Machine Learning Engineer Nanodegree

Capstone Proposal

Jing Li
November 19, 2018

Toxic Comment Classification with CNN

Domain Background

Toxic comment classification is a Kaggle challenge to identify and classify toxic online comments[1]. The abuse and harassment online discourage many people from expressing themselves and seeking different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments. Jigsaw and google have built models to study negative online behaviors like toxic comments[2]. However, the current models don't allow users to select types of toxicity. The challenge is to build a multi-headed model capable of detecting different types of toxicity like threats, obscenity, insults, etc.

I am with AT&T and one of my focus is to analyze documents (ticket, incident, event recording, etc.) to identify information patterns and classify application outage root causes. Obviously, sentence classification knowledge obtained through this challenge will benefit me greatly on automating the work me and many others are doing daily. Convolutional Neural Networks (CNNs) were responsible for major breakthroughs in image classification and are the core of most computer vision systems today. CNNs have also achieved remarkably strong performance on sentence classification[3]. For this project, I intend to use CNN for toxic comment classification.

Problem Statement

The current models built by Jigsaw and Google on study of negative online behaviors like toxic comments (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion) are still make errors and do not allow users to select which types of toxicity they are interested in finding. The solution is to build a multi-headed model that can detect different types of toxicity like threads, obscenity, insults and identity-based hate, and at the same time, reduce errors.

---

[1] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
[2] https://www.engadget.com/2017/09/01/google-perspective-comment-ranking-system
[3] Y. Zhang, B. Wallace. A sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification 2016

Datasets and Inputs

The dataset provided by Kaggle[4]:

The dataset here is from wiki corpus dataset[5] which was rated by human raters for toxicity. The corpus contains 63M comments from discussions relating to user pages and articles dating from 2004-2015.

The comments are labeled in the following six categories

- toxic
- severe_toxic
- obscene
- threat
- insult
- identity_hate

The dataset will be used to create a model which predicts a probability of each type of toxicity for each comment.

File descriptions

- train.csv - the training set, contains comments with their binary labels
- test.csv - the test set, you must predict the toxicity probabilities for these comments. To deter hand labeling, the test set contains some comments which are not included in scoring.
- sample_submission.csv - a sample submission file in the correct format (will not be used for capstone project)
- test_labels.csv - labels for the test data; value of `-1` indicates it was not used for scoring. (Note: file added after competition close!)

Pre-trained word vectors:

- crawl-300d-2M.vec.zip[6]: 2 million-word vectors trained on Common Crawl (600B tokens) click here to download

[4] Kaggle Toxic Comment Classification Challenge
[5] E. Wulczyn, N. Thain, and L. Dixon. Ex Machina: Personal Attacks Seen at Scale 2017
[6] T.Mikolov, E. Grave, P. Bojanowski, C.Puhrsch, A.Joulin. Advances in Pre-Training Distributed Word Representations

The first line of the file contains the number of words in the vocabulary and the size of the vectors. Each line contains a word followed by its vectors, like in the default fastText text format. Each value is space separated. Words are ordered by descending frequency.

The pre-trained word vector will be used in the word embedding for the dataset and feed into CNN.

Solution Statement

As was mentioned briefly in Domain Background section that CNNs have achieved great success in sentence classification and should work very well for this project. A comment can be converted to a sentence matrix. The dimensionality of the sentence matrix can be denoted as s x d matrix where s is the length of the sentence and the d is the dimension of word vectors. The sentence matrix can be treated as a s x d 'image' and convolution on it via linear filters can then be performed. As has been proven that CNN solution for sentence classification is quantifiable, measurable and replicable[7]

Benchmark Model

I plan to use Logistic Regression with TFIDF for word vectoring as benchmark model. I have used Logistic Regression in my dog project and other binary classification problems with success. I have also used TfidfVectorizer in other projects. Logistic Regression is the building block of all that constitutes Deep Learning. It is simple and fast and has been used as a baseline for sentence classification. ROC_AUC is the required metric to compare and score the performance of models for this Kaggle challenge and we can use this metric while training our Logistic Regression model with cross validation.

Evaluation Metrics

As was mentioned in Benchmark Model, ROC_AUC metric is required and used by Kaggle to score submitted models.

ROC (Receiver Operating Characteristic) curve is created by plotting the TPR (True Positive Rate) on the y-axis against the FPR (False Positive Rate) on the x-axis for every possible classification threshold. TPR is also known as sensitivity or recall and FPR as fall-out. The ROC curve depicts relative trade-offs between true positive (benefits) and false positive (costs). AUC (Area Under the Curve) is the area under the normalized ROC curve. It summarizes the ROC curve into a single number. ROC AUC statistics works well with imbalanced classes and is often used in machine learning to compare model performance. Refer to this Wikipedia page[8] for details of the ROC AUC including AUC integral calculation, its pros and cons.

---

[7] Y. Zhang, B. Wallace. A sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification 2016

[8] Wikipedia: Receiver Operating Characteristic

Project Design

The roadmap is as follows:

1. Exploratory Data Analysis
   In this step, the dataset provided by Kaggle will be analyzed. The analysis will include the type of the data, basic statistics, abnormalities, etc. Where possible, visualization of the features about the data will be provided.
2. Data Preprocessing
   In this step, any data preprocessing required will be analyzed. This include address data abnormalities, etc.
3. Benchmark
   In this step, proposed benchmark model Logistic Regression will be implemented, and the result will be analyzed. The leaderboard scores in Kaggle challenge may be used for secondary performance comparison.
4. Implement Solution
   In this step, the proposed CNN model will be implemented. Model architectures with different Layers will be tested.
5. Compare Model Performance
   The result from CNN model test score will be compared with baseline test score from benchmark. It will also be compared with Kaggle leaderboard. The final model will be selected.
6. Optimize Model
   The final CNN model will be fine-tuned for optimal performance. Parameters such as number of filters, region size, activation function, pool size, dropout rate, batch size and epoch number for fitting will be tuned.
7. Conclusions and Improvement Opportunities
   The solution and the results will be summarized here. Lessons learn and opportunities for solution improvement will be discussed

The steps laid out above are iterative in that while we will go through them one at a time, we might go back to an earlier step and revisit some of our decisions. We will be constantly evaluating past decisions and making improvements.
