Text Classifier Using Max Entropy and Multilayer Percetron

Ying Zhu

1 Data Engineering

1.1 Newsgroup

Data is parsed into a pandas data frame with each row containing a string of text going through the syntax and contextual cleaner.

- Syntax Cleaner
  The syntax cleaner outputs a list of words that are clean in syntax. The syntax cleaner handles:
  - remove digits and special characters
  - remove xml tags
  - replace newlines with spaces
  - remove zeros
  - split on 1+ spaces
  - remove single character
  - convert word to lower cases
- Contextual Cleaner
  The Contextual cleaner converts words back to standard forms and gets rids of standard email attributes, it contains:
  - Email attributes clean. Compile a list of common address words used in email such as from, to, cc etc.
  - Lemmatize and stem each word
  - Filter out stop words as provided by NLTK package
  - Filter out human names (omitted for final output due to long processing time)

1.2 Propername

Minimal Data engineering is done. All characters are kept for feature processing

2 Feature Generation

- N-gram
  N-grams features are used for both propername and newsgroup data. We compile a dictionary of characters/words and convert them into our columns
- Cutoff Frequency
  (Combination of) Characters/Words with frequency lower than a certain cutoff value will be dropped from the feature vector. The reason is that we would like the algorithm to handle unseen words for testing set.
- Binary
  Use binary representation vector for each row of data. This will ignore the frequency that a character or word shows up in a specific row of data
- Bag of Words
  Instead of consisting of only 1 and 0s, the feature vector now contains a vector of numbers that represents the frequency of each words/characters for a specific row.
- Composite N-gram
  After combining several N-grams together(with a different N for each) , we achieved a richer representation of data and better performance.

3 Model Building

- Perceptron
  Learning Rate
  r is defined as the learning rate of the perceptron learning. By deploying a learning rate smaller than 1, we avoid the oscillating behaviour of the loss when approaching local minimum
  Averaged Perceptron
  The performance of vanilla perceptron algorithm is partially determined by the order that each training sample shows up. In general, it overfits on data that shows up later in the training process and therefore the weight vector w exhibits the behavior of wandering around and does not generally converge.To help with convergence of the weight vector, we used averaged weight as defined below:

$$w(t) = \frac{1}{T}\sum_{i=0}^{T} w(i)$$

  Computing an averaged weight is costly for computation. Mathematically, one can prove that the following algorithm is equivalent to the averaged perceptron yet does not require memory storage of historical weight vector and recomputation of average weight:

Initialize w = (0,0,0....0),t=1,S=0
  for each epoch:
    for each training sample$\phi(x_i, y_i)$:

$predict\ y^* = argmax w * \phi(x_i, y_i)$
$if\ y^*\ not\ equal\ to\ y:\ g = \phi(x_i, y_i) - \phi(x_i, y^*)$
$w = w+rg$
$S = S+(t\text{-}1)rg$
$t = t+1$

Random Reshuffle
At the end of each epoch, a random shuffle is performed on the training samples to prevent entire mini batches of highly correlated example. We have observed that in practice, the performance did not differentiate much from ones without shuffling.

- MaxEntropy
Performance Improvement:
  - We implement all the computations in max entropy using sparse matrix multiplication,which reduces training time dramatically.
  - In order to solve the overflow problem, we subtract the maximum value of $w * \phi(x, y)$ for each data point
  - We apply 1/N to average the gradient magnitude to account for size of our data.

- Multi-layer Perceptron
Trainer
We experiment our training model with three trainers in multi-layer perceptron on both propername and newsgroup data. After comparing, we found that AmsGrad trainer performs better than the other two.Specific accuaracy comparison could be found in table II in Parameter tunning section.
  - Adam trainer
  - AmsGrad
  - Simple Stochastic Gradient Descent

Activation Function
Propername : We implemented two layers multi-layer perceptron using relu, tanh and softmax.
$softmax(v * tanh(q * dropout(rectify(w * x + b), 0.3) + b1))$
Newsgroup: We implemented two layers multi-layer perceptron using two tanh and softmax.
$softmax(v * tanh(q * dropout(tanh(w * x + b), 0.3) + b1))$
Our reasoning for choosing Relu, tanh and softmax are follows:
The RELU activation function brings in two benefits during training:
  - It provides much faster convergence since it zeroes out a large amount of non-active nodes
  - The " zeroed-out" hidden nodes will not be activated again during the training process, which simplifies the network structure and provides network sparsity, we believe that the network sparsity brings the following benefits:
    a) sparse representation is more likely to be linearly separable
    b) It disentangles information more efficiently. A sparse representation of input feature vector is likely be more robust handling small changes of input data. In contrast, a dense feature representation format would entangle information between different factors and a minor change in input are likely to cause huge variations in the output.
    c) It provides efficient representation of training samples since it performs dimensionality reduction

The SOFTMAX function is specifically suited for multi-class classification tasks. Different from logistic function which suits better for binary prediction, the softmax function output a value between 0 and 1 for each unit, similar to a sigmoid function. In addition, it divides output where total sum of them are 1. In a word, SOFTMAX function provides a probability of each class being true.

