Identify online patient conversations

Developer: M R Abhishek
Language: Python

Steps

- Remove unnecessary columns from data frame.
- Preprocess text.
- Feature generation.
- Evaluation metric selection.
- Model Architecture:
    - Tf-Idf with Neural Network
    - Embeddings with LSTM
- Hyper parameter tuning.
- Ensemble model(92.99 on test data).

Preprocess Data - Remove unnecessary columns

- The dataset consists of different columns of information. But, the patient can be identified only through conversation text.
- The patient_tag doesn’t depend on source, time at which it is posted. At the end of the day, the output depends only on the patient’s conversation.
- All the columns except ‘TRANS_CONV_TEXT’, are removed from the dataframe.
- Now the task is to classify the conversation into Yes or No.
- Also, there is only one row with NaN. This row has been removed.
- Later, some extra features were added. That will be discussed later.

Preprocess Text

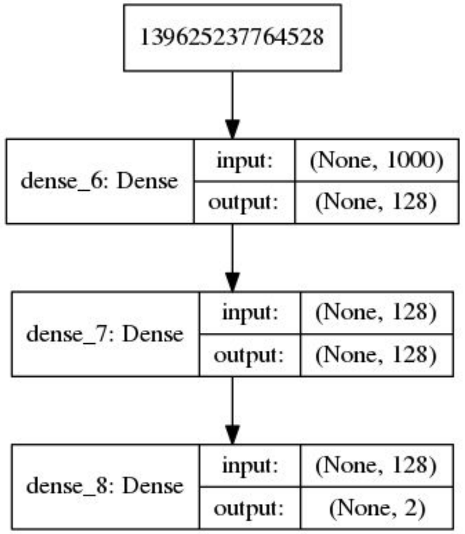
- **Punctuations** are removed.
- **Lemmatization**: This will convert a word into its root word. This process remove inflectional endings and return the base or dictionary form of the word. We may lose some information, but this technique will work in NLP tasks.
- **Removal of stopwords**: These words don’t contribute to the prediction. Hence, removal of stopwords doesn’t harm the performance of the model.

Feature Generation

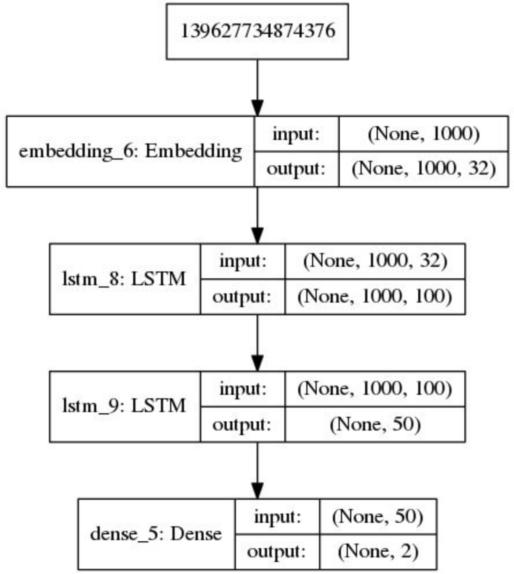
- **Approach 1**: Top 1000 words are considered. The text conversation is converted into Tf-Idf vector which is used for training. Sklearn TfidfVectorizer is used to convert text into vector.
- **Approach 2**: Each word is converted into an unique integer and Embeddings are learned using Embedding layer. As the domain is different from normal conversations, this way of learning word-embeddings is task specific and performs well. Keras is used in this approach.

Model: Architecture details

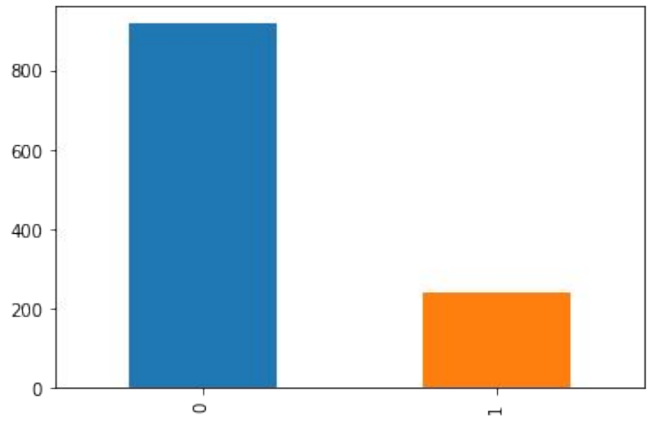
Approach 1



Approach 2



Evaluation metric



- The data is imbalanced. So, evaluating a model based on accuracy is not a good choice in this case.
- In this case, **ROC AUC** is a better metric to evaluate a model.

Hyperparameter tuning

- A lot of hyperparameter tuning has been done. Some of the parameters that were tuned are learning rate, optimizer, number of hidden layers, type of layer, number of neurons in hidden layers, dropout, number of iterations and activation function.
- 20% of train data is used as validation set. This is used while training, which helped the model work better.
- Also, the data is shuffled to make sure that there won’t be any bias while training.

Extra features

- Features added to the input vector:
  - The information of Source(YouTube, Facebook, Blogs,..etc) is added to the vector.
  - Length of the text.
  - Number of Nouns in the conversation text.
  - Length of the title.
  - The day is divided into four parts and this information is added to the vector.
    - 12 am - 6 am
    - 6 am - 12 pm
    - 12 pm - 6 pm
    - 6 pm - 12 am
- There was no improvement in the model after adding these features. So, the above features were removed.

Results

- Validation accuracy is ~ 87 - 90%. Validation AUC is ~ 97 - 98.5.
- The scores achieved on test set using variants of Approach 1 are 89.67 and 91.24. The score achieved using Approach 2 is 92.47.
- At the end, an **Ensemble** model is created using the predictions. Most occurring output from the above 3 files is considered as the final prediction. This gave a score of **92.99** on test data. This is the highest score achieved.

Improvements

- Dataset can be increased by using oversampling techniques or collecting more data.
- Feature generation step can be improved (if time permits).
- Try more different architectures.
- Ensembling different models instead of predictions.