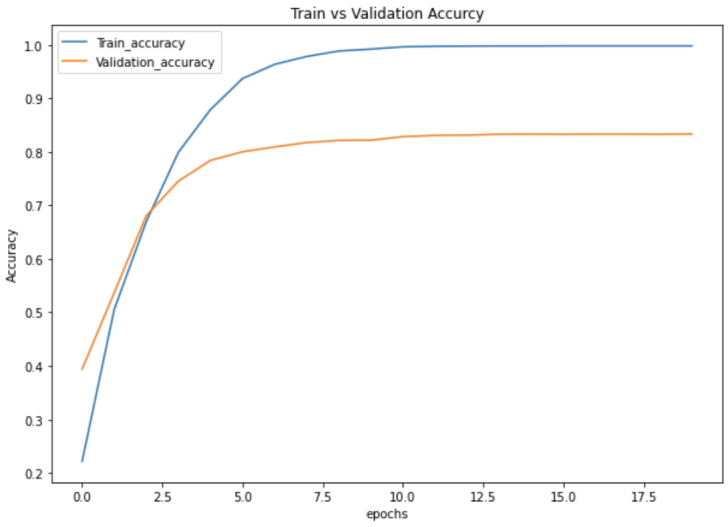
vectors with different information from the same initial sequence. These two vectors are concatenated and fed to the *Dense* which outputs the class labels. Again, both models are pre-trained in order to allow the user to make accurate inference even on simple datasets. The architecture summary and the models performance are attached below:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, None, 100)         2000000
_________________________________________________________________
bidirectional (Bidirectional (None, 200)               160800
_________________________________________________________________
dense_3 (Dense)              (None, 100)               20100
_________________________________________________________________
dense_4 (Dense)              (None, 8)                 808
=================================================================
Total params: 2,181,708
Trainable params: 2,181,708
Non-trainable params: 0
```

(a) Model Summary



(b) Accuracy Performance



(c) Loss Performance

Figure 3: Sequential Model

5 Summary and Named-Entity Recognition

The model I am using to extract summary of the text article is based on Raffel, Shazeer, Roberts, Lee, Narang, Matena, Zhou, Li and Liu (2019) Text-to-Text Transfer Transformer, developed by Google AI lab. The model is the result of a self training algorithm on a dataset created ad-hoc for the problem. The main difference with other similar models (ULMFit from Fast.ai, BERT by Google) is that it has been design to solve text-to-text problem. It is able to return sequences of text instead of a simple class labels. Due to time constrains I am not able to fine tune the model on my specific dataset. However, I am going to discuss this matter further in the *Future Works* section of this report. Named entity are part of text that can be associated with an object from the real word. For the purpose of this problem I am going to extract specific names of individuals and organization. I am using the *Spacy* built-in function to extract all these entities in the articles. The results are ranked by frequency and the top 3 entities for both individuals and organizations are returned as outputs.

6 Future Works and Improvements

The major challenge I faced in this project was finding the way to implement an Abstractive Text model which able to summarize through unsupervised learning algorithms. Abstractive models move away from the traditional extractive models and are able to output a summary which is correct not only in its meaning but also follows grammar rules. In other words, the result of Abstractive models can be correlated to the way humans analyze text. This approach moves away from classical extractive models which are based on statistical models that return, for example, subsets of text which the model consider the most relevant. However, this output is not in line with what a human being would expect in a text. During my academic years I was introduced to the theory behind Generative Adversarial Networks (GAN). Developed and introduced by Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville and Bengio (2014) in their paper *Generative Adversarial Networks* GAN are generative model that are able to generate new content from its own input. Widely used in various sector from image recognition to the movie industry GANs have also been applied to NLP problems. Liu, Lu, Yang, Qu, Zhu and Li (2017) proposed a GAN architecture where they train both generative model and a discriminant model which given text as input returns its abstractive summary. Future works for this project would consist of implementing a GAN architecture specific on this dataset. Despite the 5T model is able to provide correct results, a dataset-specific model will also provide accurate result which can lead to better actions for those who are using the system.

Furthermore, the two text classifiers used in this project can be further improved. Due to both time and computational stanfnot been able to optimize each single hyperparameter used in the models.

Ultimately, further efforts should be put in a system that based on the content of the articles gives suggestions on how managers should react to the content of the article. However, this solution is very firm-specific and it needs expertise from the sector the company operates in. However, this could be implemented by clustering articles on same topic based on their sentiment and polarity. If the elements of all the articles are consistent this should be flagged and the to the office in charge of that topic. For example, if the systems returns a flag on articles for a specific client of the company and they all suggest a negative sentiment some action must be taken in order to understand and adress the issue further.

7 Bibliography

References

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014), 'Generative adversarial networks'.

Liu, L., Lu, Y., Yang, M., Qu, Q., Zhu, J. and Li, H. (2017), 'Generative adversarial network for abstractive text summarization'.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P. J. (2019), 'Exploring the limits of transfer learning with a unified text-to-text transformer'.