Machine Learning Engineer Nanodegree

Capstone Proposal

Siddheshwar Kumar
April 12, 2018

Proposal

Domain Background

A couple of years back, I participated in my company Hackathon where I worked on the **sentiment** and **intent** analysis for a chatbot. I had used Stanford Natural Language Processing(NLP) library but, I could barely scratch the surface of NLP. I have also explored **sentiment analysis** on the tweets. In either case, I couldn't understand the field much, but it kept me intrigued and motivated to learn about it. And that's one of the reasons for me to do this course.

Sentiment Analysis is one of the interesting and exciting problems of Natural Language Processing. I plan to use this opportunity to explore this field and do something tangible. One common use of sentiment analysis is to figure out if a text, sentence or paragraph expresses negative or positive emotional feeling.

Sentiment Analysis technique can be used to understand the public perception of a brand or product based on online content and then decide appropriate business or marketing strategies. Political parties in Democracies are deciding their campaign strategies based on people's openions on social platforms (https://ieeexplore.ieee.org/document/7823280/).

Problem Statement

Perform sentiment analysis using machine learning techniques. On the internet, a lot of content is available which gives review or feedback of movies. Now, for a person to decide whether to go for a movie, is NOT so trivial if he/she has to go through countless tweets, Facebook posts/comments, and IMDB reviews.

Machine learning can play a big role in churning out data from different social media platforms and then just providing an overall positive or negative feedback of the movies i.e. *it's thumbs up or down*. Machine learning can also help to classify a movie review in more than two categories like average, good, very good, bad etc. As a user, it will be really helpful to just know a very objective feedback or rating based on the reviews provided by other users.

For this capstone project, I plan to use IMDB dataset and perform sentiment analysis. The goal is to predict whether a review is negative or positive given only the text.

Datasets and Inputs

I will be using the data from an old Kaggle competition **Bag of Words Meets Bags of Popcorn** (https://www.kaggle.com/c/word2vec-nlp-tutorial). This dataset contains 25,000 labeled training reviews, 50,000 unlabeled training reviews, and 25,000 testing reviews. **Unlabeled training** data and **testing data** doesn't have a *sentiment* field. For the project, I will rely on **labeled training** data for training as well as testing.

The file (*labeledTrainData.tsv*) is tab-delimited and has a header row followed by 25,000 rows and contains three columns/fields:

```
['id' 'sentiment' 'review']
```

- **id**: Unique identifier for each entry in the dataset; we don't need this field for modeling.
- **sentiment**: Contains binary values (1 and 0). 1 for positive and 0 for negative. This is the label of the model.
- **review**: a Detailed review of movies. This is the text or feature on which machine learning models will get trained.

I am planning to use 20% of data for testing and will report the accuracy of the model on this randomly selected dataset.

Input dataset: labeledTrainData.tsv

- *Dataset size for Training:* 20,000 (80% of 25,000)
- *Dataset size for Testing:* 5,000

Distribution of Target class

Distribution is Dataset is balanced. - *Data in label 1 (positive):* 12,500 - *Data in label 0 (negative):* 12,500

Above dataset will get randomly distributed into training and testing dataset. So distribution is still going to be (almost) balanced.

Solution Statement

I plan to use **Deep Learning** techniques as the final solution. *The number of words in the reviews are not fixed and also the meaning doesn't depend directly on words but also on the context*. The same word can have different meaning depending on the surrounding words. **Recurrent neural networks are feedforward neural networks augmented by the inclusion of edges that span adjacent time steps, introducing a notion of time to the model. It deals with time series data of variable length and has had good success off late in the area of natural languages.**

As part of this project, I plan to learn Recurrent Neural Network (RNN) and apply the learnings to this problem. But, at the same time, I would also like to explore how traditional machine learning techniques (non-deep learning) perform on this problem.

Benchmark Model

I plan to compare the performance of RNN (or LSTM: Long Short-Term Memory) with that of traditional approaches like Naive Bayes. I will apply Naive Bayes by preprocessing the text using TF-IDF and then running the multinomial Naive Bayes on the preprocessed outputs. This allows the algorithm to be run on the most prominent words within a document(https://pdfs.semanticscholar.org/a030/b72c0546a3c832627855d42bccc3f6819e75.pdf).

