Capstone Project

Machine Learning Nanodegree

Pushpankar Kumar Pushp

February 28th, 2017

Project Overview

There is a lot of data on the web and is increasing exponentially. Organizing these data and extracting useful information out of these data is very important for tasks like predicting contents, discarding inappropriate content and more. Doing this task by hand is very difficult and time-consuming. Machine learning can be used to can be used to automate these tasks. Text classification task is very different from image classification because the meaning of a word depends on its context and meaning of the whole sentence can change with one word. Also storing all words on vocabulary by naive approaches like one-hot vector can be very space consuming as we have very large number of words in our vocabulary. These naive methods for storing words also does not reveal much information about a word.

In recent years, deep learning has been widely and successfully used for classification of images. Deep learning approach has surpassed all the traditional methods of image classification. Bo Pang et al. has compared different traditional machine learning techniques in his paper[1]. In this project, I compare traditional approach and deep learning approach for text classification task. I compare them based on their accuracy, precision, recall and F1 score of sentiment prediction of a text.

The approaches covered in this project can be used for similar tasks like predicting hashtags of a tweet, categorizing news article, clearly understanding meaning of a word based on its context, fetching relevant information for a search and more.

In this project, I have created and compared models to predict the sentiments of a movie review. This is a binary classification problem. The movie review can either be positive or negative. The review can be of arbitrary length. I predict sentiment by analyzing all words in a review.

I used IMDB movie review dataset[2] for this task. This dataset contains both labeled and unlabeled examples. It has a total of 100,000 multi-paragraph movie reviews. All the reviews are of different length. Labels in labeled examples are in binary format, 0 for negative sentiment and 1 for positive sentiment.

Most of the information about sentiment lies in commonly used words, unlike categorization where rare word tell us a lot about the class. So, I also compared different vocabulary sizes against their training time and other metrics.

First I predicted using Naive Bayes method to get a benchmark of this task. For this task, we need the probability of each word in a class. I get this probability by creating a bag of words which is the count of words in each class.

In next step, I use word2vec technique to create word embedding which is a dense vectorized representation of words. I train these the embedding using skip-gram[3] model. Then finally, I use these word embedding for training recurrent neural networks and random forest. I also try different types of memory cells in RNN like LSTM[4] and GRU. I am using these methods because the cells help in countering the vanishing gradient problem in recurrent neural network.

Metrics

I am using accuracy, precision, recall and F1 score as my metric for model evaluation. Accuracy is the proportion of correct classification out of total classification. Precision is the probability being the correct class given that our model has predicted the class. Recall is the probability of predicting a class provided that the data belongs to that class. And F1 score is the weighted average of the precision and recall.

Accuracy alone is not sufficient for this binary classification task. We might achieve high accuracy if there is a lot more positive class than negative class and model learns to always predict positive class or vice-versa. This issue can be overcome by using precision and recall too. I used F1 score along with precision and recall because F1 score can give combined result but precision and recall will help in visualizing the kind of mistakes the model is making.

Analysis

The IMDB dataset contains two types of data: labeled and unlabeled. The labeled dataset contains 25000 movie reviews and unlabeled dataset contains 50000 movie review. In labeled dataset labels are in binary format. 0 is used for negative sentiment and 1 is used for positive sentiment. The dataset also contains an id for each review which is irrelevant for our task.

The review contains HTML tags which is should be removed. Out of 25000 labeled reviews, 24904 are unique. Since almost all reviews are unique do not need to remove duplicates.

Statistics for number of characters in labeled reviews:

Count:	25000.000000
Mean:	1329.710560
Std:	1005.239246
min:	54.000000
25%:	705.000000
50%:	983.000000
75%:	1619.000000

max: 13710.000000

The unlabeled dataset has two columns: review and its id. This dataset can be used for unsupervised learning task like training word vectors. Out of 50000 reviews, 49507 reviews are unique.

Statistics for number of characters in unlabeled reviews:

Count:	50000.000000
Mean:	1334.618420
Std:	1006.913086
Min:	45.000000
25%:	707.000000
50%:	987.000000
75%:	1624.000000
Max:	14318.000000

The standard deviation of the number of characters in the review is very large. We will have to normalize the data by padding them. There are 74218 words in the dataset. This is a huge vocabulary size for a task like sentiment analysis. I tried different vocabulary sizes to save some computation time. Also, larger vocabulary size requires more space for storing it.

