CSE 472 : SOCIAL MEDIA MINING

Project II : Fake News Classification using textual entailment approach.

Vishal Kumar
Arizona State University
1215200480
vkumar47@asu.ed

Shubham Vipul Majmudar
Arizona State University
1215200298
smajmuda@asu.ed

Introduction:

The problem description tells about finding a relationship between two pair of sentences where given the title of a fake news, we need to find whether the title of the coming news article is in agreement with the previous article or is in disagreement or is completely unrelated to it. This is a type of classification task in which we need to classify a pair of sentences into 1 of the 3 categories. The general term for such task is called a textual entailment problem.

Our approach for this problem is to find the logical similarity between the pair of sentences using the word embedding of words in a vector space, specifically, GloVe: Global Vectors for Word Representation by Jeffrey Pennington. And then using a neural network to learn the relationship between 2 sentences.

Before moving on to the next sections, we provide a certain terminology, which are essential to understand the approach.

Terminology:

- One-Hot-vector: Vector representation for a number of classes in a classification problem, each dimension in the vector corresponds to a class. For a given class, the one-hot-vector will contain one '1' corresponding to that class and all other elements will be zero. One-hot-vectors are required cause there exist no such ordinal relationship between categorical variables, and the one-hot-vectors thus provide us a confidence in each class.

- Cosine similarity: cosine similarity is the dot product of two vectors divided by the multiplication of each of their magnitudes—it thus ranges from 0 to 1.

- Word-embeddings: word-embeddings or word-vector embeddings are vector representations of words that make syntactic sense; the dimensions of these vectors can be customized based on the space-information trade-off and similarity metrics (cosine similarity) can be used to verify the correctness of these vectors.

Preprocessing and Feature Engineering:

For observation and pre-processing of the data, we used the 'pandas'[4] library for Python. This whole project was done using Python 3.4. And we used the English translation of the sentences. Thus, following things were used from the dataset –

1. From 'train.csv': 'title1_en', 'title2_en' and 'label'.
2. From 'validation.csv': 'title1_en', 'title2_en' and 'label'.
3. From 'test.csv': 'ids', 'title1_en' and 'title2_en'.

Dataset problems and solutions: Following problems were observed in the 3 given files –

1. There were observations in 'train.csv' for which extra column values were present i.e. more than 8 entries were present, to overcome this, we only read the required columns from a file instead of reading the whole csv, example: we only read 0^{th}, 5^{th} and 6^{th} column from test.csv.
2. All three files ('train.csv', 'validation.csv' & 'test.csv') contains certain observations in which among one of the required column, the value was not present, example – absence of sentence1

or label. To overcome this, we deleted all such observations from the dataset.

3. File ‘train.csv’ had 7 observation for which label was not appropriate, instead of labels sentences were present, to overcome this, we first lower-cased the whole dataset and filtered removed the observations for which the label was different than one of the three required values.

Next, we converted our sentences pair into an appropriate feature space, before this we converted the provided labels into their One-hot-vector representation.

Feature Engineering:

As our approach consists of using a neural network, we can not feed the words directly in to the network and for this, we converted the sentences into a vector of integers where each integral value corresponds to a word. To do this, we used the Tokenizer provided by the Keras library[1].

Each sentence is tokenized based on the default delimiters (includes most special characters) and each token is assigned an integer value based on its index in a dictionary. Eg, “The quick, brown fox” becomes [‘the’,’quick’,’brown’,’fox’] after split, which in turn is converted to integer values based on their index, say [3, 16, 298, 122].
This is the feature vector for a sentence that will be used as input to the model.

But sentences have variable lengths, and our feature vector needs to be standardized(the input length of the neural network is fixed), the length of each vector is set to cover a vast majority of the dataset that has length within this—those sentences having words less than this are padded with empty strings for the rest of the vector while those with larger length are truncated.

Lately, a huge number of natural language processing applications make use of representing words in a fixed number of dimensions instead of bag of words approach, this is a good practice as bag-of -words approach contains a large vector representation which are mainly sparse in nature[2]. Therefore we are using the word embedding approach, where the words are represented by dense vector in a continuous vector space.

After this, we made two different embedding matrices, 1 for the words present in the group of sentence 1 and other for the sentence 2 group. These word embedding matrices are 2D in nature and consists the embedding for a word, where the word is identified using the index. Example: in the embedding matrix the value(a list) present at the index 123 is the embedding for the word corresponding to the value 123 created by our tokenizer. We used the GloVe word embeddings to perform this operation.

