Quora Insincere Questions Classification

Jiajun Bao
jb6771
baojiajun2013@gmail.com

Yayun Fan
yf1290
yf1290@nyu.edu

Abstract

Millions of questions are posted on Quora every year. While most of the questions are being very helpful and inspiring, a few of the questions posted contains some negative information and thus should be blocked before they go to public. Detecting the insincere questions posted on Quora is a classical text classification problem. This paper implements and evaluates the performances of some frequently used text classification models: Multinomial Naive Bayes (MNB), Support Vector Machine (SVM), SVM with NB feature (NB-SVM) model and Convolutional neural network (CNN). We also demonstrate the process of creating our own models, CNN-LSTM model and CNN-GRU model.

Key words: Text classification, MNB, SVM, NB-SVM, CNN and Quora insincere questions.

1.Introduction

The internet has dramatically changed the way that people share and learn knowledge. It is now quite common for people to post their questions or search for answers on the web for many purposes. Quora is a well-known platform where people can do these. However, due to the fact that there is no qualification control, anyone can write anything on the web, which results in many low quality questions, sometimes even pornography or insults questions. In order to filter these insincere questions and maintain a better internet environment, we focus on detecting insincere questions in this paper.

Quora insincere questions detection is a typical text classification problem in natural language processing field. Many great works have been done on text feature extraction and modeling. Le and Mikolov represent each document as a dense vector to overcome the weakness of bag-of-words. [1] Kim came up with an excellent CNN model with hyperparameter tuning and static vectors. [2] Recurrent neural networks (RNN) and long short-term memory (LSTM) have also been successfully applied in text classification. [3] For most recent studies, Yin and Schutze introduce attention mechanism. [4]

In this paper, firstly, we train some traditional machine learning models to get a baseline for future improving models. We use the term frequency-inverse document frequency (TF-IDF) to extract features from the text, then input the text features into some models including Multinomial Naive Bayes (MNB), Support Vector Machine (SVM), and SVM with NB feature (NB-SVM) model. Moreover, introducing word2vec to extract features can often get more precise features and improve model accuracy.

Moving into the deep learning, CNN has been proven as one of most popular models for text classification. As a model originally created for picture processing using layers and convolving filters, it was extended to the text field with great performance most of time. Here we come up with a modified CNN model.

2. Data and preprocessing

2.1. Dataset

We use the question dataset provided by Quora. There are two datasets, training set and test set, both including three attributes: qid (Unique ID number), question_text, and target (label: 0/1). Each question is labeled "1" when it is insincere and "0" when it is sincere. The learning task is to classify the new question into insincere or sincere.

2.2. Preprocessing

We separately use two feature extraction methods, TF-IDF and word2vec. Figure 1 shows the steps of our data preprocessing.

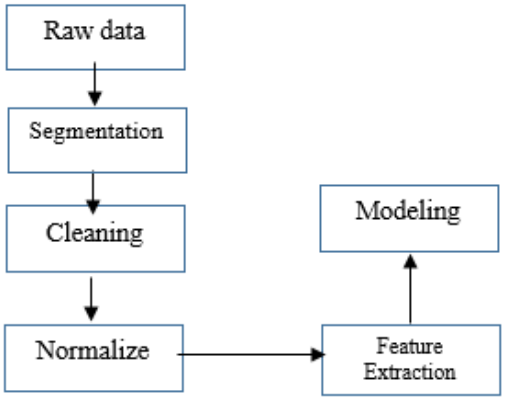

Figure 1: Data Preprocessing

In TF-IDF, we firstly split question text into words. Secondly, we do some data cleaning, like processing special symbols, correcting misspelling, removing stop-words. Thirdly, we do some stemming and lemmatization. Then we use TF-IDF weighting scheme to extract features.