Here is a sample review:

"*The film starts with a manager (Nicholas Bell) giving welcome investors (Robert Carradine) to Primal Park . A secret project mutating a primal animal using fossilized DNA, like ¨Jurassik Park¨, and some scientists resurrect one of nature's most fearsome predators, the Sabretooth tiger or Smilodon . Scientific ambition turns deadly, however, and when the high voltage fence is opened the creature escape and begins savagely stalking its prey - the human visitors , tourists and scientific.Meanwhile some youngsters enter in the restricted area of the security center and are attacked by a pack of large pre-historical animals which are deadlier and bigger . In addition , a security agent (Stacy Haiduk) and her mate (Brian Wimmer) fight hardly against the carnivorous Smilodons. The Sabretooths, themselves , of course, are the real star stars and they are astounding terrifyingly though not convincing. The giant animals savagely are stalking its prey and the group run afoul and fight against one nature's most fearsome predators. Furthermore a third Sabretooth more dangerous and slow stalks its victims.

The movie delivers the goods with lots of blood and gore as beheading, hair-raising chills,full of scares when the Sabretooths appear with mediocre special effects.The story provides exciting and stirring entertainment but it results to be quite boring .The giant animals are majority made by computer generator and seem totally lousy .Middling performances though the players reacting appropriately to becoming food.Actors give vigorously physical performances dodging the beasts ,running,bound and leaps or dangling over walls . And it packs a ridiculous final deadly scene. No for small kids by realistic,gory and violent attack scenes . Other films about*

Sabretooths or Smilodon are the following : ¨Sabretooth(2002)¨by James R Hickox with Vanessa Angel, David Keith and John Rhys Davies and the much better ¨10.000 BC(2006)¨ by Roland Emmerich with with Steven Strait, Cliff Curtis and Camilla Belle. This motion picture filled with bloody moments is badly directed by George Miller and with no originality because takes too many elements from previous films. Miller is an Australian director usually working for television (Tidal wave, Journey to the center of the earth, and many others) and occasionally for cinema (The man from Snowy river, Zeus and Roxanne,Robinson Crusoe). Rating : Below average, bottom of barrel."

These review contains some HTML tags which are redundant for sentiment analysis task. The reviews also contains number which does not add much information to the sentiments of the review.

This review starts by explaining about the movie. Its sentiment is not clear exactly clear till the end. For this kind of review which is very long, striping it will affect its sentiment prediction. Padding all the reviews to maximum review length is required to achieve best results.

Exploratory Visualisation

There are few words that greatly affect the sentiment of the sentence. Their probability of occurring in one class is very high.

Word	Positive sentiment probability	Negative sentiment probability
'boring'	0.18	0.81
'worst'	0.09	0.91
'wonderful'	0.82	0.18
'spectacular'	0.73	0.26

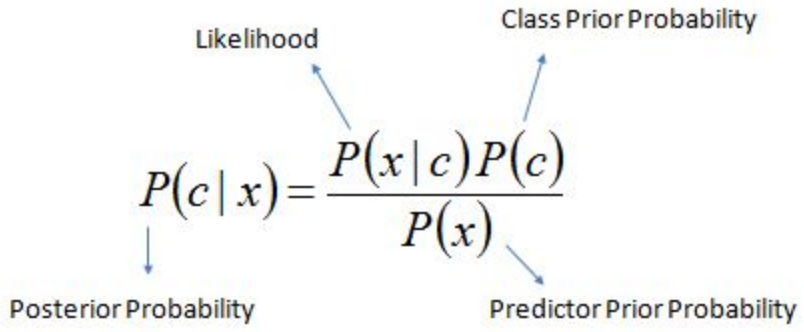
If we see words with high probability in once class then we can predict sentiment with very high probability. These kind of words makes other information redundant for this task. Word 'worst' has 91% probability of being part of negative sentiment review.

Creating a set of these words that has a high probability of being in one class and finding only them in reviews and discarding all other words can still give very high accuracy along with saving a lot on computation at runtime.

Algorithms and Techniques:

I classify movie review by three different methods:

