School of Computer Science

THE UNIVERSITY of ADELAIDE

Assessment Cover Sheet

Student Name	Zhang zhe
Student ID	a1775543
Assessment Title	
Course/Program	Big data analysis & projec
Lecturer/Tutor	
Date Submitted	27 Oct 2019
OFFICE USE ONLY Date Received	

KEEP A COPY
Please be sure to make a copy of your work. If you have submitted assessment work electronically make sure you have a backup copy.

PLAGIARISM AND COLLUSION
Plagiarism: using another person's ideas, designs, words or works without appropriate acknowledgement.
Collusion: another person assisting in the production of an assessment submission without the express requirement, or consent or knowledge of the assessor.

CONSEQUENCES OF PLAGIARISM AND COLLUSION
The penalties associated with plagiarism and collusion are designed to impose sanctions on offenders that reflect the seriousness of the University's commitment to academic integrity. Penalties may include: the requirement to revise and resubmit assessment work, receiving a result of zero for the assessment work, failing the course, expulsion and/or receiving a financial penalty.

I declare that all material in this assessment is my own work except where there is clear acknowledgement and reference to the work of others. I have read the University Policy Statement on Academic Honesty & Assessment Obligations (http://www.adelaide.edu.au/policies/230).
I give permission for my assessment work to be reproduced and submitted to other academic staff for the purposes of assessment and to be copied, submitted to and retained by the University's plagiarism detection software provider for the purposes of electronic checking of plagiarism.

Signed…… Date ……

Zhe Zhang A1775543

Big Data Analysis & Project
Oct 27 2019

Quora Insincere Questions Classification:

Detect toxic content to improve online conversations

Abstract

Natural Language Processing (NLP) is a research field of designing methods and algorithms for input and output to unstructured natural language data. This paper proposes a neural network for detect toxic content to improve online conversations based on questions on Quora. I choose Python as the programming language. The main tools I used in this project are Pytorch and Keras. I develop and train the LSTM + GRU model and discuss methods to enhance the performance. The final results I got on Kaggle is 0.69178.

Keywords：Deep learning, Neural network, NLP, Data science.

Introduction

This project aims to detect toxic content to improve online conversations based on questions on Quora. Participants seek to develop models that take as input the text of a question in English, and outputs a 0 or 1 corresponding to whether the question should be approved as “sincere” or flagged as “insincere.”

This is a highly relevant problem today, with online forums that act as platforms for people to resolve their curiosities—such as Reddit, Yahoo Answers, and StackExchange—struggling to handle posted questions that violate the guidelines of the forum. Though there often exist humans that act as moderators, usually individuals passionate about the discussion topic, the number of questions to monitor far outweighs the capacity of these moderators. The ability to automatically flag disingenuous, malicious, discriminatory questions—broadly, “insincere” questions—will help remove such negative content from these communities quickly and deter potential posters.

The Kaggle provided us with the data sets needed for this competition, including the training data set (train.csv) for the training model and the test data set (test.csv) for testing the performance of the model.

Insincere questions, according to Quora, are defined as ones that meet any of the following (non-comprehensive) criteria:

• *Has a non-neutral tone*, such as an exaggeration to emphasize a comment about a particular group of people

• *Is disparaging or inflammatory*, such as an attempt to seek confirmation of a stereotype or present a discriminatory remark, a comment based on an outlandish premise about a group of people, or the disparaging of a natural, not fixable, or immeasurable characteristic

• *Is not grounded in reality*, such as a reference to false information or absurd assumptions

• *Uses sexual content for shock value*, such as references to pedophilia, incest, or bestiality outside of an attempt to find genuine answers

The organizer also offers four embeddings for us to choose from:

GoogleNews-vectors-negative300
glove.840B.300d
paragram_300_sl999
wiki-news-300d-1M

The structure of the data is not complex, with a total of three columns of training data: qid, question_text and target (0-not insincere, 1-insincere). The training data was about 1.31 million.

In this project, participants are required to execute their code onto Kaggle's Kernel and submit the result generated from online computing. Thus the code running time is limited (GPU within 2 hours, CPU within 6 hours), and external data is prohibited.

Therefore, the most important thing is to ensure that the Kernel can complete the training and testing of the model in a specified time to achieve effective results. The final prediction is whether each question ID is insincere, with F1-score as the evaluation index.

The main tools I used in this project are Pytorch and Keras.

PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing. It is primarily developed by Facebook's artificial intelligence research group. It is free and open-source software released under the Modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ frontend. Furthermore, Uber's Pyro probabilistic programming language software uses PyTorch as a backend. PyTorch provides two high-level features:

> Tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU)
>
> Deep neural networks built on a tape-based autodiff system

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

Methodology

Data Collection
The quantity & quality of the data dictate how accurate the model is. The outcome of this step is generally a representation of data which we will use for training. Using pre-collected data, by way of datasets from Kaggle, UCI, etc., still fits into this step.

Data Preparation
The preprocessing here mainly refers to data cleaning, such as special symbol processing, spelling error correction and so on.

In text preprocessing, some competitors replace words that are not in the word vector matrix, such as: {'Babchenko':' Arkady Arkadyevich Babchenko faked death',}.

In fact, these may not be equivalent, so blunt word replacement work is not necessarily for the model. It is a wise decision to skip this step directly.

Other methods such as replacing LaTeX characters, numeric substitution, etc., have reduced the score, and therefore I did not use it.

Embedding

Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers. Conceptually it involves a mathematical embedding from a space with many dimensions per word to a continuous vector space with a much lower dimension.

The most popular way is to using the mean of 4 embeddings. However, I find that using glove*0.6 + para*0.4 behaves better than using the average one.

Attention

Attention allows the decoder to focus on different parts of the input sequence at each output.

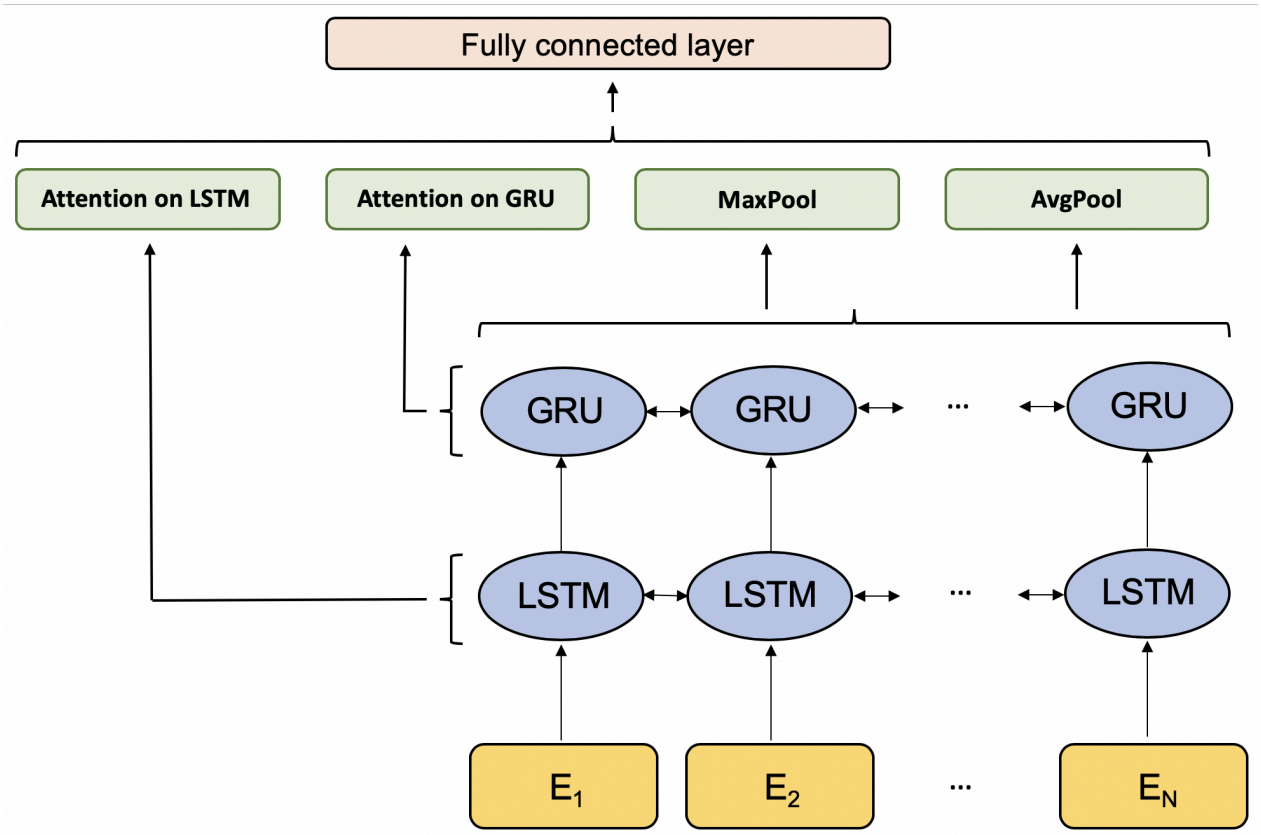
Choose a Model

Using a single model or multiple models is very controversial. I chose to use a single model, because a single model can better adjust parameters, and it is easier to optimize the model in the later stage.