The TANH function is chosen due to the following reasons:

a) tanh function has stronger gradients but not necessarily make the network dead like RELU
b) It avoid bias in the gradients.

Loss Function
Since we compare float number to generate loss, we choose negative log as our loss function.
Performance Improvement
MLP overfits easily on newsgroup data, we take the following measures to control the overfitting problem:
  - Introduce epoch to increase model prediction accuracy. After observation, we reduce epoch training iterations from 20 to 10 to prevent overfitting
  - Use batch size of 30 to speed up training process
  - Use lookup parameter to reduce the size of our computation graph and speed up learning process
  - Reduce feature complexity by by reducing composite Ngram to contain only 1-gram and 2-gram
  - Reduce network hidden nodes
  - Add dropout rate of 0.3
  - Utilize averaged loss function instead of cumulative loss function

4 Parameter Tuning

Methodology
Parameter Tuning has played an important role in achieving a favourable performance. We consider the following factors that might influence the performance and performed a greedy strategy on identifying the best set of parameters. We run each combination 3 times and compute average as measurement statistics. This is illustared with the propername dataset:

- Data Layer:
  - Representation of features: binary or frequency of characters and words
  - Number of features: (n,m) gram is defined as combination of (n,n+1,n+2,n+3,..n+m) grams as composite features

- Neural Nets Configuration
  - MLP activation function: rectifier, tan, logistic and relu, we always choose softmax as our outmost layer due to reasons illustrated in the previous sections
  - Trainer Function : simple gradient , adam trainer,etc
  - Network Topology: we have fixed a 3 layer network and could adjust the number of nodes for hidden layers; we pick different combinations of nodes (128,256,512) , (64,128,256) , (32,64,128)
  - We adjust the dropout rate from 0.1 to 0.3
  - We can iterate either 5 or 10 times

Identify the influence of features representations and activation functions :(training accuracy, dev accuracy)

Epoch = 5	(Tanh,tanh) Frequency	(Tanh,relu) Frequency	(Tanh,Tanh) Binary	(Tanh,relu) Binary	(relu,relu) Binary
adam trainer Ngram (2,2)	0.8437, 0.7812	0.9053, 0.79191	0.9401, 0.8216	0.9624, 0.8279	0.9648, 0.8220
adam trainer, Ngram (2,4)	0.8474, 0.7888	0.8960, 0.7871	0.9925, 0.8462	0.9950, 0.8559	0.9930, 0.8448
adam trainer, Ngram (2,6)	0.8386,, 0.7822	0.8888, 0.7777	0.9949, 0.8524	0.9967, 0.8562	0.9917, 0.8476

Figure 1: Table1

Tabel 1 Observation:

- Combination of Tanh and RELU function perform well on propername data
- The more composite N-gram we add, the better performance we achieve.
- Binary representation works significantly better than frequency representation for propername data.

Epoch = 5,Ngram=(2,6),Binary	(32,64,128)	(64,128,256)	(128,256,512)
Adam trainer	0.9964, 0.8600	0.9980, 0.8635	0.9967, 0.8562
Simple SGD	0.9497, 0.8272	0.9980, 0.8566	0.9864, 0.8465
AmsGrad	0.9972, 0.8569	0.9988, 0.8621	0.9985, 0.8607

Figure 2: Table2

Ngram=(2,6),Binary, AmsGrad, (64,128,256)	Dropout=0.1	Dropout = 0.3	Dropout=0.5
Epoch 5	0.9992, 0.8662	0.9983, 0.8635	0.9978, 0.8517
Epoch 8	0.9993, 0.8628	0.9994, 0.8572	0.9987, 0.8597
Epoch 10	0.9994, 0.8572	0.9993, 0.8583	0.9988, 0.8600

Figure 3: Table3

Table 2 Observation:

- Adam and Ams grad trainer perform better than simple GSD with a epoch equals to 5
- Topology does have a significant effect on performance, we want to design a network that is not too simple nor over complicated

Tbale 3 Observation:

- As epoch (>5) increases, the trainer overfits on training data
- Dropout value could help with preventing overfitting

In general, the MLP classifier overfits the data. We applied techniques following methods listed in previous sections to tackle such problem to reach a favourable solution. We achieved a dev accuracy of 0.8645 using optimzied parameter set.

