HOMEWORK 9: NLP TASK REPORT

Luyi Ma(luyim), Yilan Xu(yilanx2)

Problem Statement

The NLP task is to identify sentence chunks and tag them using IOB tags given the original sentences and corresponding pos(part-of-speech) tags. As pos tag itself overlooks the information embedded in the content of the consecutive words, IOB tagging is a more precise way to interpret the structure of the sentences.The train set is composed of sections from WSJ(Wall Street Journal), and the test set is a smaller portion of the WSJ. Our goal is to design and train a model to do IOB tag prediction accurately. The performance of the model is measured by F1 score, a combination of tagging accuracy and recall rate.

Data Pre-processing and Feature Engineering

The train data consists three columns separated by tabs: token, pos-tag and IOB-tag. The tokens are arranged in the order of the original chunks and there are empty lines presented in the middle of the dataset. Data Information:

	Train	Test
token counts	157850	53919
unique tokens	16449	9238

num of unique IOB tag	num of unique pos-tag
24	45

Pre-processing: Our proposal is to use a CNN-like model, so the most obvious task in data pre-processing is to covert the words into a numerical representation. We considered mainly two choices. The first one is using one hot vector. One-hot encoding treats each distinct word as a feature, so it's good for later feature extraction. However, this scheme will effectively creates a (157850, 16449) matrix as input for training set and words with similar meaning and type will be treated as distinct words, and under the 32-bit operating system, the total memory size required would exceed 8 GB RAM easily. An alternative solution is to use Glove. Glove encoding words with information from the cooccurrence rate, and the vector size is customizable. So we made decisions to use Glove to encode the word tokens (with vector size = 1000) and Simple One-hot encoding for pos-tag. Since the ylabel should also be represented numerically, we created a BIO-tag vocabulary, and represent each tag by its corresponding index in the vocabulary vector. As a result, a train data set with n words will be covert to a (n, 1000+45) matrix input and a (n, 1) vector of ylabel.

Feature Engineering:
Typically, the input to the CNN model should be images. In the case of NLP, we define an image as a phrase or a short sentence. First we vectorize words and POS tags to generate lookup table. Then, based on the lookup table, we go through a dataset and produce an image for each word. For example, an ?image? is built by stacking N word-POS vectors (word GloVe vector – POS one-hot vector) before the central word, the central word?s word-POS vectors and N word-POS vectors after the central word. So the height of an image is (2N + 1). Based on this data representation, we classify a word as a image of (2N+1)gram. IOB tags predicted for each image are the tags for the central words.

Model Analysis

Image size tunning:
we tried many image sizes and finally chose 5-word window to produce input images. The reasoning behind the number is that we want to balance between the information of the surrounding words and the noises introduced by them. When classifying some rare IOB tags, we found the accuracy is generally low due to the existence of popular IOB tags around the word token.

CNN structures(3): In the first CNN structure, the input data is passed to a convolution matrix with the same weight of the input data matrix but a height of (M / 2 + 1). If the input image size is M x W where M is the number of words in the input window and W is the length of the word vector. With zero padding and 1 stride size, the dimension of output image is (M / 2 + 1). The reason for this window size is that the central word vector is used for all (M / 2 + 1) convolution operations but the number of convolution operations that other words are involved is less than (M / 2 + 1) and determined by their