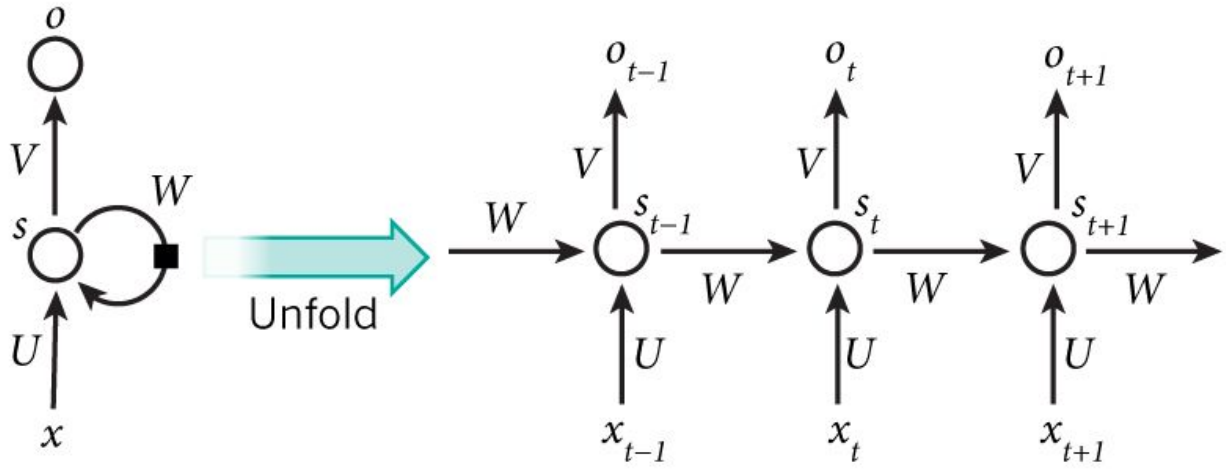
1. Naive Bayes - This method work by computing the probability of each word in a review. The total probability of a class is given by the following equation:-



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Naive Bayes assumes that the words are independent of each other which is not true. The meaning of a word can change based on its surrounding. This method can give good result in this case because there are very few words that determine the sentiment of the review.

2. Recurrent neural network - It is the type of neural networks whose output depends on its previous input.



It allows us to model arbitrary length sequences of data. This is achieved by performing same action on all elements of a sequence.

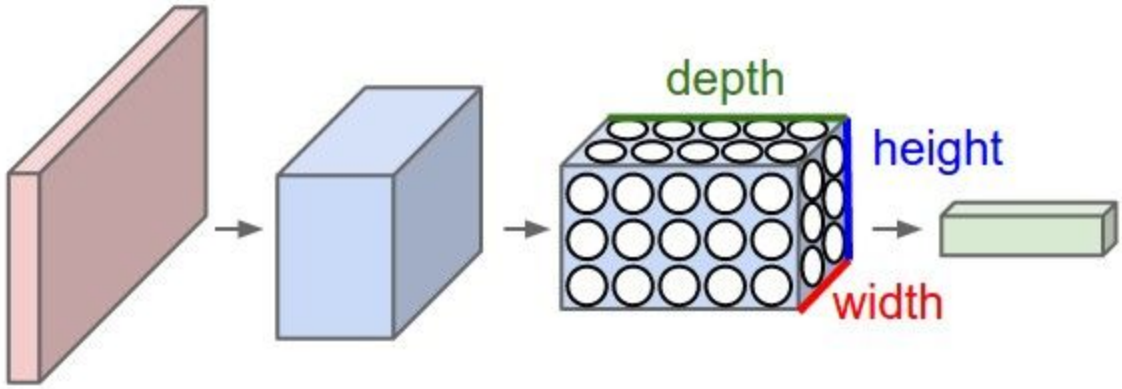
RNN takes two inputs at every time step: data at time t and its previous hidden state. It outputs a result at every time step. Hidden layer to hidden layer and input to hidden layer weight matrix are optimizer at the end of the unroll. The simplest optimizer is Gradient Descent optimizer. It calculates the gradient of a batch and changes the weight to reduce the loss.

These neural networks are very powerful and they can overfit the data i.e. they might work well on training data but might give poor result on unseen data. To avoid this we can use dropouts. Dropout is a technique of ignoring some of the random node of a layer. This helps in increasing the robustness of the model. But it should not be used between hidden layers because these layers are repeated. Randomly killing some of the neurons at every time step will result in killing all the neurons that are connected to layers few time step back. Dropout can be used on inputs on the RNN.

RNN has gradient vanishing and exploding problem. Vanilla RNN can not retain much information of inputs provided few time steps before. If the gradient is small then multiplying it multiple times result in very small gradient and if it is large then gradients will increase drastically. These shortcoming of RNN are overcome by LSTM which has a extra memory cell. It has gates to kill, learn and retain information of previous time steps. LSTM does not have vanishing gradient problem as it adds gradients to the memory cell instead of multiplying them. Exploding gradient problem can be overcome by simply capping the value of gradient.

This is heavily used for tasks like machine translation, language modelling etc. RNN is very suitable for text based data as the meaning and the inputs are time dependent. To predict the class the model should also consider all the input it has seen in the past and it does so perfectly.