- GloVe[3] word embeddings:

GloVe is an unsupervised learning algorithm developed in Stanford University to convert words to vector representations. The word representations are learnt not through neural networks but using gradient descent on a function consisting of the log probability of co-occurence. GloVe embeddings are applied on the input before using it as a network-layer while training our model.

Model Description:

Our approach consist of making a multi-input neural network, for this we used the Keras functional api. The motivation and information about embeddings was obtained from a tutorial by Jason Brownlee[5].

The model consists of sending the padded vector representation of the sentences into a group of layers, both sentences are parallelly and separately to two separate, identical group of layers, this group consist of 1 input layer, which takes the vector representation of the sentence, 2nd layer is the embedding layer, which makes use of the embedding representation of that layer, 3rd layer is used to flatten the output of the 2nd layer as we wished to connect the output of the 2nd layer to a dense layer and to do so, the

flattening of the output is required, 4th layer is a dense layer which is again connected to the final 5th Dense layer.

The output from these 2 set of layers in than concatenated and fed to a shallow neural network, which contains a dense layer and an output layer of size 3 giving us the confidence in each class label.

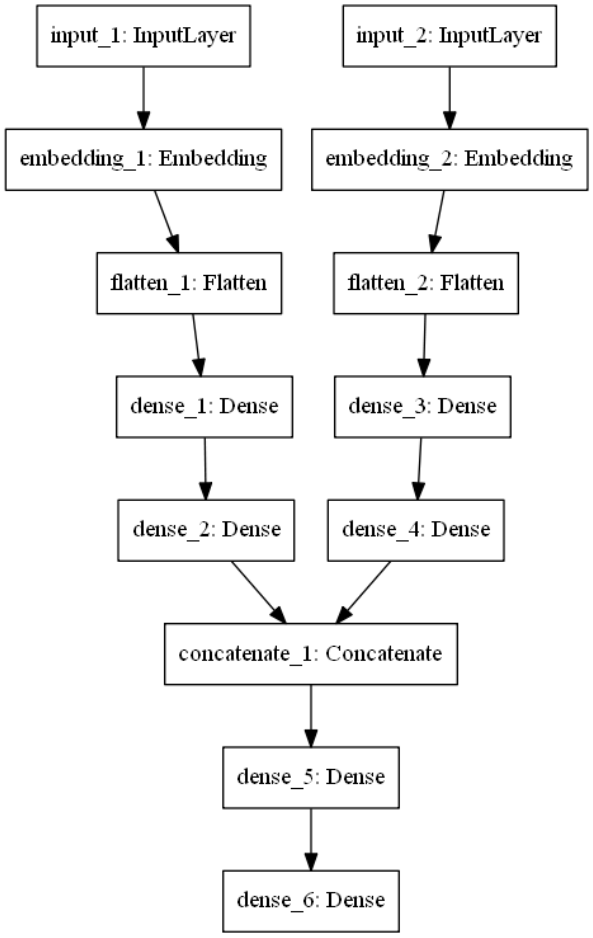

Figure 1: Pictorial representation of the model.

Results:

After tuning the hyperparameters, it was found that the best accuracy among the experiments done was achieved with a hidden layer size of 500 for previous set of layer and a hidden layer of size 50 in the final set of layer with a learning rate of 0.005 with a 'rho' value of 0.9 in the Root Mean Square optimizer of the model. The training was done in batch-sizes of 512 for a total of 40 epochs over the set 19228 training samples.

The following accuracies were observed on the Training and testing dataset.

1. Training Dataset: The accuracy on the 40th epoch: 95.89%.
   The training accuracy may vary on training model again even with the same hyper-parameters.
2. Test Dataset: 65.03%

The trained model is saved in Hierarchical Data Format file name 'smm_project2_model.h5' and the output generated from the test.csv file are saved in the file named 'submission.csv'.

id	label
41937	unrelated
117063	unrelated
306750	agreed
5069	unrelated

Table 1: Sample from 'submission.csv'.

The output generated has less entries as compared to test.csv as some entries had NULL values.

Note: We have made different files for model training and testing. The files are as follow:

1. project2.py: Used to train and save the model.
2. validation.py: Used to check the accuracy on validation set using the model generated by project2.py.
3. test_prediction.py: Used to generate the labels from test.csv using model generated by project.py.

Additional, information is in the file 'Readme.txt'

References:

[1] https://keras.io/preprocessing/text/
[2] Mishra, Anish. "Deep Learning Techniques in Textual Entailment." (2018).
[3]Stanford GloVe project (https://nlp.stanford.edu/projects/glove/)
[4] https://pandas.pydata.org/
[5] 'How to Use Word Embedding Layers for Deep Learning with Keras', Jason Brownlee. https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/