CS5740: Assignment 1
https://github.com/cornell-cs5740-20sp/assignment-1-fixed-kushandjay

Jack Rossi
jdr342

Kushal Singh
ks2377

1 Introduction (5pt)

This paper explores the text classification problem using Perceptron learning, Maximum Entropy Learning, and multi-layer Perceptron learning. The data comes from two corpora, news documents and proper name samples.

We implemented and tested the performance of several featurizations of the data. For news documents, we used bag of words, bigrams, and document length, and for proper names we used letter-grams. For news group classification, we found that the Perceptron with bag-of-words performed best. Similarly, the Perceptron was best able to classify proper names, this time using letter trigrams.

2 Features (20pt)

Bag of words and n-grams were expected to be very sparse features, so we represented features for a given example as a dictionary, where entries represented non-zero features, and everything else was zero.

Word n-grams We featurized the newsgroup dataset using word n-grams with n values of 1 and 2. the word n-gram with n = 1 is the same as bag of words, but sometimes our 'words' were sets of special characters separated by spaces. A document and vectorization example from the misc.forsale group follows.

> "From: lorne@sun.com (Lorne R. Johnson - Sun IC Region SE) Subject: WARRIORS TICKETS FOR SALE Organization: Sun Microsystems, Inc. Lines: 22 Distribution: ca Reply-To: lorne@sun.com NNTP-Posting-Host: normajean.west.sun.com ********** * WARRIORS TICKETS FOR SALE * ********** I have 2 tickets that I can't use (Last pair this year). Section 109, Row P, Seats 8 9 DAY DATE OPPONENT TIME — —- ——— —- WED 4/21 Sacremento 7:30 Price: 45.00 = MY COST Call or email if you are interested in these tickets. Lorne Johnson lorne@sun.com (408) 562-6003 "

{'From:' : 1, 'lorne@sun.com' : 1, '(Lorne' : 1, ...}

Letter n-grams We featurized the proper name dataset using letter n-grams of size 1 through 3. These letter n-grams consisted of adjacent letters and/or special characters. An example from the proper names corpus and the 2-gram letter vectorization follow:

> Two Days, Nine Lives

{ 'tw' : 1, 'wo' : 1, 'o ' : 1, ' d' : 1, ...}

Document length One of the features we tried was the length of the proper name. However, this feature appeared to have detrimental results on our accuracy. One possible hypothesis is that a lot of proper names have the same word length, but different n-grams. Therefore, the likelihood that those words are similar increases, due to the match in the word length feature. We can corroborate this assumption by looking at the corpus of proper names, where the key is the length of the word, and the value is the number of proper names with word length equal to that key.

('Corpus: ', 1: 3, 2: 16, 3: 91, 4: 232, 5: 517, 6: 954, 7: 1345, 8: 1506, 9: 1490, 10: 1419, 11: 1389, 12: 1367, 13: 1356, 14: 1251, 15: 1154, 16: 1010, 17: 904, 18: 788, 19: 666, 20: 594, 21: 536, 22: 445, 23: 428, 24: 388, 25: 333, 26: 289, 27: 281, 28: 281, 29: 234, 30: 205, 31: 158, 32: 146, 33: 147, 34: 132, 35: 106, 36: 112, 37: 87, 38: 76, 39: 78, 40: 59, 41: 57, 42: 51, 43: 38, 44: 47, 45:

Dataset	Count
Newsgroups	
Train	9051
Test	2263
Propernames	
Train	23121
Test	2894

Table 1: Size of training and development sets for Newsgroups and Propernames.

37, 46: 28, 47: 29, 48: 29, 49: 30, 50: 37, 51: 24, 52: 19, 53: 11, 54: 11, 55: 14, 56: 10, 57: 7, 58: 14, 59: 5, 60: 8, 61: 8, 62: 2, 63: 4, 64: 1, 65: 4, 66: 3, 67: 3, 68: 2, 69: 4, 70: 3, 74: 1, 76: 2, 78: 1, 79: 1, 81: 1, 83: 1, 88: 1.

It is apparent that there are a lot of proper names with word length $\in [7, 16]$.

3 Experimental Setup

Data (5pt) Table 1 shows the high-level statistics for the datasets. The Proper Name and Newsgroup datasets are both split into three subsets: training, development and test. The training, dev, and test for Proper Names have 23121, 2893, 2862 samples, respectively. Each subset contains proper names belonging to one of five different categories; person, place, movie, drug, company. The training, dev, and test for Newsgroup have 9053, 2263, 7530 samples, respectively. Each subset contains articles belonging to one of six different themes: comp, rec, sci, politics, religion, misc. These can be further categorized into 20 categories.

Data Preprocessing (5pt) News data was tokenized and lower cased. Special characters were not removed, because we felt they would have different distributions between news groups. For example, strings of special characters might indicate page style that was characteristic of a particular news group. Tokenization split strings on spaces.

