MBTI type Prediction and Analysis

Pranav Kumar Sivakumar
CU Boulder
prsi6835@colorado.edu

Abstract

This project presents a deep learning model implemented on several individual's text data, which classifies them by predicting their personality type.

1 Introduction and Background

1.1 objective

To develop an efficient system that predicts MBTI (Myers Briggs Type Indicator) personality types of persons based on their writings/posts. According to its system everyone is classified into one of sixteen distinct personality types across four axis,
Introversion(I) – Extroversion(E)
Intuition(N) – Sensing(S)
Thinking(T) – Feeling(F)
Judging(J) – Perceiving(P)
Examples types look like INTP, ENTJ, ISTJ, ESFP...

1.2 MBTI Personality

In developing the Myers-Briggs Type Indicator [1], Myers et al. addressed the two related goals in the developments and application of the MBTI instrument:

The identification of basic preferences of each of the four dichotomies specified or implicit in Jung's theory. The identification and description of the 16 distinctive personality types that result from the interactions among the preferences.

Favorite world: Do you prefer to focus on the outer world or on your own inner world? This is called Extraversion (E) or Introversion (I).

Information: Do you prefer to focus on the basic information you take in or do you prefer to interpret and add meaning? This is called Sensing (S) or Intuition (N).

Decisions: When making decisions, do you prefer to first look at logic and consistency or first look at the people and special circumstances? This is called Thinking (T) or Feeling (F).

Structure: In dealing with the outside world, do you prefer to get things decided or do you prefer to stay open to new information and options? This is called Judging (J) or Perceiving (P).

Your Personality-Type is decided with certain questions on situations based on above details, which can be expressed as a code with four letters like the ones in the figure 1.

1.3 LSTM Networks

Neural networks have become increasingly popular for the task of text classification, Whereas feed-forward networks only exploit a fixed context length to predict the next word of a sequence, practically,

.

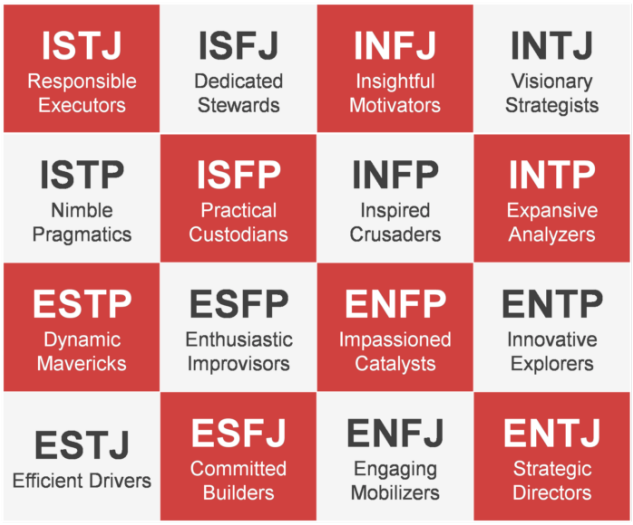

Figure 1: All MBTI Personality types

standard recurrent neural networks cannot take into account all of the predecessor words and therefore are unlikely to show the full potential of recurrent models. These problems are addressed by a the Long Short-Term Memory neural network architecture.

Long Short Term Memory networks – usually called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter et al. (1997)[2]. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell is responsible for "remembering" values over arbitrary time intervals; hence the word "memory" in LSTM. LSTMs were developed to deal with the exploding and vanishing gradient problem when training traditional RNNs. Relative insensitivity to gap length gives an advantage to LSTM over alternative RNNs and hidden Markov models

In this, I have implemented an LSTM-based deep-learning architecture for the above mentioned classification problem.

2 Dataset

The dataset was hosted in Kaggle, it contains over 8600 rows of data and each row has a person’s:

- Type (This person’s 4 letter MBTI code/type)
- A section of each of the last 50 things they have posted (Each entry separated by "|||" (3 pipe characters))

3 Model

3.1 Preprocessing

Once the text and labels are loaded from the dataset using pandas framework, the 16 labels are transformed into one-hot encoding(categorical labels encoded into a binary representation where one column denoting that label is with ’1’ with others as ’0’). All the text data(posts) are converted to lowercase. All the punctuations, URLs and links are removed using regular expressions. The emojis/emoticons are left remained. Then a look-up table was created (i.e., dictionary in Python) for the vocabulary of the dataset, (which is already sorted decreasing based on the respective frequencies of the words) which maps all the words to integers.All the words in the posts are then converted into lists of integers according to the vocabulary.