In word2vec, since we use pre-trained embeddings, in order to avoid losing valuable information, which would help our model to figure things out, we keep the stop-words and do not adopt stemming. Figure 2 shows the format of embedding matrix. We also directly tokenize the words in question text and convert them into unique matching numbers. Basically we build a dictionary of words and their unique numbers, and we replace the words with the numbers that matches. Now we have a giant matrix of numbers that each row represents a question posted on Quora. Notice that here each row has different length since the questions have different number of words. In order to transform the data into structured data that can be passed into our models, we limit the length of the question and fill in zeros for the missing values.

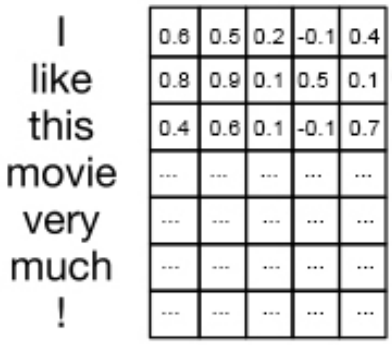

Figure 2: Embedding Matrix [5]

3.Models

We firstly build four traditional models to classify the questions, which do not own high classification F1-score. And then we create our model based on CNN architecture and LSTM, which get a better performance.

3.1 Multinomial Naive Bayes (MNB)

In naive bayes field, since features are discrete variables, we choose Multinomial naive Bayes to create our model.

In MNB, feature vector $x = (x_1, \dots, x_n)$

conditional probability: $\theta_{y_i} = p_{(x_i|y)}$

$\theta_y = (\theta_{y^1}, \dots, \theta_{y^n})$

$$\hat{\theta}_{y_i} = \frac{N_{y^i} + \alpha}{N_y + \alpha n}$$

Where $N_{y^i} = \Sigma_{x \in T}$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y = \sum_{i=1}^{n} N_{yi}$ is the total count of all features for class y. α is the smoothing parameter.[5]

3.2 Support Vector Machine (SVM)

Comparing with L1 regularization, we adopt L2 regularization term to make the SVM less susceptible to outliers and improve its overall generalization.

$$\text{Minimize } Q(w, \varepsilon) = \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{m} \varepsilon_i^2$$

$$\text{Subject to } y_i(x_i \cdot w + b) \geq 1 - \varepsilon_i \quad i = 1, \ldots, m$$

$$\mathbf{w}^T\mathbf{w} + C\Sigma_i \max\left(0, 1 - y_i(\mathbf{w}^T\hat{\mathbf{x}}_i + b)\right)^2$$

We adopt RBF kernel function and use grid search to find optimal penalty parameter and kernel coefficient. [6]

3.3 SVM with NB feature(NB-SVM)

We formulate our main model variants as linear classifiers, where the prediction for test case k is

$$y_i = \text{sign}(\mathbf{w}^T\hat{\mathbf{x}}_i + b)$$

$$r = \log\left(\frac{p/||p||}{q/||q||}\right)$$

$$p = \alpha + \sum_{i:y_i=1}^{m} y_i \quad q = \alpha + \sum_{i:y_i=-1}^{m} y_i$$

Otherwise identical to the SVM, except we use $\hat{\mathbf{x}}_i = \hat{\mathbf{r}} \circ \hat{\mathbf{x}}_i$ which is the element-wise product. Furthermore, there is an interpolation between MNB and SVM performs excellently for all documents:

$$\mathbf{w}' = (1 - \beta)\bar{w} + \beta\mathbf{w}$$

where $\bar{w} = \|\mathbf{w}\|_1/|V|$ is the mean magnitude of w, and $\beta \in [0,1]$ is the interpolation parameter.[7]

3.4 Convolutional neural network (CNN)

In this section, we will move into deep learning model. After embedding the words, we train a simple one layer CNN model. The model's F1 score is 60.25%, which is even lower than previous basic models, so we will try to make improvements in next section. Figure 3 presents an abstract view of a simple CNN model.

