**HW3 Report**

Emanuele Alessi 1486470

**Preface**

In this homework I have implemented a Semantic Role Labeling model with the aim of identifying and labeling predicates and roles of the arguments given in the CoNLL 2009 dataset.

**1. Mandatory task**

**1.1. Role classifier**

The architecture of the neural network is represented in three layers: the first layer contains the concatenation between the Glove embeddings (100 dimesions, not trainable), the part-of-speech tag embeddings (50 dimension, trainable) and the 0/1 value which represents the predicate flag (if the current word of the sentence has not an associated predicate then the value is 0; 1 otherwhise); the second layer and the third layer contains respectively a bidirectional LSTM and a softmax classifier which gives in output the labels (roles represented as integer values) with the maximum score.

The main algorithm is the following:

1. Parse the CoNLL 2009 dataset in order to convert the lemmas, roles and part-of-speech tags of the sentences into unique integer values and save them in a list;
2. Divide the list into batches;
3. Train the LSTM passing the batches (lemmas + part-of-speech tags + flags) and the labels (roles)
4. Save the model to continue training if necessary

I also implemented a technique to reduce the class imbalance and obtain better results: I pass to the LSTM a vector (1 dimension) of coefficients which multiplies the masked losses; the vector contains 1 or 0.2 values (1 for the lemmas that have an associated predicate; 0.2 otherwhise).

**1.2. Results**

To train the LSTM the BiLSTM I choose the following parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| **Batch size** | **BiLSTM hidden size** | **Optimizer** | **Learning rate** |
| 10 | 128 | Adam | 0.001 |

After the training phase, I have obtained with 10 epochs the following scores on the CoNLL 2009 development set:

|  |  |  |  |
| --- | --- | --- | --- |
| **F1 (Macro)** | **Precision** | **Recall** | **Accuracy** |
| 76,8% | 67,7% | 88,8% | 96,6% |

For this task the reference files are *role\_classifier.py*, *data\_preprocessing.py* and *evaluation.py*.

**2 Extension 1.2 (predicate identification and disambiguation)**

**2.1. HW2 system**

For the **Extension 1.2** I decided to use my HW2 system (a.k.a. “WordSenseDisambiguator”,located in the “WSD” folder) to implement the word sense disambiguation.

Because of the slightly low scores achieved in the previous delivery, I wanted to modify it in order to make it more performant, so I fixed many bugs and functions and I added many tricks such as class imbalance reducing technique (explained in point 1.1).

The neural architecture (BiLSTM) and the hyper parameters are the same of the last delivery.

**2.2. HW2 system results**

For the training phase, the chosen hyper parameters are:

|  |  |  |  |
| --- | --- | --- | --- |
| **Batch size** | **BiLSTM hidden size** | **Optimizer** | **Learning rate** |
| 10 | 100 | Adam | 0.001 |

After 15 epoch of train on Semcor dataset I achieved the following scores:

|  |  |  |
| --- | --- | --- |
|  | **F1** | **Accuracy** |
| **Senseval2** | 69,2% | 81,4% |
| **Senseval3** | 70,5% | 84,8% |
| **Semeval2007** | 66,2% | 92,8% |
| **Semeval2013** | 63,0% | 89,4% |
| **Semeval2015** | 64,6% | 79,6% |

**2.3. BabelNet to PropBank alignment creation**

After the training phase, I created the association (one-to-one) between the synsets of BabelNet and the predicates of PropBank.

The idea (pseudocode) of how I implemented the alignment is the following:

*wsd = WordSenseDisambiguator() # HW2 system instance*

*d = {} # keys are synsets and values are predicates lists*

*for each sentence in CoNLL 2009 do*

*predictions = wsd.predict(lemmas of sentence) # a list of predictions (BabelNet synsets)*

*for i = 1 to predictions length do*

*if predictions[i] ≠ ‘UNK’ and predicate of sentence[i] ≠ ‘\_’ then*

*d[predictions[i]].append(predicate of sentence[i])*

*for each pair (synset, predicates) in d:*

*write a line in babelnet2propbank.txt which contains the synset and the most common predicate in predicates*

For this task the reference files are all the code located in WSD folder, *babelnet2propbank.py* and *pos\_map.py* (this one is used to map the part-of-speech tags of Propbank to the part-of-speech tags of Semcor).

