

Classify spam messages with two machine learning models: RNN and ELMo

Chaeun Hong (hong.930)

The main purpose of the project is to compare two machine learning models with respect to performance in classifying spam messages. Based on the result of the project, you can refer to decide which model you use for classification related to Natural language processing (NLP).

The basic RNN named vanilla RNN is used for the project, and it uses the forwarding language model for embedding. To be specific, the biggest feature in RNN is that the next hidden layer gets the prior hidden layer's activation function value as input. As a result, the hidden state value gradually reflects the contextual information. ELMo uses both the forward language model and the backward language model. Different from the previous models including RNN, ELMo uses biLM (Bidirectional Language Model) which is pre-trained on a large text corpus in the internal states. During this embedding process, ELMo brings each layer's outputs from the forwarding language model and the backward language model, then concatenates them. Both models do not share their hidden state before concatenation and biLM combines both with the formulation which jointly maximizes log-likelihood for each direction model (Equation 1.).

$$\sum_{k=1}^N (\log p(t_k | t_1, \dots, t_{k-1}; \theta_x, \vec{\theta}_{LSTM}, \theta_s) + \log p(t_k | t_{k+1}, \dots, t_N; \theta_x, \vec{\theta}_{LSTM}, \theta_s)) \quad (\text{Equation 1.})$$

Weights are given to each concatenated one, in this time, weights are calculated based on the purpose of embedding (task-specific). Then, sum all outputs from each layer and multiply the parameter that decides the size of the vector. As a result, ELMo representation comes out, so ELMo can distinguish the homonym by the context well.

In this project, the SMS spam collection dataset from Kaggle is used. About 87 percent are classified as ham (not spam) and the others are spam after removing duplicated data. Therefore, the project is set 80 percent of the total data for training and the others for testing. In addition, the spam messages on average consisted of 150 words, and the ham messages on average consisted of 50 words. The max length of the message is 189 words, and this number is used for padding data in vanilla RNN.

To run the models, Google Colab is used. The project uses Tensorflow and Keras because they provide useful functions for machine learning. Also, TensorFlow Hub has a model for ELMo named elmo, so it is used to make simpler code for ELMo. For vanilla RNN, each word in the data is integer indexing by its frequency using Tokenizer before embedding. RNN is implemented through the basic sequential model, so the model is made with simply piled layers. ELMo does not have to tokenize data because elmo can take untokenized sentences as input. However, because elmo is from TensorFlow Hub and the model is set from Keras, the function is necessary that converts data to be available in elmo and returns a vector that is fitted to Keras format. After embedding through ELMo, the data is processed through the hidden layers with relu activation function. RNN uses the same activation function for each hidden layer's output. Relu activation function can protect from vanishing gradient problem which can occur from sigmoid function because its derivative is 1 for the values greater than 0. Also, it is simple, and the speed of learning is faster than sigmoid. Both models are using sigmoid activation function to classify. Sigmoid

activation function returns 1 if the value is greater than 0.5, so it is useful for binary classification. They use adam as an optimizer. Optimizer is necessary for minimizing the value of the loss function, and it contributes to making the model which shows better performance at the end. The adam is based on SGD and momentum, and it is one of the universal basic optimizers. Both models are trained with 10 epochs and 64 batches. In other words, the models update their weights after 64 samples, and it repeats 10 times.

For comparing their performance, the project uses loss. Both models' loss is calculated with the `binary_crossentropy` function which is useful calculate for binary classification. The function is for minimizing the loss during the training with the optimizer, but it can be used for comparing each model's performance. The equation of the function is the same as below (Equation 2.). If the predicted value is the same as the real value, it returns 0. On the other side, if the predicted value is 0 and the real value is 1, it returns ∞ . The value from binary classification (the predicted value) is the probability value between 0 to 1. However, it is rare that the predicted value becomes 0, so the loss function returns a large value if the predicted value is not close to the real value.

$$L = -\frac{1}{N} \sum_{i=1}^N t_i \log(y_i) + (1 - t_i) \log(1 - y_i) \quad (\text{Equation 2.})$$

For testing in every epoch, 20 percent of the training data is used, so its loss is considered a loss for the test. It is useful to decide whether the model is overfitting or not. Accuracy is also calculated for both training and testing, and it is the ratio of true positive and true negative. Different from the loss function, the value is round before comparing to the real value, so the default threshold is 0.5, and if the value is over 0.5 the predicted value is changed to 1. Then the test accuracy comes out by computing the mean accuracy rate across all predictions. Therefore, the sum of the loss and accuracy is not 1.