5 Error and Performance Analysis

We perform error analysis based on the multi-layer perceptron classifier.

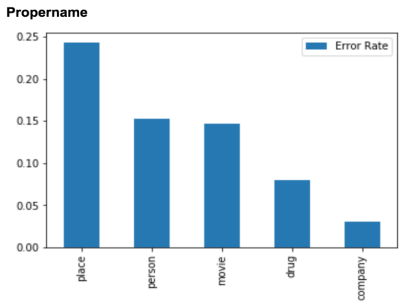

Figure 4: Propername classification erro rate

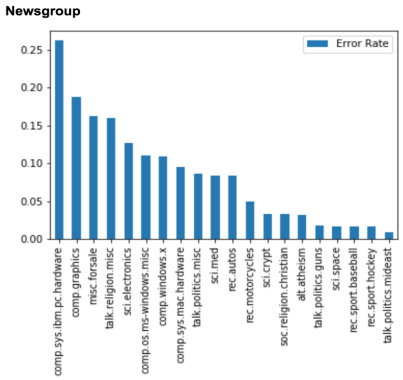

Figure 5: Newsgroup classification error rate

We perform error analysis based on the multi-layer perceptron classifier.For propername data(Figure 4,6,7) the classifier has only

75% accuracy predicting places. Here are a few wrongly classified examples:
[8,′ $BlaenauFfestiniog$′,′ $place$′],
[11,′ $Barry$′,′ $place$′],
[16,′ $Bristol$′,′ $place$′],
[20,′ $Easingwold$′,′ $place$′],
[28,′ $Belgium$′,′ $place$′],
[102,′ $Beaugency$′,′ $place$′],
We found it is non trivial to differentiate such examples from a human perspective. It is easy to miss-classify places or person as movies, since movies could generally contain place or human names. A possible way to improve on the result is to create dictionary that would capture words that's likely associated with place,movie and person names.Such lookup method was tested costly time-wise and was omitted for feature generation

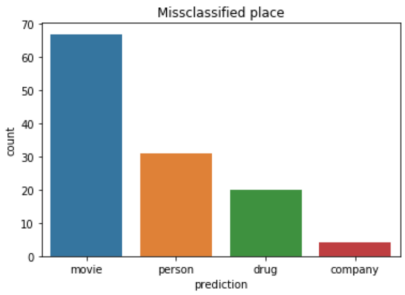

Figure 6: Miss-classified classes

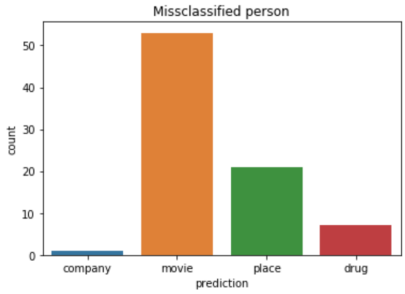

Figure 7: Miss-classified person

Regarding newsgroup data(Figure 5 and 8), not surprisingly, pick the most miss-classified category: computer system hardwares, the text is most easily mixed with ms-windows, electronics and hardware texts.

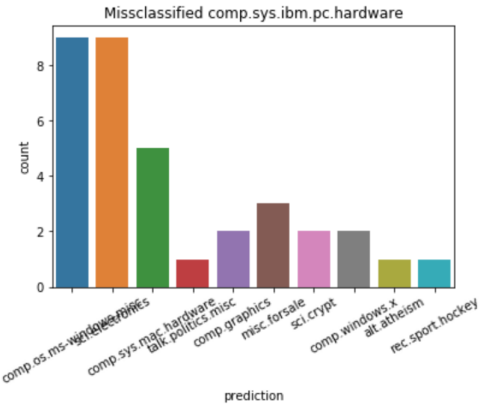

Figure 8: Most miss-classified class in news group

6 Final Performance

We observed that there is a large discrepancies among train,development and test set accuracy for newsgroup data. We suspect that it is due to the following reasons: 1. Overfitting on the training data. 2. test set contains many computer system related classes, which is hard for our classifiers to predict. We could further check our data cleaners to see if there is systematic destruction of computer-related syntax 3. The sampling processes for newsgroup test and development data set are different

Best Performances	DataSet	Train	Dev	Test
Perceptron	Propername	0.9848	0.8645	0.8498
	Newsgroup	0.9986	0.8811	0.7799
MaxEnt	Propername	0.9944	0.8514	0.8400
	Newsgroup	0.9997	0.8758	0.7728
MLP	Propername	0.9983	0.8645	0.8400
	Newsgroup	0.9986	0.9160	0.8205

Figure 9: Final Result