Lund University
Faculty of Engineering (LTH)

EDAN95

Applied Machine Learning

Project 4

Author:
Marcel Attar, 941127-2173

The report was handed in on: December 5, 2019

1 Building a Simple Recurrent Neural Network

Before start building the network building some preprocessing of the data needed to be done. This was done using the provided instructions in the lab description (found *here* [2]) and the *GitHub* [3] link provided.

After the preprocessing a network with the following architecture was created

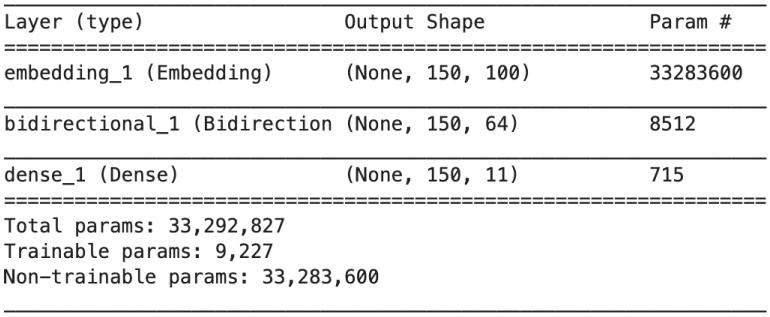

Figure 1: The network architecture of the first RNN model.

This architecture resulted in the following results after just 3 epochs

```
processed 46666 tokens with 5648 phrases; found: 5123 phrases; correct: 3626.
accuracy:  93.05%; precision:  70.78%; recall:  64.20%; FB1:  67.33
              LOC: precision:  76.88%; recall:  72.18%; FB1:  74.46  1566
             MISC: precision:  65.77%; recall:  45.44%; FB1:  53.75  485
              ORG: precision:  60.41%; recall:  56.23%; FB1:  58.25  1546
              PER: precision:  76.61%; recall:  72.29%; FB1:  74.39  1526
```

Figure 2: Using a simple recurrent neural network gives an accuracy of 93.05%. The best performing entities are Location and Person.

2 Building a LSTM Network

Now a LSTM network of this architecture was used

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 150, 100)          40259700
_________________________________________________________________
bidirectional_2 (Bidirection (None, 150, 200)          160800
_________________________________________________________________
bidirectional_3 (Bidirection (None, 150, 200)          60200
_________________________________________________________________
dense_2 (Dense)              (None, 150, 200)          40200
_________________________________________________________________
dropout_1 (Dropout)          (None, 150, 200)          0
_________________________________________________________________
dense_3 (Dense)              (None, 150, 11)           2211
=================================================================
Total params: 40,523,111
Trainable params: 263,411
Non-trainable params: 40,259,700
_________________________________________________________________
```

Figure 3: The architecture of an LSTM network with 2 bidirectional layers.

This network took substantially longer time to train compared to the previous one, which is expected since a LSTM is more complex. The network yielded the following results after 15 epochs

```
processed 46666 tokens with 5648 phrases; found: 5673 phrases; correct: 4657.
accuracy:  96.30%; precision:  82.09%; recall:  82.45%; FB1:  82.27
              LOC: precision:  87.58%; recall:  86.27%; FB1:  86.92  1643
             MISC: precision:  69.23%; recall:  65.38%; FB1:  67.25  663
              ORG: precision:  73.96%; recall:  78.51%; FB1:  76.17  1763
              PER: precision:  90.71%; recall:  89.98%; FB1:  90.34  1604
```

Figure 4: Using the LSTM network an accuracy of 96.30% was achieved, which can be compared to 93.05% from the previous network. The overall F1-score was 82.27.

Our LSTM network performed better than the simple recurrent neural network, 3.25 percentage points higher accuracy. Also, the desired minimum F1-score of over 82 was achieved.

3 Article

The article *Contextual String Embeddings for Sequence Labeling* [1] by Akbik et al. examines a new approach to word embedding that the authors call contextual string embeddings. This new approach produces results that beat the state-of-the-art, it performs especially well on Named Entity Recognition (NER) tasks. The F1-score for the CoNLL03 NER task increase by almost 10 pp in German and 1 pp in English, a significant increase.

3.1 Contextual string embeddings

This new approach to contextualized characterlevel word embedding have three nice attributes: (1) the ability to be able to pre-train the model on an unlabel corpora, (2) it captures word meaning in context, therefore it is able to produce different embeddings for a polysemous word depending on it usage, and (3) model words and context fundamentally as sequences of characters, which the authors explain gives the model the ability to handle misspelled word as well as model prefixes and suffixes better than other models.

This network feeds in the sentence as a character sequence in a pre-trained character language model, which outputs a contextual word embedding that is then feed into a sequence label model, see figure 5. The approach used by us differed from this, we used the GloVe set which has a built in embedding that is based on the entire word, therefore we do not get the benefit of context, i.e. a word has a single embedding no matter if it is polysemous or not. For example, the sentences *The computer's mouse was broken* and *The mouse was hiding in the attic* both have the word mouse in them but with completely different meanings. Most likely the model that the article uses would perform better on this sentence compared to the one we used.

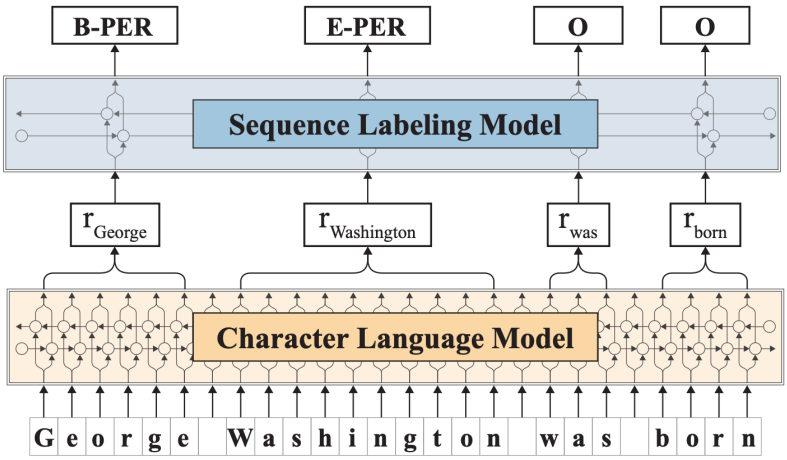

Figure 5: A high level illustration of the model used in the article. In this case the model was used to perform the NER task. Figure was taken from the article [1].

References

[1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL: https://www.aclweb.org/anthology/C18-1139.

[2] Pierre Nugues. Lab session 4. URL: http://cs.lth.se/edan95/lab-programming-assignments/lab-session-4/.

[3] Pierre Nugues. pnugues: Github. URL: https://github.com/pnugues/edan95/blob/master/programs/5.2-monoinput.ipynb.