distance to the central word. So the assumption is that the impact of words on the central word classification decreases from the central word to both ends. In this CNN structure, we simply use one convolution layer, an IP layer, an ReLu layer and finally a softmax layer. The test accuracy is around 92%
In the second CNN structure, based on the assumption that the impacts of surrounding words do not decrease and information from different words can be combined to be meaningful, smaller window size are chosen . The convolution matrix is set to (2 x 2) and two convolution layers and two max pooling layers are used. Test results indicate that the test accuracy is still around 93% even with a more complex CNN structure. Since GloVe vectors and POS one-hot vectors are from different generative models, we believes we cannot simply use a convolution matrix to merge them together. Then we try to change the weight of convolution matrix to the greatest common divisor G of the size of word GloVe vector and POS one-hot vector. This time, with zero padding and G stride size, we only extract information in each sub-vector. Following the same LeNet structure, we obtain a test accuracy about 93%.
In the third CNN structure, the input data pass a convolution matrix with convolution matrices of different sizes in the same layer. The assumption used here is the input words may have some substructures. For example, if an input contains 5 words, maybe each continuous 4 words, 3 words or even 2 words form a phrase that is useful for IOB tag classification. So we design following convolution neural network to extract information from input images. In Figure 1 , the input image is a 5 x 90 matrix (45 word token + 45 pos-tag). The input image is parallel processed by different convolution matrices. The output of each convolution operation is sent to a max pooling layer and all max pooling outcomes will be stacked to become a column vector to be the input of the following inner product layers (not shown in the figure). The test accuracy of this CNN architecture after 10000 iterations with batch size 64 is about 94%. But we think the parameter tuning can be more subtle. So we tried predicting an extra test set, and compute the F score on the prediction. We found that IOB tags like NP, PP are predicted with high accuracy while other tags like PRT and ADJP are predicted badly. So we rebuild our CNN structure and add more convolution matrices. Finally, our model has the structure shown in the Figure 2. We want to extract information in different levels. The convolution matrix in light yellow parses the whole image. The reason we add this filter is because it can extract the central word better with consideration of its prefix and suffix. We also add a smaller convolution matrix(dark red) to extract the each word information. Finally, we train this model with 20000 iterations and the test accuracy returned in training time is 95.2%. The F1 score is 88.2 for our test data set.
For the data set of this homework, our model only achieve a F1 score of 58.1. The reason for this decrease is unclear. Finally we use prediction made by a mix of bigram, unigram and trigram tagger from NLTK package for this homework.

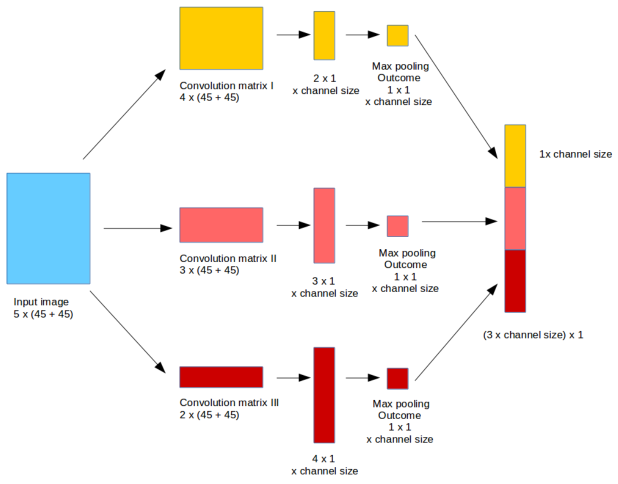


Figure 1: Initial CNN model with assumption 3

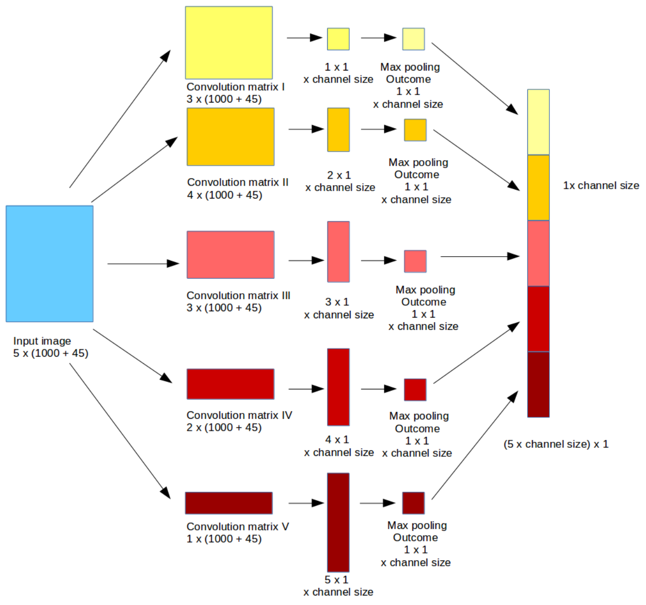


Figure 2: Final CNN model

Results Analysis

Since we are not given the correct labels for the test set, we used 4-fold cross validation embedded in the LeNet training code to evaluate the model. We also downloaded an extra test set from the cornll200 official website (http://www.cnts.ua.ac.be/conll2000/chunking/test.txt.gz) to tune the final model. For the final model, we used the extra dataset to calculate the F1 score. We first used 8 convolution filters(height = 4,4,3,3,2,2,1,1) for this model, and tried increasing the word vector size to classify words better. Results shows that, when the vector size is 1000, the F1 score peaks.