I implement a LSTM + GRU model has been shown to perform well for this task. The architecture of the LSTM + GRU model is as follows:

1. Bidirectional LSTM with attention on input sequence
2. Bidirectional GRU with attention on LSTM output sequence
3. Separate average-pool and max-pool layers on the GRU output sequence
4. Linear layer (320, 16)
5. ReLU layer over linear layer outputs
6. Dropout layer with p = 0.1

The diagram below illustrates this architecture:



The architecture of the LSTM + GRU baseline model. The attention and pooling outputs are concatenated as an input to the fully connected layer.

This is a typical Recurrent Neural Network (RNN). RNN can represent sequences of arbitrary length as fixed-length vectors while focusing on the structured properties of the input.

The LSTM structure is designed to solve the gradient disappearance problem and is the structure of the first introduction mechanism.

The complexity of the structure makes LSTM difficult to analyze, and the computational cost is also high. GRU is an alternative to LSTM. Similar LSTM, GRU is also based on the gate mechanism, but generally uses fewer doors and does not have separate memory components.

Train the Model

The goal of training is to answer a question or make a prediction correctly as often as possible.

Evaluate the Model

Cross-validation, it's a model validation techniques for assessing how the results of a statistical analysis (model) will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

The goal of cross-validation is to define a data set to test the model in the training phase (i.e. validation data set) in order to limit problems like overfitting, underfitting and get an insight on how the model will generalize to an independent data set. It is important the validation and the training set to be drawn from the same distribution otherwise it would make things worse.

The final private score I got is 0.69955 and the public score is 0.69178.

Experiment

Data and preprocessing

The dataset that we use is provided by Quora through their "Insincere Questions Classification" challenge on Kaggle.

Insincere questions have a low prevalence of 6.187% across the train sets. Additionally, Quora has suggested that there is an unidentified amount of noise in the labels, which places an upper bound on the theoretical performance capacity of the model.

I set the embed size as 300, max features as 95000, max length as 70.

The embed size represents how big is each word vector, the max features represents how many unique words to use (i.e num rows in embedding vector) and the max length represents the maximum number of words in a question to use.

Since many participants spent a lot of time on preprocessing but finally hurt the score, I decide simply add a space on each side of the special characters to separate them from the original sentence.

```
PUNCTS = [',', '.', '"', ':', ')', '(', '-', '!', '?', '|', ';',
"'", '$', '&', '/', '[', ']', '>', '%', '=', '#', '*', '+', '\\',
'•',  '~', '@', '£', '·', '_', '{', '}', '©', '^', '®', '`',
'<', '→', '°', '€', '™', '›',  '♥', '←', '×', '§', '″', '′', 'Â',
'█', '½', 'À', '…', '“', '★', '”', '–', '●', 'Â', '►', '−', '¢',
'²', '¬', '░', '¶', '↑', '±', '¿', '▾', '═', '¦', '‖', '—', '¥',
'▓', '—', '‹', '─', '▒', ': ', '¼', '⊕', '▼', '▪', '†', '■', '’',
'■', '¨', '▄', '♫', '☆', 'É', '¯', '♦', '¤', '▲', 'È', '¸', '¾',
'Ã', '⋅', '‘', '∞', '∙', ') ', '↓', '、', '│', ' (', '»', ', ',
'♪', '╩', '╚', '³', '・', '╦', '╣', '╔', '╗', '━', '❤', 'Ï', 'Ø',
'¹', '≤', '‡', '√', ]

DEF CLEAN_TEXT(X):
    X = STR(X)
    FOR PUNCT IN PUNCTS:
        X = X.REPLACE(PUNCT, F' {PUNCT} ')
    RETURN X
```

I’m not sure why a good preprocessing hurts the score. I saw a discussion on Kaggle but have few comment. Some participants insist that this is due to the high cover rate of word embeddings provided by the Kaggle.