3. Convolutional neural network:- Convolutional neural network is usually used for images. It takes an input of fixed dimensions and does some prediction. It does not remember input from previous time steps.



CNN identifies some feature of the input by scanning the input. It can identify certain features of data irrespective of their spatial location. It does so by scanning the data using a small patch. Usually smaller patches like 3*3, 5*5 are used. Multiple features of data is extracted at every layer. There are two techniques to reduces the spatial dimension of the input at every time step: pooling and using strides. In pooling, some neighbouring nodes are merged into one. Max pooling is most popular technique where highest activation of a patch is only considered. Stride is the step size of the patch. Stride can also be used with pooling. Dropout are usually not usually used convolutional layer instead it is applied at final fully connected layers.

CNN can be used in sentiment analysis by representing a review as vector. These vectors can be created by training a word embedding using techniques like skip-gram. To make the vectors of same dimensions the each review should be padded with special token. CNN can give good results as the sentiment of one review does not depend on others. Convolutional networks are easy and fast to train as it has very good gpu support by libraries like tensorflow and keras.

Benchmarks

Bo Pang et al. achieved accuracy of 78.4% using naive bayes in his paper. I also use Naive Bayes as the benchmark because there are few features of the review like high probabilities of a word being in a class. Naive Bayes can harness this property of the reviews to achieve high accuracy and outperform other techniques. The other benefit of Naive Bayes is that it is easy to implement.

Methodology

First the HTML tags in the reviews are removed using python library BeautifulSoup. Then I removed all the non-letter characters. Finally, I assigned each word an ID. ID is required to count the size of vocabulary, reduce the size of the vocabulary, convert them to one hot vectors. I converted the reviews to a lists of id of words it contains. I also removed stopwords like 'the', 'a', 'with' etc. which does not add any value to the sentiments. Removing stopwords does not affect metrics but reduces computations.

Naive bayes implementation

To predict a class using naive bayes probability of each word and the probability of the class is required. To get these, I created a bag of words and counted the number occurrence of each each word in both the classes. Then, the probability of a class can be calculated by dividing the total number of words in the class and total number of words in all class. Conditional probability of word given the class can be calculated by dividing the number of occurrence of word in the class.

$$P(C) = \frac{Count\ of\ words\ in\ c}{total\ number\ of\ words\ in\ all\ \ class}$$

$$P(d_i \mid C) = \frac{Count\ (d_i, C)}{\Sigma_i Count\ (d_i \mid C)}$$

But the probability of each word in a class is small because of large number of words in that class. This was causing the $P(D|C) \cdot P(C)$ to become 0 as the multiplications of many small

number created very small probability which was less than the precision of 32 bit floating number. I took all the probabilities to log space as I only had to compare the probability of positive sentiment and negative sentiment. Logarithm reduces the numbers proportionally, so this did not change the result and this helped in stopping the underflowing of probabilities.

Word vectors

Creating word vectors is an intermediate step for training recurrent neural networks and convolutional neural networks. It creates a very dense word embeddings from sparse one-hot vectors.

I used skip-gram model to create word embeddings. This model work by looking at the center word and predicting the surrounding words. I used NCE loss as it is helps in learning word embeddings more efficiently. Then finally I am normalizing the word embedding by taking its root mean square.

Recurrent neural network

I used the word embedding created in previous step to train the RNN. I also trained the embedding along with the classification. There was negligible drop in accuracy. This step requires dropouts as the word embedding 128 dimensional and the network overfits it. I used adam optimizer as this has fastest convergence. I also padded the sequence to get same dimensional reviews.

In final architecture, I padded the input so that each input has 120 words in it. If a review contains more than 120 words, I ignored remaining words. Then I fed it the model to first create an embedding. Then I used the used LSTM with dropout of 0.2. I used fully connected layer, which takes outputs of LSTM, to predict a binary class. I used sigmoid for converting the activation to probability.

Convolutional neural network

First I padded the preprocessed review data. Then I fed it to the model and made it to learn everything including the word embeddings. The model is inspired by Yoon Kim paper[3]. I used one 1D convolutional layer. Then I did max pooling to reduce the dimensions to half and finally I used two fully connected layers to classify the label. The first fully connected layer has 512 nodes with relu activation and only one node in layer two with sigmoid activation.

Increasing the number of words to used to create the sentence to 500 increased almost all the metrics by 2-3% in case of RNN and CNN. Increasing the number of epoch more than 3 does not help as the model converges in 2-3 epochs.

