CS5740: Assignment 1
Github Repository Link

Jiayun Liu	Rashmi Sinha	Dan Witte	Haoran Yin
jl3935	rs2584	dw592	hy522

1 Introduction (5pt)

We are asked to build two text classifier models - the perceptron and MLP, for two different NLP tasks. The first task is to assign proper nouns to one of five classes. The data includes strings of proper nouns. The second task involves predicting the newsgroup that an email document belongs to. The data includes email headers in addition to the actual body.

For the propernames dataset we experimented with n-gram character and word length features. For the newsgroup dataset we experimented with bag of words and word n-grams. We observed that the best performing model is the MLP with character N-grams for propernames with an accuracy of 85.7% and the MLP with the bag of words for the newsgroup with an accuracy of 81.7%.

2 Features (20pt)

For propernames, we used character n-gram featurization with tail end multi-gram featurization. For example, for n = 4, 'Malham' has the following grams - 'Malh', 'alha', 'lham', 'ham', 'am', 'm'. When the number of letters in the word is less than n, we started finding all possible grams in the leftover word. We also created four additional features: '1 word present', '2 words present', 'more than 2 words present' and 'digits present'. We also had a bias term to increase model flexibility.

For the newsgroup dataset, we used the bag of words (BoW), as it yielded the best results for our MLP implementation. We also included a bias term. Unlike the case for propernames, we also removed stop words from the feature list, stripped out the special characters and additional delimiters. We also experimented with n-grams. For example, for sentence "Hello, NLP TA's!", the BoW features would look like ["*hello*", "*nlp*", "*ta*", "*s*"].

3 Experimental Setup

Data (5pt) The proper name dataset includes alphanumeric characters, punctuation, and non-English characters. There are 23,121 train, 2,893 dev, and 2,864 test samples. The names are from one of five categories: person, place, movie, drug, or company. The dataset includes 24,647 unique words. The length of the names varies from 1 word to 14 words, and 1 character to 29 characters.

The newsgroup dataset have 9,051 train, 2,263 dev, and 7,532 test samples. Each document sample contains both the email header and body and belongs to one of twenty classes partitioned into six themes. After lowercasing all alphabetical characters the dataset includes 223,951 unique strings. The document length is about 320 words on average, varying from 10 words to 16,602 words.

Data Preprocessing (5pt) For the propernames dataset, since we observed that each sample only consists of a few words, we wanted to limit the amount of processing, as any processing leads to loss of information. We found that lowercase conversion decreased accuracy, as the position of the capitalization of alphabets would play a deterministic role in identifying the final class. Even the dominant presence of special characters such as hyphens in drugs can give the model valuable insights. Hence we made a conscious decision through empirical analysis to not do any preprocessing for the propernames dataset.

For the newsgroup dataset, we tokenized each email into words split by a single space. Prior to tokenization we experimented splitting words at additional delimiters, such as apostrophes, colons, and other punctuation. We also experimented with lowercasing words, removing stop words and special characters prior to tokenization. After tokenization, we have created a dictionary for all unique words and implemented either a binary

System	Accuracy
Development Results	
Perceptron (full model)	86.40%
Perceptron w/o Digits presence	86.00%
Perceptron w/o Length of words	85.70%
Perceptron w/o Word splitting	80.30%
Perceptron 4-Gram Word splitting	75.30%
MLP (full model)	88.21%
MLP with 3 hidden layers	87.89%
MLP with dropout layers	20.87%
Test Results	
Perceptron	83.7%
MLP	85.7%

Table 1: ProperNames result

BoW or a binary n-gram. We then converted each sample into a vector representing the bag of words or n-gram.

Perceptron Implementation Details (5pt) The perceptron model has a weight for every class and feature and one additional weight per class for the bias. All weights are initialized to be 0. We have to two hyperparameters for the perceptron model: learning rate and number of epochs. For the perceptron implementation, we make use of block feature vectors. To increase efficiency, we made use of hash table as the vectors in block format structure is very sparse. For the dev set and test set, we only relied on the features computed in the train set.

MLP Implementation Details (8pt) We implemented the MLP model with PyTorch neural net modules.The model contains 2 hidden layers with 32 features. We choose ReLU as the activation function, ADAM as the optimizer, and CrossEntropyLoss as the loss function.

We also created the corresponding PyTorch dataloader, which will load the data in batches in shuffled order. The default batch size we choose is 64. The learning rate for the training is $1e-3$ and the training will stop after 10 epochs.

4 Results and Analysis

4.1 Proper Name Classification (12pt)

Table 1 contains the results for the propername dataset. We initially applied the perceptron model only with n-character gram, with n from 2 to 5. We observed that the model overfits beyond n=4. Initially we split all the words and applying n-character gram individually. However, when we applied the n-character gram on the entire entity together, the dev accuracy improved.

Observing the dev set qualitatively for classification, we found that the place category contained mostly of single word entity, and names consisted of 2 words entity. Hence, we thought it useful to include features such as whether the entity contains 1 word, 2 words or more than two words, i.e. the word length feature.