Embedding

The Kaggle challenge additionally provides two sets of word embeddings that we use to train the LSTM + GRU model:

1. *glove.840B.300d*, i.e the ones created during the GloVe research project at Stanford. There are four embedding sets: one based on Wikipedia, two based on Common Crawl, and one based on Twitter.

2. *paragram_300_sl999*, a.k.a. Paragram Embeddings, developed by the Cognitive Computa- tion Group at the University of Pennsylvania.

```
EMBEDDING_MATRIX = GLOVE_EMBEDDINGS*0.6 + PARAGRAM_EMBEDDINGS*0.4
EMBEDDING_MATRIX[0] = NP.ZEROS([300,])
```

Setting embedding_matrix [0] to full 0 vector is effective for getting a good score.
Use dropout after embedding will also increase the score.

```
CLASS SPATIALDROPOUT(NN.DROPOUT2D):
    DEF FORWARD(SELF, X):
        X = X.UNSQUEEZE(2)
        X = X.PERMUTE(0, 3, 2, 1)
        X = SUPER(SPATIALDROPOUT, SELF).FORWARD(X)
        X = X.PERMUTE(0, 3, 2, 1)
        X = X.SQUEEZE(2)
        RETURN X
```

Training

I set the hidden layer size as 40. Higher hidden layer size will enhance the performance of prediction, but it will consume more time to execute.

I set the batch size as 512 and the number of epochs as 6. The batch size represents how many samples to process at once and the number of epochs represents how many times to iterate over all samples.

Evaluation method

The evaluation metric for our models is the F1 score (the harmonic average of precision and recall), which penalizes the model's approval of insincere questions and flagging of sincere ones.

Findings

At first, I used the race more commonly used keras, but found that keras has a fatal problem, the result can not repeat, and the keras speed is slow. At this point I decide to change the framework, pytorch is a very good choice.

The problem lies within CuDNN. CuDNN's implementation of GRU and LSTM is much faster than the regular implementation but they do not run deterministically in TensorFlow and Keras. In this competition were speed is essential you can not afford to keep determinism by using the regular implementation of GRU and LSTM.

Keras solves some of these problems with TensorFlow but it has a high-level API. I think that when doing research, it is often preferable to be able to interact with the model on a low-level. And you will see that the lower level API still doesn't make it complicated to work with PyTorch.

Discussion & Conclusion

In this project, I have learned the basic concepts and applications of deep learning, the most commonly used machine learning tools, the advantages and disadvantages of these tools.

I skimmed through a lot of contestants' schemes and summed up several ways to improve scores efficiently:

1. RNN based model, including LSTM and GRU, using bi-directional structure.
2. Dropout after embedding
3. When data is considerably large, forced destruction of distribution behavior may be futile, such as data cleaning. Basically 100 times effort, few of profit.

Following methods are highly depend on parameter adjustments, they don't necessarily improve overall scores, sometimes even have negative affects.

1. BN and dropout, and their relative positions.
2. Data preprocessing: drop, shuffle, replace…
3. Cyclical Learning Rate.
4. Changing seed.

Reference

Goldberg, Y. (n.d.). Neural network methods for natural language processing.

Raschka, S. (n.d.). Python machine learning.

Kaggle.com. (2019). Deterministic neural networks using PyTorch | Kaggle. [online] Available at: https://www.kaggle.com/bminixhofer/deterministic-neural-networks-using-pytorch [Accessed 27 Oct. 2019].

Kaggle.com. (2019). Different embeddings with Attention! [Fork] | Kaggle. [online] Available at: https://www.kaggle.com/danofer/different-embeddings-with-attention-fork [Accessed 27 Oct. 2019].

En.wikipedia.org. (2019). Keras. [online] Available at: https://en.wikipedia.org/wiki/Keras [Accessed 27 Oct. 2019].

Kaggle.com. (2019). LSTM + Attention Baseline [0.672 LB] | Kaggle. [online] Available at: https://www.kaggle.com/suicaokhoailang/lstm-attention-baseline-0-652-lb [Accessed 27 Oct. 2019].

En.wikipedia.org. (2019). PyTorch. [online] Available at: https://en.wikipedia.org/wiki/PyTorch [Accessed 27 Oct. 2019].

Kaggle.com. (2019). TFIDF-NaiveBayes-LogReg Baseline | Kaggle. [online] Available at: https://www.kaggle.com/ryanzhang/tfidf-naivebayes-logreg-baseline [Accessed 27 Oct. 2019]