Multi-class Sentiment Analysis using Deep Learning

Karan Atulkumar Shah
dept. of Computer Science
1100690
Lakehead University
Thunder Bay, Canada
kshah8@lakeheadu.ca

***Abstract*—In this study, I explore various natural language processing (NLP) methods to perform sentiment analysis. I look at one datasets, with multi-class labels. For the multi-class case i applied word2vec models followed by Convolutional Neural Network (CNN).**

I. Introduction

Sentiment analysis is a well-known task in the realm of natural language processing. Given a set of texts, the objective is to determine the polarity of that text. [1] provides a comprehensive survey of various methods, benchmarks, and resources of sentiment analysis and opinion mining. The sentiments can consist of different classes. In this assignment, I worked on multi class analysis. A Movie review is 0-Negative, 1-somewhat Negative, 2-neutral, 3-somewhat positive, 4positive.

For this type of sentiment analysis i choose "Rotten Tomatoes Movie Review" data set and perform various preprocessing steps, for feature extraction i use Word2Vec and to train this data i use Convolutional Neural network(CNN). As in many natural language tasks, the first task here is to clean up, and convert the input texts (movie reviews) into numbers. This can be done using a variety of methods such as bag of words, word to vector, etc. Afterwards, We train the classifier.

II. Dataset

In this assignment i worked on "Rotten Tomatoes Movie Review" data-set. This is a multi-class classification problem, which simply means the data set have more than 2 classes(binary classifier). The five classes corresponding to sentiments: 0-Negative, 1-somewhat Negative, 2-neutral, 3-somewhat positive, 4positive.The data set is quite challenging as the data-set is split into separate phrases and each one of them have a sentiment label. Moreover, obstacles like sentence negation, sarcasm, terseness, language ambiguity, and many others make this task very interesting for Natural Language Processing

Let us have a look at first few phrases of the training dataset in fig 1: Now let's see the number of comments across the categories in fig 2. As we can see that the dataset is imbalanced, the dataset have to resampled so, that our model is not biased.

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
1	2	1	A series of escapades demonstrating the adage ...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2

Fig. 1. Rotten Tomatoes Movie Review.

```
data['Sentiment'].value_counts()
```

```
2    79582
3    32927
1    27273
4     9206
0     7072
Name: Sentiment, dtype: int64
```

Fig. 2. number of comments across the categories.

III. Technical Approach and Models

A. Data Preprocessing

1) TEXT PREPROCESSING: In Natural language processing(NLP), the model doesn't need the irrelevant parts of the corpora; it just needs the important words and phrases required to execute the task at hand. Thus, text preprocessing techniques involve prepping the corpora for proper analysis and for the machine learning and deep learning models. Text preprocessing is basically telling the machine what it needs to take into consideration and what it can disregard.

I will be using the NLTK Python library in the following tasks. NLTK stands for Natural Language Toolkit and is the simplest and one of the most popular Python libraries for natural language processing, which is why i will be using it to understand the basic concepts of natural language processing.

2) TEXT PREPROCESSING TECHNIQUES: I have tried several basic sentiment analysis methods as described below. For all methods, I performed a preprocessing step to clean up the data. This includes removing the numeric data , unnecessary punctuation , convert to lower case, and removing

stop words2. You can find the clean-review() method inside my implementation for data pre-processing.

To convert a cleaned sequence of words to numerical feature vectors i tried the following method.

B. Word2Vec

Another way to numerically represent texts is to transform each word of the text to a vector [2]. This transformation should preserve the semantics of words, that is if the meanings of two words are close, their vectors should be close as well (in an L2-distance sense). One important aspect of the word2vec task is that it is independent of the main objective (here sentiment analysis), and does not require a labeled dataset. Therefore, here we used all of the 75,000 reviews (25,000 labeled and 50,000 unlabeled training sets) as the corpora to train word vectors. Aside from the regular preprocessing steps on the raw reviews, to train word vectors we need to split paragraphs into sentences, since word2vec algorithm accepts sentences as input.

1) Word2Vec Approaches: We can perform Word2Vec in two different approaches.

- Using Pre-Trained Model: For this approach we can use googles word embedding pre-trained model for word embedding. We can use this model using gensim Library.
- Creating own model using dataset: I have used this approach in my assignment for word embedding.In this approach, we can first separately learn word embeddings and then pass to the embedding layer. This approach also allows to use any pre-trained word embedding and also saves the time in training the classification model.
  I have used the Gensim implementation of Word2Vec. The first step is to prepare the text corpus for learning the embedding by creating word tokens, removing punctuation, removing stop words etc. The word2vec algorithm processes documents sentence by sentence. After creating this word2Vec model i have saved it and use it in my Convolutional neural network(CNN) to build my model.You can find this saved model in my git-hub repository.