Although the improvements are subtle, the test results suggest that using word vector with size 1000 gives the best prediction. [See Figure 1, 2, 3, in appendix for score details]

vector size	45	500	1000
test accuracy	92.60%	92.75%	93.08%
F1 score	88.08	87.97	88.14

Maintaining the same word vector, we tried other two models:
1. 8 convolution filters: (height = 5, 5, 4, 4, 3, 3, 2, 1) [See Figure 4 in appendix for score details]
2. 10 convolution filters: (height = 5, 5, 4, 4, 3, 3, 2, 2, 1, 1) [See Figure 5 in appendix for score details]

model	1	2
test accuracy	93.09%	93.04%
F1 score	88.17	88.20

The F1 score slightly increased with these above models, and there are also a relatively major improvement in prediction of rare IOB tags. However, even with complex model, we still couldn't make significant progress on the F1 score, so it seems that 93% accuracy is the limit of our CNN model.
We here provide some possible explanations for this issue:
1: The length of a sentence varies. Arbitrarily determine the fixed window size is less flexible when parsing the input article and extract the correct segments for IOB tags.
2: The length of phrase varies. So it is not good idea to fix the size of convolution matrix.
3: There is a tradeoff between increase the average F score and enhance the sensitivity of extracting rare IOB tags. So it needs a more flexible model to make a decision based on prefix and suffix with different length.
4: We think CNN is good at extracting information following some patterns. Our model can still have 95% test accuracy and 93% prediction accuracy because most of sentences share similar patterns. Also we think CNN is good at recognizing patterns like whether a sentence is inverted or not. We can use CNN to extract pattern information and use this information for further classification. So we think the assumption of patterns for segmentation is not strong enough so that more flexible model should be used to capture information in other dimension (like RNN).

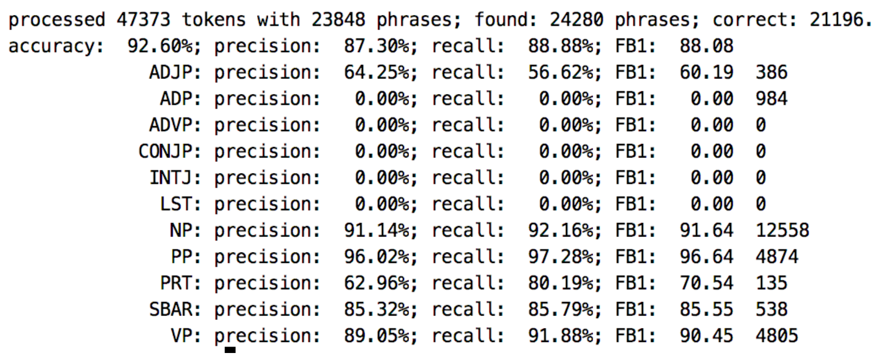
Collaboration

We worked in a group of 2. Both of us designed the model structure and analyses test results. Luyi mainly worked on the model construction and hyperparameters tuning, Yilan worked more on the data-preprocessing part.
We neither offer help, nor receive any help from other groups.

Time Spent

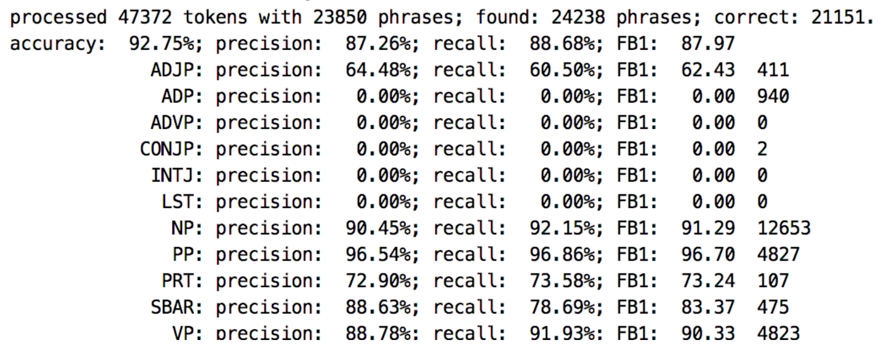
Time Spent is about 40 hours for each of us.