3.5 CNN-LSTM

In this section, we will describe the process of creating our own CNN-LSTM model and how to choose optimal hyperparameters. Since CNN is good at extracting local information but could not capture long-distance dependencies, we add a LSTM layer to process the former CNN layer's output. [8]

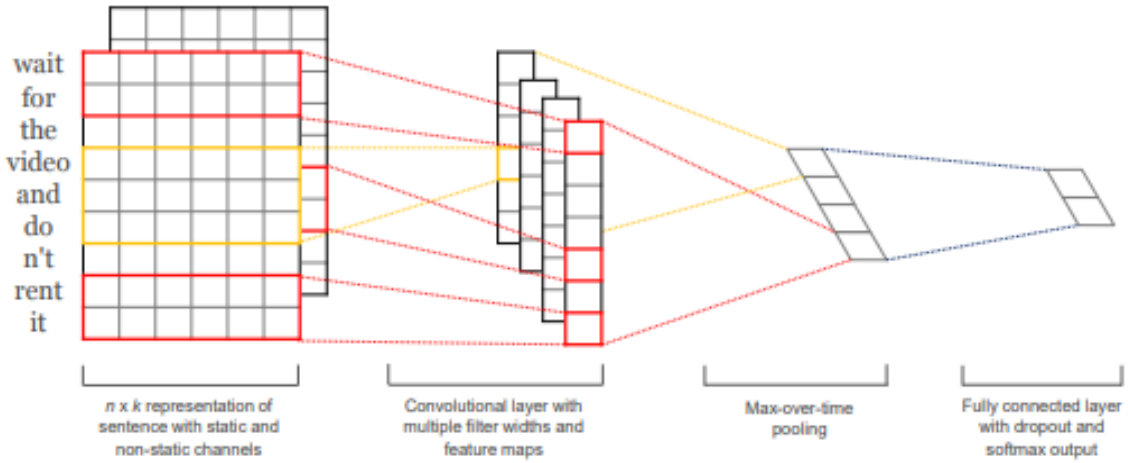

Figure 3: CNN Model [2]

The following Figure 4 shows an abstract image of different layers mentioned above.

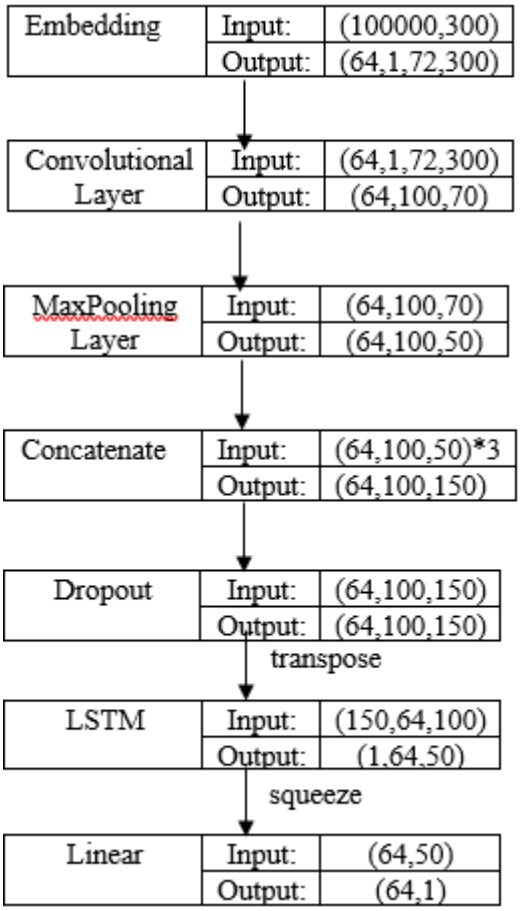

Figure 4 Architecture of CNN-LSTM

First, we have a convolution layer. We apply filters with different sizes and values on the input matrix. Different from image processing, filters usually have fixed length that is equal to the length of the input matrix. We also apply the relu functions to the result to remove negative values.

Next we have a max pooling layer. We choose max pooling to extract the most outstanding features and reduce dimensions. Then we concatenate the pooled results together. A dropout layer is also implemented to avoid overfitting.