Since the lengths of the posts are varying, made all of them to a fixed size (say 1000 words) and padded those posts with zeros if length of it is less than the size. This part is essential for neural

networks. Later, the computed features split into sets for training, validation, testing in the ratio of 8:1:1.

3.2 Neural Network Model

The model architecture is illustrated in figure 2 below. The RNN implemented here consists of an LSTM layer of 256 cells and then followed by two dense layers which finally gives away the predicted class as the output. The first dense layer classifies the internal representation obtained from the LSTM layer into a predicted class that is contained in 16 cells. The final Dense layer is meant to be an output layer with softmax activation. The reason behind the dense layer after the LSTM layer is that many frameworks will give away only the internal state h as output, so the dimensionality of this output only equals to the number of units/cells, which is probably not the dimensionality of our desired target. That's why we have to specify a last dense layer, which merely correspond to this equation: $y_t = W * h_t$, where y is the logits we have to pass in a softmax layer, and W the weight connection matrix of the last layer. Batches are generated using a separate function for all the sets. The graph is built (using Tensorflow) with the required variables and placeholders with cost function as mean squared error and optimizer as Adam optimizer.

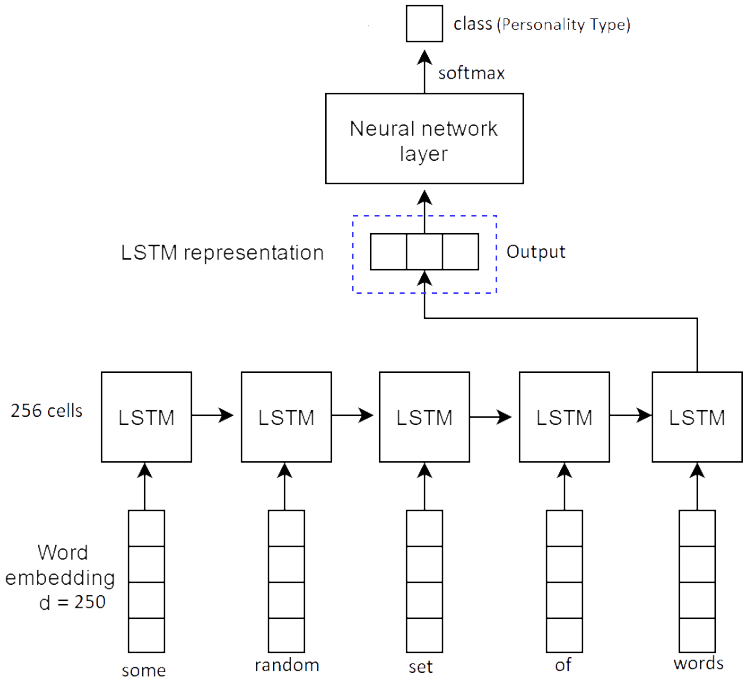

Figure 2: Model Architecture.

3.3 Embeddings

The word embeddings were randomly generated for look-up with the help of a random-uniform function that gives output values in the range of -1 to 1. The dimensionality of the embeddings are set at 250.

Both pre-trained embeddings from Word2Vec and GloVe embeddings didn't contain most of the words in our vocabulary(about 67% of our dataset). Many words in our dataset were mostly composed of texting-slang words and abbreviations, numbers and open-compound words. So, these embeddings weren't preferred.

3.4 Other Parameters

Other parameters that were set are: the learning rate was at 0.01, batch size was 256, the number of epochs was 3 and the cost function was mean-squared error.

4 Experiments and Results

The best accuracy that was obtained over the above specified/split test set is 93.3%. The main implication that can be inferred from the implementation is that the proposed model performs well by learning appropriate features and also retaining the long-term dependencies.

The implementation can be obtained at: https://github.com/PranavPKS/rnn-mbti

5 Limitations and Future Work

The number of words per user was limited to 1000 (as mentioned before), due to memory constraints. A better option would be to go with the mean-0.5*SD, which is around 5000.

Meaning space wasn't explored with the pre-trained word-embeddings.

Other advanced neural architectures like Bi-LSTMs, GRUs can also be implemented which may or may not improve the accuracy.

References

[1] Isabel Briggs Myers. The myers-briggs type indicator: Manual (1962). 1962.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.