Appendix: F1 score details



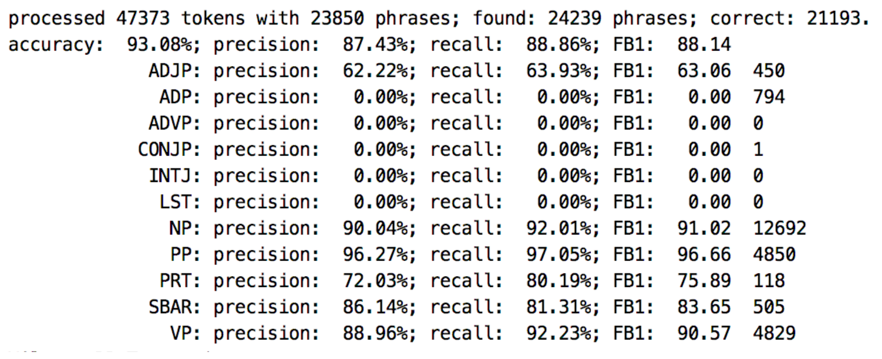
```
processed 47373 tokens with 23848 phrases; found: 24280 phrases; correct: 21196.
accuracy:  92.60%; precision:  87.30%; recall:  88.88%; FB1:  88.08
             ADJP: precision:  64.25%; recall:  56.62%; FB1:  60.19  386
              ADP: precision:   0.00%; recall:   0.00%; FB1:   0.00  984
             ADVP: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
            CONJP: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
             INTJ: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
              LST: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
               NP: precision:  91.14%; recall:  92.16%; FB1:  91.64  12558
               PP: precision:  96.02%; recall:  97.28%; FB1:  96.64  4874
              PRT: precision:  62.96%; recall:  80.19%; FB1:  70.54  135
             SBAR: precision:  85.32%; recall:  85.79%; FB1:  85.55  538
               VP: precision:  89.05%; recall:  91.88%; FB1:  90.45  4805
```

Figure 3: word vector size = 45



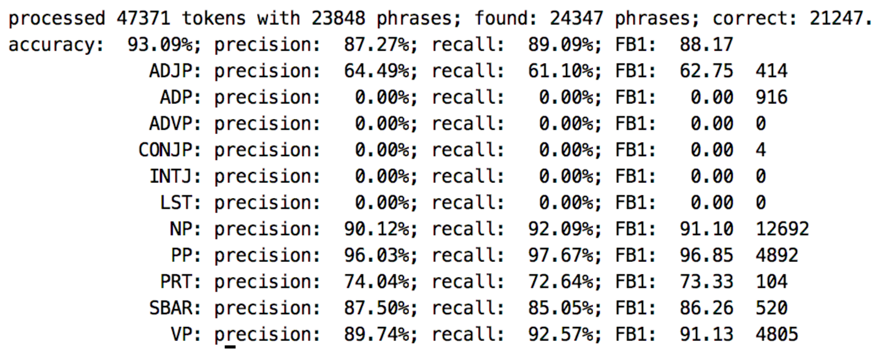
```
processed 47372 tokens with 23850 phrases; found: 24238 phrases; correct: 21151.
accuracy:  92.75%; precision:  87.26%; recall:  88.68%; FB1:  87.97
             ADJP: precision:  64.48%; recall:  60.50%; FB1:  62.43  411
              ADP: precision:   0.00%; recall:   0.00%; FB1:   0.00  940
             ADVP: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
            CONJP: precision:   0.00%; recall:   0.00%; FB1:   0.00  2
             INTJ: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
              LST: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
               NP: precision:  90.45%; recall:  92.15%; FB1:  91.29  12653
               PP: precision:  96.54%; recall:  96.86%; FB1:  96.70  4827
              PRT: precision:  72.90%; recall:  73.58%; FB1:  73.24  107
             SBAR: precision:  88.63%; recall:  78.69%; FB1:  83.37  475
               VP: precision:  88.78%; recall:  91.93%; FB1:  90.33  4823
```

