EDAN20 Assignment 4
Chunker

Matilda Andersson, ma4748an-s

October 2020

1 Introduction

The purpose of this laboratory report is to describe and comment on the machine-learning program composed in this assignment. The program was written and trained to detect detect partial syntactic structures, noun groups, and used the Python library scikit-learn's linear regression for training and classifying. The data used for training and validating the model was the same data used for a task during the Conference on Computational Natural Language Learning in 2000 (CoNLL-2000).

2 Process description

The following subsections contain a more in depth description of the process for creating the machine-learning program.

2.1 Preparing the data

In order to create a program that could detect different groups of nouns, and divide a text into chunks, we used the annotated data and annotation scheme available from CoNLL 2000. The data consisted of partitions of the Wall Street Journal corpus, was made up of totally 259104 tokens and had been annotated and split into a three column structure of the current word, its part-of-speech tag and a chunk tag.

2.2 Baseline chunker

Applying a minimal technique for predicting the noun group, we were able to receive a baseline prediction of the tasks difficulty, which later on in the process could be used for comparing and evaluating the accuracy and performance of further programs and techniques. The baseline measurement were obtained by implementing the method proposed by the organizers of the CoNLL 2000 shared task, which constituted of selecting the chunk tag which was most frequently associated with the current part-of-speech tag. A method that, in my opinion, skews the results as it does not make an reasoned prediction and implements a bias towards the most frequent tags. The baseline result of 0.7729066846782194 only tells us that 77% of the time, the most frequent tag combinations can be correctly applied.

2.3 The first ML approach

In the original method implemented by Kudoh and Matsumoto trained their classifier based on support vector machines while we, in this assignment use logistic regression. Furthermore, in this first program we trained our model using only 2/3rds of the elements that made up the feature vector used by Kudoh and Matsumoto. Omitting the dynamic parameters, we were left with a feature vector consisting of:

- The values of the five words in this window: $w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$
- The values of the five parts of speech in this window: $t_{i-2}, t_{i-1}, t_i, t_{i+1}, t_{i+2}$

Running our training data through the auxiliary functions we obtained, for each word, a dictionary consisting of the aforementioned features. These feature dictionaries were vectorized with one-hot encoding and then used for training our model using the linear regression function provided by the sklearn library. Moving on to processing and preparing the test data, we extracted the feature vectors of the test data which, were run through the trained model in order to obtain a predicted chunk for each word. Evaluation the predictions gave us a score of 0.9157688948047087.

2.4 ML approach using dynamic features vectors

In order to reach the goal of this task and coming forward with a machine learning method which after the training phase could, as well as possible, recognize the chunk segmentation of a dataset, we needed to iterate on the above outlined process complementing the feature vectors with an additional dynamic parameter. These dynamic parameters consisted of a predicted chunk which were computed on run time using the two previously predicted chunk tags. Firstly, we had to complement the feature vector we used in the previous section with the two dynamic features, c_{i-2}, c_{i-1}, in order to train a new model capable of making predictions using this augmented feature vector. After obtaining the full feature vectors for the training data we moved on to training our new model, also this time using Linear Regression. Finally, loading the test corpus once again we were ready to make predictions using this new dynamic approach. Using the code seen in figure 1, we were able to word by word go through the sentences and predict the chunks corresponding to that word. This prediction were then used as a basis for succeeding words and their chunk predictions. This approach, together with the default parameters from sklearn, resulted in a F1 score of 0.9232606372949532.

```
y_test_predicted_dyn = []
```

```
from tqdm import tqdm
i=0
for sentence in tqdm(test_corpus):
    sentence_features, y_test = extract_features_static([sentence], w_size, feature_names)
    for word in sentence_features:
        if word['word_n2']=='bos' and word['word_n1']=='bos':
            word['chunk_n2'] = 'BOS'
            word['chunk_n1'] = 'BOS'
        elif word['word_n2']=='bos' and not word['word_n1']=='bos':
            word['chunk_n2'] = 'BOS'
            word['chunk_n1']= y_test_predicted_dyn[i-1]
        else:
            word['chunk_n2'] = y_test_predicted_dyn[i-2]
            word['chunk_n1'] = y_test_predicted_dyn[i-1]
        predicted = classifier2.predict(vec2.transform(word))
        y_test_predicted_dyn.append(predicted[0])
        i+=1
```

```
100%|██████████| 2012/2012 [00:16<00:00, 125.66it/s]
```

Figure 1: The code used for prediction using a dynamic feature vector.

3 Additional task

Reading the article by Akbik et al. referred to in the assignment, I find, to begin with, that the model proposed by the researchers looks at character-level data, and then combines this technique with sequencing of words, which is the approach used in our lab. Using neural networks to process the character data allows for long-term dependencies between words, unlike e.g. n-grams. This model allows for a different type of process for individual word identification than hard coded approaches such as spacing, and instead uses the probability of a word given its context. The model used by the researchers achieves a result of 96.72, a significant improvement to Kudoh and Matsumoto's 93.48.