Refinements

I explored following refinements-

1. Review length- I initially kept the size of reviews to 128 words per review. I got 0.83 accuracy score and 0.82 f score. Then I increased it to 512 words per review gave the 0.876 accuracy score and 0.873 F1 score.
2. Decreasing the number of nodes in fully connected layer of CNN from 512 to 256 did not affect the scores but the training time decreased.
3. Adding more convolutional layer has no effect on measurement metrics.
4. Decreasing the vocabulary size to 5000 from 20000 increased the accuracy by 1%, recall by 4% and F score by 2%. The precision has decreased by 2%.
5. LSTM and GRU gives almost the same result.
6. Decreasing the word embedding size to 32 did not affect the result and is more desirable because of the computation cost.

Results

Model evaluation and Validation

CNN- In final model, I am considering only 5000 most frequent words and padding each review to 512 words. I am using one 1D convolution network with filter length of 3. These convolutional networks extracts 128 features input. I am also using max pooling before the fully connected layer. The fully connected layer has 256 nodes.

RNN- I am using LSTM cells to avoid gradient problem. The vocabulary size here is 30000 as lower size affected the metrics. All the review has been padded to 512 words. The outputs of LSTM are directly used for binary classification using a fully connected layer which has only one node. I am using sigmoid activation function.

Model	Precision	Recall	Accuracy	F score
Naive bayes	**93.49%**	88.69%	**91.26%**	**91.03%**
CNN	86.85%	**90.04%**	88.22%	88.10%
RNN	84.32%	84.30%	84.28%	83.92%

Justification

Naive Bayes has outperformed all the other models. Naive Bayes is giving the best results here because there are few words that determine the sentiment of whole sentence.

Naive bayes does only try to find those words. This feature was also shown by CNN when we decreased the vocabulary size the model's performance increased. Decreasing vocabulary size forces CNN to only find most relevant word for sentiments. RNN was not able to outperform because it tries to find the meaning of whole sentence first before determining the sentiment of the sentence. This is not required for sentiment analysis. RNN will be very suitable for more complex task.

CNN has slightly lower accuracy than compared to the naive bayes but is fastest to compute because of its efficient implementation in libraries and their gpu support. CNN is very robust as its accuracy and other metric scores are not affected much by small changes, like changing embedding size, in the model.

Naive bayes will perform poorly if the sentence contains a lot of negative word but still positive. It has best scores in all type of metrics but longer training time than other model because of gpu support.

Conclusions

Free-form visualisation

Although Naive Bayes has the best accuracy but it does not work well on reviews where the most of words are of positive sentiment but sentence as a whole is negative. But only RNN was able to correctly classify reviews like:

> *"This film should be brilliant. It sounds like a great plot, the actors are first grade, and the supporting cast is good as well, and Stallone is attempting to deliver a good performance. However, it can't hold up"*

Word like *brilliant* has very high positive sentiment and Naive Bayes only look at these things. CNN was also not able to classify it correctly. But RNN looks at whole sentence and considers all the words and their order, so it was able to correctly classify the sentiment of this kind of review.

Reflections

The goal of this project was to find a high accuracy and robust model. Its result are very surprising for me as I expected Naive Bayes to perform least well and choose it as base model. Since Naive Bayes has outperformed other techniques, this means that the sentiments of a sentence mostly depends on the word it contains instead of their arrangement and connections.

Sentiment of sentence is determined by a small set of words which is very small compared to vocabulary size. So, for this task we do not need to model the language and understand it. We only need to keep eye on few words that determine the sentiment of a sentence. Considering whole sentence is important to determine the sentiment of the sentence.

The results shows that the sentiments can be determined with high accuracy just by considering only 5000 most frequent words.

Improvement

The score of RNN and CNN might improve if the features of both model can be merged. Tough examples like “This was very long and dull movie. This movie can bore many people but I liked it” can be correctly classified by RNN and CNN can give robust models in other cases. Also ensemble of multiple model helps in achieving slightly more accuracy. To create an ensemble of RNN and CNN, output of CNN can be fed to RNN instead of feeding it to fully connected layers.

References:-

1- Thumbs up? Sentiment Classification using Machine Learning Techniques | https://arxiv.org/pdf/cs/0205070.pdf
2- Data source | https://www.kaggle.com/c/word2vec-nlp-tutorial/data
3- Distributed Representations of Words and Phrases and their Compositionality | https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf
4- LSTM: A Search Space Odyssey | https://arxiv.org/pdf/1503.04069.pdf
5- IMPLEMENTING A CNN FOR TEXT CLASSIFICATION IN TENSORFLOW | http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/
6 - Convolutional Neural Networks for Sentence Classification | https://arxiv.org/abs/1408.5882