SAKI SS19 Homework 2

Author: Daniel Seitz

Program code: https://github.com/Anto4ka/Resume_NER

Summary

For a given set of resumes, the challenge is to develop a NER (=”Named Entity Recognition”) model, that is able to extract named entities (labeled individual data) of the categories “Skills”, “Degree” and “Location”.

The chosen approach is a NER-model using the NLP/NER-library “Flair”.

Every raw data entry is a dictionary, consisting of the keys “content” (the actual resume text) and “annotation”. The latter is a list of dictionaries with the keys “label” (describing the label of a named entity), “points” (describing the labeled content and its starting -/ endpoint) and others, which are irrelevant for our model (e.g “metadata”).

This obvious redundancy of information motivated the conversion and cleaning of the raw data.

In the first step the irrelevant key/pairs were removed and the redundant format was improved. A single entry is now a tuple, consisting of two elements: the text (datatype: string) and a dict, containing the “annotation” information. While the text stayed the same, the “annotation” dict now only has one key “entities”, which contains a list of tuples (each with three elements). Instead of having an extra key to represent the label, each tuple consists of a starting point, an end point and the corresponding label. The resumes missing labels were also removed.

In the second step the NER-library “spaCy” was used to perform some additional preprocessing on the data. Besides removing bad data, all the resumes that do not contain at least one instance of an entity from the categories “Skills”, “Degree” or “Location” were left out. The data was also split into two representative subsets for training and testing of the future model.

The most important step in the preprocessing progress was the conversion of the data to the “BILOU”-format, which is used to map resumes to (“Pandas”-)tables consisting of a “Token”- and a “BILOU-TAG” - column. The “BILOU”-scheme is a more detailed version of the “BIO”-scheme, which encodes the Beginning, the Inside and the last token of multi-token chunks.

At last, the resulting tables were cleaned from faulty/irrelevant rows, and sentence segmentation was performed. The sentences were created very intuitively, using full stops as seperating marks.

The final, clean data set is a table, consisting of the columns “text” (the Token), “predicted” (results of an earlier trained Spacey-NER-model, which are completely irrelevant for the following model) and “ner” (which represents the actual label of a Token).

For the final NER-model, a NLP-framework called “flair” was used. The main decisions that had to be made, were the choice of batch size, embeddings and amount of epochs.

While the batch size had to be set rather low, as the used cloud computing service “Google Colab” restricted the possible batch size through its offer of RAM, the embeddings presented some freedom of choice.

It was decided to use “flair” embeddings, which are characterized by the fact that their usage results in a very long training duration. But proportional to the immense computing power needed, the results can be astonishing.

The amount of epochs, the model would train for, was chosen to be pretty big (150 epochs). This was a rather unrefined but effective approach, since it made possible to pinpoint the moment, when the model converged, and the minimization of the loss function halted.

Evaluation.

The metric of choice was the balanced F-Score, also known as the F1-Score.

As one of the most frequently used metrics, the F1-Score describes the weighted average of the Precision and Recall.

$$Precision = \frac{True\ Positives}{(True\ Positives + False\ Positives)} \qquad Recall = \frac{True\ Positives}{(True\ Positives + False\ Negatives)}$$

Which results in the following formula:

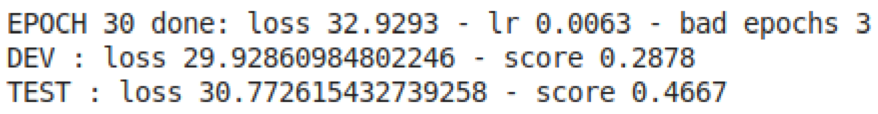
$$F1 = \left(\frac{Precision^{-1} + Recall^{-1}}{2}\right)^{-1}$$

Even though this metric is not as intuitive as the raw ratio between correct and incorrect classifications (Accuracy), it offers great benefits by taking both false positives and false negatives into account.

Now this definition is only applicable to binary problems. To scale this metric to a multiclass problem, we need to chose a strategy to calculate a scalar value out of the F1 scores of each class. An example would be to calculate the metric for each label, and to find their unweighted mean. The problem with this approach is, that it does not take label imbalance into account. Since our training data might be highly imbalanced, this would be less than optimal.

A more appropriate approach is to again calculate the F1 score for every label and to weigh the scores with their corresponding label-frequency. This is called a “weighted macro F1-Score”.

After a training phase of ~12 hours, the model converged to a weighted macro F1- Score of 0.4667.



```
EPOCH 30 done: loss 32.9293 - lr 0.0063 - bad epochs 3
DEV : loss 29.92860984802246 - score 0.2878
TEST : loss 30.772615432739258 - score 0.4667
```

Naturally, this result is far from astonishing, and definitely does not reflect the sheer power and possibilities of the flair-framework.

Possible Improvements

While much effort was put into the preprocessing of the raw data, there were a couple points in the pipeline that could profit from a more sophisticated approach. An obvious example would be the sentence segmentation. Instead of just using the rather primitive method of seperation by full stop, one could have added other characters as seperation marks, to at least improve this simple rule based system.

One could also implement highly-performant, trainable sentence sequencing algorithms, e.g a statistical maximum entropy model (Reynar, J., Ratnaparkhi, A.: A maximum entropy approach to identifying sentence boundaries, 1997).