Finally, we build a fully connected layer using LSTM layer and Linear layer that connects each features with two final classes: 0 or 1.

During hyperparameter optimization, we tested various CNN configurations. Some parameters we tested:

- Batch size: {32, 64, 128}
- Activation function: {softmax, tanh, sigmoid, relu}
- Dropout rate: {0.1, 0.3, 0.5}
- Max number of features: {50000, 75000, 100000}
- Filter size: {[2,3,4], [3,3,3], [3,4,5]}
- Learning rate: {0.001, 0.005, 0.01}

In the final model, we choose the epoch number of 5, batch size of 64, learning rate of 0.001 and 3-fold cross validation. We use 3 covolutional filters with a filter size of 3, 4, 5 followed by a max pooling layer. We adopt a dropout rate of 0.1. We choose BCEWithLogitsLoss loss function, which combines a sigmoid layer and the binary cross entropy loss. [9]
binary cross entropy:

$$l_n = -w_n[t_n \cdot \log \sigma(x_n) + (1 - t_n) \cdot \log(1 - \sigma(x_n))]$$

Sigmoid function:

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

Our CNN-LSTM model has the higher F1-score compared with the above models. This could be explained by the fact that our deep learning approach has the ability to capture more complex features than traditional learning approaches.

3.6 Model Analysis

In the above models, we always use cross validation method to decrease model overfitting. When converting predicted probabilities into class prediction, we all adopt dynamic threshold search to find best threshold. When evaluating models, we also

consider multiple indicators, like F1-score, Average precision score, Log loss. Detailed results are shown in table 1.

	F1-score
MNB	56.72%
SVM	62.69%
NB-SVM	63.23%
CNN	60.25%
CNN-LSTM	64.21%
CNN-GRU	63.94%

Table 1 Model Results

From the results, it is apparent that combining different models is a fancy approach to create better models. Introducing NB classification weight into SVM improves the model performance. Including a LSTM layer in CNN model also increase performance. LSTM and GRU layer give us similar result. [10]

6.Conclusion and Future Work

These results come with several surprises and lessons learned.

First, data preprocessing and feature extraction are more important in terms of increasing model performance than model selection or hyperparameter selection in this situation. Second, combining basic machine learning models can get better outcome, like NB-SVM, CNN-LSTM. Third, in deep learning model, adding more layers do not guarantee the improvement in performance, but usually increase more overfitting and time consuming. [11]

In future work, we will try more approaches to modify our model. For data preprocessing, adding more text statistical features can also enhance model performance. Concatenating multiple word embedding is also a fancy approach, which is similar with adding more text features. For model architecture, we wish to try Bidirectional LSTM model with max pooling or introducing attention mechanism. Using weighted sum of different model prediction results is also a promising potential method.

References

[1] Quoc Le, Tomas Mikolov. 2014. Distributed representations of sentences and documents

[2] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification.

[3] Pengfei Liu, Xipeng Qiu, Xuanjing Huang. 2016. Recurrent Neural Network for Text Classification with Multi-Task Learning

[4] Wenpeng Yin, Hinrich Schütze, Bing Xiang, Bowen Zhou. 2015. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs

[5] https://en.wikipedia.org/wiki/Naive_Bayes_classifier#Multinomial_naive_Bayes

[6] https://pytorch.org/docs/stable/nn.html?highlight=bcewithlogitsloss#torch.nn.BCEWithLogitsLoss

[7] Sida Wang and Christopher D. Manning. 2012. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification

[8] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, Francis C.M. Lau. 2015. A C-LSTM Neural Network for Text Classification

[9] https://pytorch.org/docs/stable/nn.html?highlight=bcewithlogitsloss#torch.nn.BCEWithLogitsLoss

[10] Jason Poulos. Predicting Stock Market Movement with Deep RNNs

[11] Mark Hughesa, Irene LIa,b,1, Spyros Kotoulasa and Toyotaro Suzumurab. 2017. Medical Text Classification using Convolutional Neural Networks