I plan to use roc_curve to calculate accuracy for Naive-Bayes and softmax classifier for RNN model. Also, this is a Kaggle problem and the leaderboard shows good success rate. So, planning to achieve the accuracy of at least **90%** on the test data.

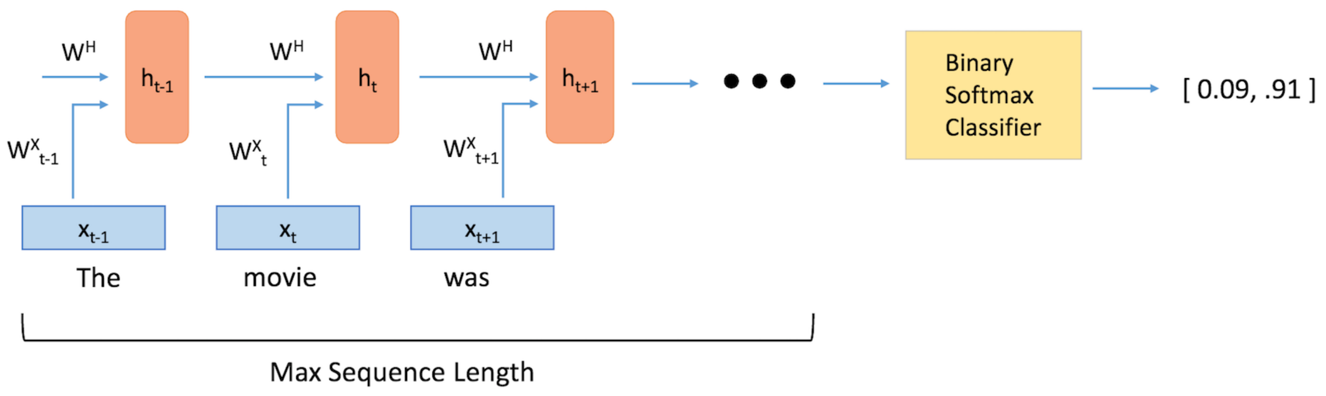
Evaluation Metrics

As discussed above I will apply traditional technique as well as Deep learning technique to evaluate this problem. And the final recommended approach should achieve at least 90% accuracy.

Project Design

Below are major components or stages of the modeling/training.

- **Preprocessing:** Before applying the data to modeling, I will ensure that it's preprocessed and in the right form.
  - Load the data in python and explore it to ensure that, data is randomized with the similar number of positive and negative sentiments.
  - Remove special characters like punctuations and stop words from the text. These characters/words are noise and they don't convey any meaningful input for sentiment analysis. They also increase the text size which will further slow down the modeling process.
- **Create Vocab:** This will help to know how many unique words are there in the whole dataset and based on that plan to take appropriate decision to optimize the model performance.
- **Word Embedding:** Machine learning mostly work on the vector (of integers or floats), so I need to transform the text into their number representation. One of the technique is to use **one-hot encoding** on the vocab. However, this encoding is inefficient, requiring as many bits as the vocabulary. Further, it offers no direct way to capture similarity aspect between words in the encoding itself (like it fails to understand that love and adore are similar words). I also plan to explore if pre-trained word vectors (like Word2vec, GloVe etc) can be used to quicken the modeling.
- **Modeling/Training:** I plan to model the problem using the traditional technique like **Naive Bayes** as well as Deep learning techniques **RNN/LSTM**. Below is a high-level diagram of how it performs sentiment analysis of text.



- **Testing:** Test the model and record the accuracy percentage on the test dataset.

Environment/Tools/Libraries

- Python 3.6
- Tensorflow 1.7
- Pandas
- Numpy
- Matplotlib
- Scikit-learn
- Nltk

References

- A Critical Review of Recurrent Neural Networks for Sequence Learning: https://arxiv.org/pdf/1506.00019.pdf
- Understanding LSTM Networks: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- The Unreasonable Effectiveness of Recurrent Neural Networks: http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- Kaggle Problem Reference: https://www.kaggle.com/c/word2vec-nlp-tutorial#description
- Perform sentiment analysis with LSTMs, using TensorFlow https://www.oreilly.com/learning/perform-sentiment-analysis-with-lstms-using-tensorflow