Figure 4: word vector size = 500



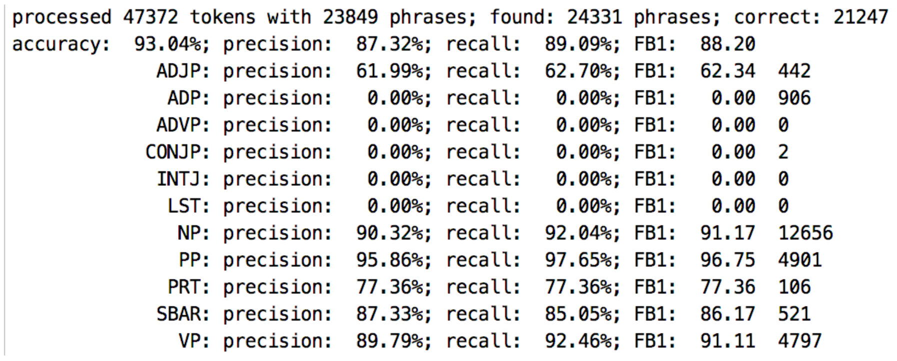
```
processed 47373 tokens with 23850 phrases; found: 24239 phrases; correct: 21193.
accuracy:  93.08%; precision:  87.43%; recall:  88.86%; FB1:  88.14
             ADJP: precision:  62.22%; recall:  63.93%; FB1:  63.06  450
              ADP: precision:   0.00%; recall:   0.00%; FB1:   0.00  794
             ADVP: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
            CONJP: precision:   0.00%; recall:   0.00%; FB1:   0.00  1
             INTJ: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
              LST: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
               NP: precision:  90.04%; recall:  92.01%; FB1:  91.02  12692
               PP: precision:  96.27%; recall:  97.05%; FB1:  96.66  4850
              PRT: precision:  72.03%; recall:  80.19%; FB1:  75.89  118
             SBAR: precision:  86.14%; recall:  81.31%; FB1:  83.65  505
               VP: precision:  88.96%; recall:  92.23%; FB1:  90.57  4829
```

Figure 5: Iword vector size = 1000



```
processed 47371 tokens with 23848 phrases; found: 24347 phrases; correct: 21247.
accuracy:  93.09%; precision:  87.27%; recall:  89.09%; FB1:  88.17
             ADJP: precision:  64.49%; recall:  61.10%; FB1:  62.75  414
              ADP: precision:   0.00%; recall:   0.00%; FB1:   0.00  916
             ADVP: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
            CONJP: precision:   0.00%; recall:   0.00%; FB1:   0.00  4
             INTJ: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
              LST: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
               NP: precision:  90.12%; recall:  92.09%; FB1:  91.10  12692
               PP: precision:  96.03%; recall:  97.67%; FB1:  96.85  4892
              PRT: precision:  74.04%; recall:  72.64%; FB1:  73.33  104
             SBAR: precision:  87.50%; recall:  85.05%; FB1:  86.26  520
               VP: precision:  89.74%; recall:  92.57%; FB1:  91.13  4805
```

Figure 6: filter:(5,5,4,4,3,3,2,1)



```
processed 47372 tokens with 23849 phrases; found: 24331 phrases; correct: 21247
accuracy:  93.04%; precision:  87.32%; recall:  89.09%; FB1:  88.20
             ADJP: precision:  61.99%; recall:  62.70%; FB1:  62.34  442
              ADP: precision:   0.00%; recall:   0.00%; FB1:   0.00  906
             ADVP: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
            CONJP: precision:   0.00%; recall:   0.00%; FB1:   0.00  2
             INTJ: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
              LST: precision:   0.00%; recall:   0.00%; FB1:   0.00  0
               NP: precision:  90.32%; recall:  92.04%; FB1:  91.17  12656
               PP: precision:  95.86%; recall:  97.65%; FB1:  96.75  4901
              PRT: precision:  77.36%; recall:  77.36%; FB1:  77.36  106
             SBAR: precision:  87.33%; recall:  85.05%; FB1:  86.17  521
               VP: precision:  89.79%; recall:  92.46%; FB1:  91.11  4797
```

Figure 7: filer:(5,5,4,4,3,3,2,2,1,1)