C. 1D convolutional Neural Network

To Build my model i have used convolutional neural network and using this model i have train and test my dataset. To understand the architecture of a CNN, it is best to separate it into two sections as

- Feature extraction: The first section of a CNN is all about feature extraction. Conceptually, it can be interpreted as the model's attempt to learn which features distinguish one class from another.Feature learning occurs through a set of three operations repeated a number of times, as follows:
  - Convolution: Convolution can be thought of as looking through a small window as we move the window to the right and down. Convolution, in this context, involves iteratively sliding a "filter" across an data, while applying a dot product as we move left and down.This window is called a "filter" or a "kernel". To build my model i have implement two convolutional layer with two different filter size and take the kernel size unique for both layers.
  - An activation function: Activation functions are used all across machine learning. They are useful for introducing non-linearity and allowing the a model to learn non-linear functions. In this particular assignment, I apply the Rectified Linear Unit (ReLU). It basically replaces all the negative values with zero.
  - Pooling: Pooling is a downsampling process that involves reducing dimensionality from a higher to a lower dimensional space. In machine learning, pooling is applied as a way to reduce the spatial complexity of the layers. This allows for fewer weights to be learned and consequently faster training times. Historically, different techniques have been used to perform pooling, such as average pooling and L2-norm pooling. The most preferred pooling technique is max pool. Max pooling involves taking the largest element within a defined window size.So for this assignment i have used GlobalMaxPooling1D.
- Neural network: The second section of a CNN is more task-specific. For the task of classification, this is a fully connected neural network. The input to the fully connected layer is a flattened vector that is the output of section one. Flattening converts the matrix into a 1D vector.In fig 3 you can see my build model summery.

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 48, 300)           4819200
_________________________________________________________________
spatial_dropout1d_2 (Spatial (None, 48, 300)           0
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 46, 100)           90100
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 44, 128)           38528
_________________________________________________________________
global_max_pooling1d_4 (Glob (None, 128)               0
_________________________________________________________________
dense_8 (Dense)              (None, 10)                1290
_________________________________________________________________
dense_9 (Dense)              (None, 5)                 55
_________________________________________________________________
dense_10 (Dense)             (None, 5)                 30
=================================================================
Total params: 4,949,203
Trainable params: 130,003
Non-trainable params: 4,819,200
```

Fig. 3. number of comments across the categories.

D. Results

Using Word2Vec and 1D Convolutional neural network i have build my model.After splitting and preprocessing the data i have created my own woed2Vec vocabulary model named "1100690-embedding-word2vec.txt" and save it for further use. Then i have build my convolutional model and add the Embedding layer in it. Then i have trained my model on training data and save this model named "1100690-1dconv-reg" to get the testing accuracy. After performing 50 epochs i

have got the testing acuracy 66.80 percent and after loding model again for testing i have got 58.86 percent testing accuracy. You can see the results in table.

Phase	Accuracy
Training	66.80
Testing	58.86

CONCLUSION

In this assignment, I studied a wide range of NLP classification models. My research and implementation consisted of two main parts. In the first part, I used the dataset "Rotten Tomatoes Movie Review" and applied the word2vec models to represent words numerically. In the second part, I implemented the convolutional neural network to train a multi-class sentiment analyzer. The training of standard CNN was computationally very fast. We showed that the 1D can achieve comparable accuracy.To Boost this accuracy we can use Google's pre trained word embedding model.

GitHub repository : https://github.com/karanshah10/NLP-Multi-class-Sentiment-Analysis-using-Deep-Learning-

REFERENCES

[1] Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis." Foundations and trends in information retrieval 2, no. 1-2 (2008): 1-135.

[2] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in Neural Information Processing Systems. 2013.

[3] https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras/

[4] Arman S. Zharmagambetov ; Alexandr A. Pak, Sentiment analysis of a document using deep learning approach and decision trees,IEEE

[5] Xi Ouyang ; Pan Zhou ; Cheng Hua Li ; Lijun Liu, "Sentiment Analysis Using Convolutional Neural Network," IEEE.

[6] Liang-Chih Yu,Jin Wang, K. Robert Lai and Xuejie Zhang, "Refining Word Embeddings for Sentiment Analysis, Department of Information Management, Yuan Ze University, Taiwan.

[7] Aliaksei Severyn,Alessandro Moschitti, Twitter Sentiment Analysis with Deep Convolutional Neural Networks,Publication:SIGIR '15: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval,August2015