Proper name data was also lower cased but it was not tokenized. Special characters were again retained.

Perceptron Implementation Details (3pt) Our Perceptron model maintained a weight vector for each class in the text classification problem. The weight vectors were represented as dictionaries for consistency with the feature representation. Training of the model requires specifying the number of epochs, training data, and development data. After each epoch of training, the model calculated accuracy on both the training and development data. This information allowed us to retrain using the smallest number of epochs that provided high performance on the development set.

MaxEnt Implementation Details (3pt) For our MaxEnt model, we used SciPy's optimization function: *scipy.optimize.fmin_l_bfgs_b*. We passed in $self.fn\_log\_likelihood$ as the loss function, which uses softmax to calculate the probabilities of a particular label, given the dot product between the query instance and the weights dictionary. We also passed in $self.fn\_log\_likelihood$ as the derivative with respect to the weights of the loss function. The final result was an estimation of the position of the optima. With regards to hyperparamters, we set the *maxiter* parameter in the optimization function above equal to 20, as that yielded the best results from experimentation. We also had to set the weights dictionary to be $= (4 * (len(corpus) + 1) * dimensions)$, since this was the minimum size that ensured no indices were out of bounds.

MLP Implementation Details (10pt) In terms of architecture for the computation graph, we used 2 total layers (250 dimensions for each), with tanh as the activation function in the first layer and softmax in the final output layer, ultimately choosing the label that yielded the highest probability from softmax. Through experimentation, we found that tanh for activation in the first layer yielded better results than sinh and cosh. The optimizer we used was the AdamTrainer, as this yielded better accuracy results compared to the SimpleSGDTrainer, MomentumSGDTrainer, and AdagradTrainer. Our learning rate was set to 0.001, which is the default rate for AdamTrainer. Our stopping criteria is determined by the number of epochs through the training data. Through experimentation, we concluded that the loss and accuracy appeared to stagnate after around 8-10 epochs. We also used a lookup variable that maps numbers to vectors, specifically for word embedding. We decided to have 7000 rows, each with 10 dimensions, as this ensured that we did not have any lookups out of bounds.

Model	Test Acc
Newsgroups	
Perceptron BoW	0.6013
MLP, BoW, Alphanum	0.4782
MaxEnt, BoW, Alphanum	0.0521
Propernames	
Perceptron Trigrams	0.7795
MaxEnt Bigrams	0.6359
MLP Bigrams	0.6512

Table 2: Test accuracy for final chosen models. Legend: BoW = Bag of Words; Alphanum = text is reduced to alphanumeric characters only.

Model	Dev Acc
Newsgroups	
Perceptron BoW	0.7101
Perceptron Bigrams	0.6893
Perceptron, BoW, Alphanum	0.7021
Perceptron, Bigrams, Alphanum	0.6999
Propernames	
Perceptron Unigrams	0.6013
Perceptron Bigrams	0.5499
Perceptron Trigrams	0.7912

Table 3: Accuracy of Perceptron ablations on both datasets.

4 Results and Analysis

4.1 Proper Name Classification (11pt)

Table 2 shows the test performance of the selected Propernames classifiers. The Perceptron performed best with accuracy of 0.7795, followed by MaxEnt with accuracy of 0.6359, and finally MLP with accuracy 0.6512.

In testing various alternatives, we found that trigrams worked best for the Perceptron, and then Bigrams worked best for both MaxEnt and MLP. And example of the ablation analysis is presented in Table 3. We also found that adding the length of the proper name as an additional feature did not improve the dev accuracy.

From our results, we noticed that the labels for people and movies were often classified incorrectly. Since we used n-Grams as our feature, this observation makes sense. From the training data, we see that a lot of movies have places in their titles: *Going to California, Adieu Babylone, California Girls, Loin de Syracuse, Hollywoodrymlingar.*

4.2 Newsgroup Classification (11pt)

Table 2 shows the test accuracy for the best fit newsgroups models. The perceptron model with bag of words featurization led in accuracy with a score of 0.6013. Next up was MLP, with an accuracy score of 0.4782. Last was MLP, with an accuracy score of just 0.0521. This last test score is hypothesized to result from a bug in the Max-Ent model when running newsgroup features. Table 3 shows the ablations we tested for newsgroup classification. We ended up using bag of words with all characters included because it provided the highest accuracy on the development set, a score of 0.601.

5 Conclusion (3pt)

Overall, word and character $n-grams$ seemed to be a very good baseline feature, regardless of the model or dataset. The Perceptron algorithm yielded the best results on the test set, compared to Max-Ent and MLP (Multi-layer Perceptron). However, MLP yielded better results on the training and dev set, compared to Perceptron and Max-Ent. For the future, identifying and experimenting with additional, nuanced features, such as ratio of vowels to consonants, ratio of consonants to vowels, number of syllables could result in better accuracy.