Finally it was observed that the drug class also consisted of digits and hence we added the feature of containing digits to result in slight improvement of the dev accuracy.

We used similar features for the MLP implementation, then we added the drop-out layers with drop-out rates ranging from 0.1 to 0.8, but the accuracy decreased significantly. We then tried different numbers of layers, layer sizes, and activation functions, and observed that the model performed the best using the ReLU activation function with 2 hidden layers and 32 features.

For the qualitative analysis, we found that the company and drug performed significant better than the other three categories. We hypothesized this is because companies generally have indicative words such as 'Ltd', 'Corp', 'LLC', and drugs have idiosyncratic grams and digits. On the other hand, the model struggled to distinguish between movies, place and person. This is because movies can also be based on a person name or place name. Similarly places generally don't have any specific distinguishing feature about them. Some of the miss-classified examples are 'Driven', which is actually a place, but classified as a movie and 'Marion Davies' which is a movie and classified as a person.

4.2 Newsgroup Classification (12pt)

System	Accuracy
Development Results	
Perceptron	84.44%
Perceptron with sanitized text	86.56%
Perceptron 2-gram	83.73%
Perceptron 3-gram	84.82%
Perceptron 3-gram w/sanitized text	86.60%
MLP (full model)	91.52%
MLP with dropout layers	23.45%
MLP with 3 hidden layers	90.32%
MLP with lowercase conversion	90.17%
Test Results	
Perceptron	75.70%
MLP	85.71%

Table 2: Newsgroup result

To set a baseline for the newsgroup perceptron, we trained on a vanilla binary BoW. With a tuned learning rate of .85, this model achieved 84.4% accuracy after 50 epochs. We subsequently experimented with sanitizing the text, which we found to be beneficial in both the BoW featurization and the n-gram featurization. We hypothesized this is a result of decreasing the variance in the many forms of the same word found in the emails, allowing for better feature matching across documents. While trigrams with sanitized text performed best on the development dataset, the BoW with sanitized text ultimately performed best on the test dataset, with an accuracy of 75.7%. A plausible explanation for this result is that longer n-grams are more prone to overfitting, as they're less likely to occur in a document.

For the MLP model, we went through similar experiments and chose to use the similar model with the ReLU activation function with 2 hidden layers and 32 features. Overall the full model gave 85.71% accuracy over the test dataset.

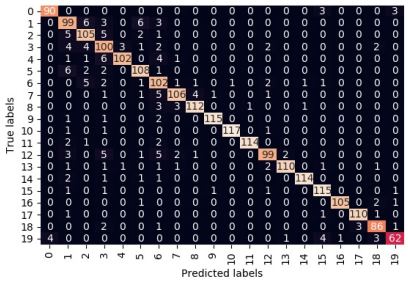

Figure 1: Newsgroup Confusion Matrix

According to the confusion matrix, the model tends to have errors with the following categories: religions, electronics and hardware. One example of the common miss-classification is that the model often treat emails belongs to soc.religion.christian to talk.religion.misc.

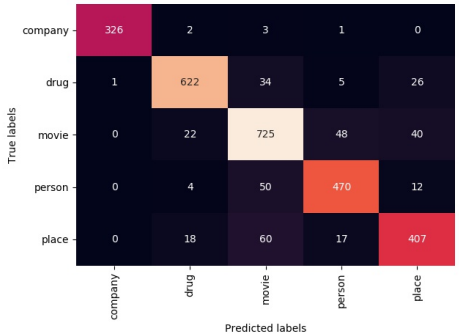

Figure 2: Propername Confusion Matrix

Batch Size	Average Speed Per Sample
Propername Dataset	
1	23.00ms (SD = 4.83)
25	1.40ms (SD = 0.28)
100	1.02ms (SD = 0.17)
1000	0.96ms (SD = 0.10)
Newsgroup Dataset	
1	56.00ms (SD = 22.71)
25	5.36ms (SD = 1.20)
100	3.37ms (SD = 2.90)
1000	3.26ms (SD = 0.28)

Table 3: Batch Benchmarking Results

4.3 Batching Benchmarking (4pt)

We calculated the average training speed per sample for both dataset, with batch size equals to 1, 25, 100, 1000. In general, the MLP model spent about 1ms training a propername sample and 3ms training a newsgroup sample. From the results on Table 3, we found that the average speed decreased and stabilized when batch size increased.

5 Conclusion (4pt)

In conclusion, we found that the best performing model is the MLP with character N-grams for propernames with an accuracy of 85.7% and the MLP with BoW for the newsgroup with an accuracy of 81.7%. The result from MLP was better than perceptron on both tasks, since MLP had more hidden layers and more complex structure, thus making it more capable to approximate the true decision boundary in the data distribution.

We also learnt that it was valuable to have our decisions empirically evaluated. We realized simplified preprocessing and model architectures for small dataset greatly increased the accuracy. Simple featurization such as n-gram character and BoW performed better than more complicated techniques.