As the two graphs are seen below, both models occur overfitting. For the RNN, it shows after the 4th epoch, and, for the ELMo, it shows after the 2nd epoch. When comparing the average test loss, ELMo (0.0623) presents a lower loss than RNN (0.1709). After training each model, the model is tested with the test dataset that is not used for training at all. The test accuracy for ELMo is 99.15 percent and the test accuracy for RNN is 99.44 percent.

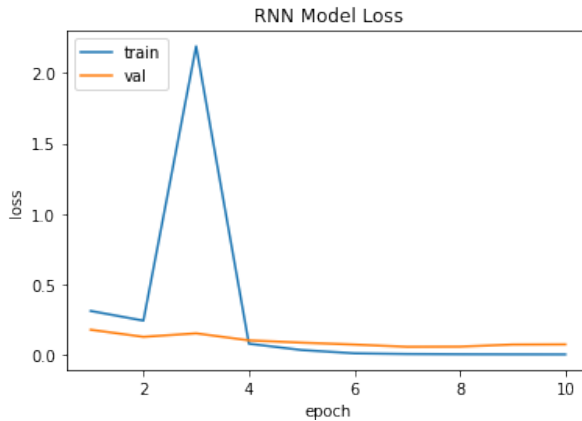


Figure 1. RNN Model Loss Graph

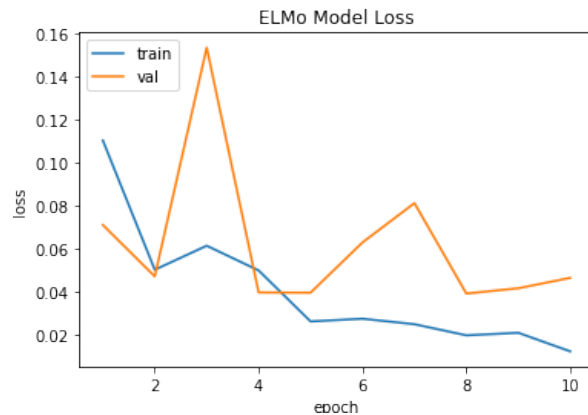


Figure 2. ELMo Model Loss Graph

When it comes to comparing them with respect to time, RNN is much faster to train the model. RNN takes on average 24.1 seconds, and ELMo takes on average 991.9 seconds. This is because that ELMo uses raw string data as input, and it uses biLM as the pre-trained model.

The main problem for both models is overfitting. To solve the problem, it is possible to L2 regularization (Equation 3.), but it seems to cause less accuracy & more loss and does not solve the problem (see *Figure 4.*), so it is better to collect more data classified as spam.

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^N \omega_i^2 \quad (Equation 3.)$$

```
model = Sequential()
model.add(Embedding(voca_size, 256))
model.add(SimpleRNN(256, activation = 'relu', kernel_regularizer='l2'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
```

Figure 3. Modified RNN Code - with L2 Regularization

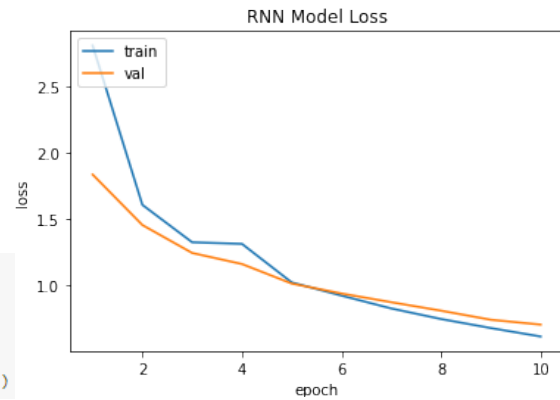






Figure 4. Modified RNN Model Loss Graph

Consequently, ELMo shows better performance with respect to its loss on average, but the two models show similar performance with respect to their accuracy and best number of epochs which does not occur the overfitting problem. Also, the training time is one of the most important elements to decide which models to use, and RNN spends time about 40 times less than ELMo. The biggest advantage of ELMo is deciding the exact meaning of homonyms based on the context, but this is not useful for the classification of spam messages because spam messages and ham messages have more other different words than homonyms. Therefore, it is better to use RNN to classify spam messages.

References

 natural language processing (Nlp)  for beginners. (n.d.). Retrieved April 26, 2022, from <https://kaggle.com/faressayah/natural-language-processing-nlp-for-beginners>

Nlp  glove, bert, tf-idf, lstm...  explained. (n.d.). Retrieved April 26, 2022, from <https://kaggle.com/andreshg/nlp-glove-bert-tf-idf-lstm-explained>

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *ArXiv:1802.05365 [Cs]*. <http://arxiv.org/abs/1802.05365>