**2.4. Predicate classifier**

Similarly to the neural architecture of the role classifier is represented in three layers: the first layer contains the concatenation between the Glove embeddings (100 dimesions, not trainable) and the part-of-speech tag embeddings (50 dimension, trainable); the second layer contains the bidirectional LSTM and the third layer contains a softmax classifier which gives in output the labels (predicates represented as integer values) with the maximum score.

First of all, In this task I tried two different approaches in order to compare them and choose the most performant one.

In each approach I used the same hyper parameters and training techniques. I have also implemented the class imbalance reducing technique in this task (see point 1.1)

* The first approach is to train the predicate classifier passing as input the batches and the labels which are contained in CoNLL 2009 and Semcor datasets, exploiting *babelnet2propbank.txt* to map the synsets of BabelNet with the predicates of PropBank;
* The second approach is to train the predicate classifier passing as input only the batches and the labels which are contained in CoNLL 2009 dataset

**2.5. Results**

I noticed that using only CoNLL 2009 dataset as trainset for the predicate classification task is the best choice, so I decided to train the predicate classifier using the second approach.

For the training phase I choose the following parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| **Batch size** | **BiLSTM hidden size** | **Optimizer** | **Learning rate** |
| 10 | 100 | Adam | 0.001 |

After 15 training epochs I achieved the following results:

|  |  |  |  |
| --- | --- | --- | --- |
| **F1 (Macro)** | **Precision** | **Recall** | **Accuracy** |
| 88,6% | 85,3% | 92,1% | 95,7% |

For this task the reference file is *predicate\_classifier.py*.

The next pages contain the references to the papers that inspired me, the neural architecture picture and the confusion matrix for the mandatory task.

NOTE: due to time constraints, I could not load backups of LSTM models on gitlab, if you are interested in them, please send me an email.

**References**

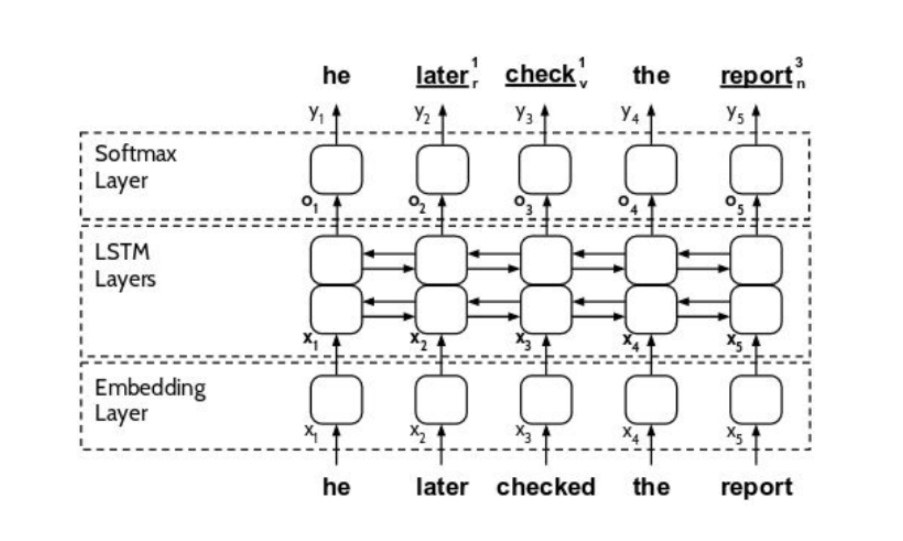
Roth M., Lapata M. - Neural semantic role labeling with dependency path embeddings

Hajic J. et al – The CoNLL 2009 shared task: syntactic and semantic dependencies in multiple languages

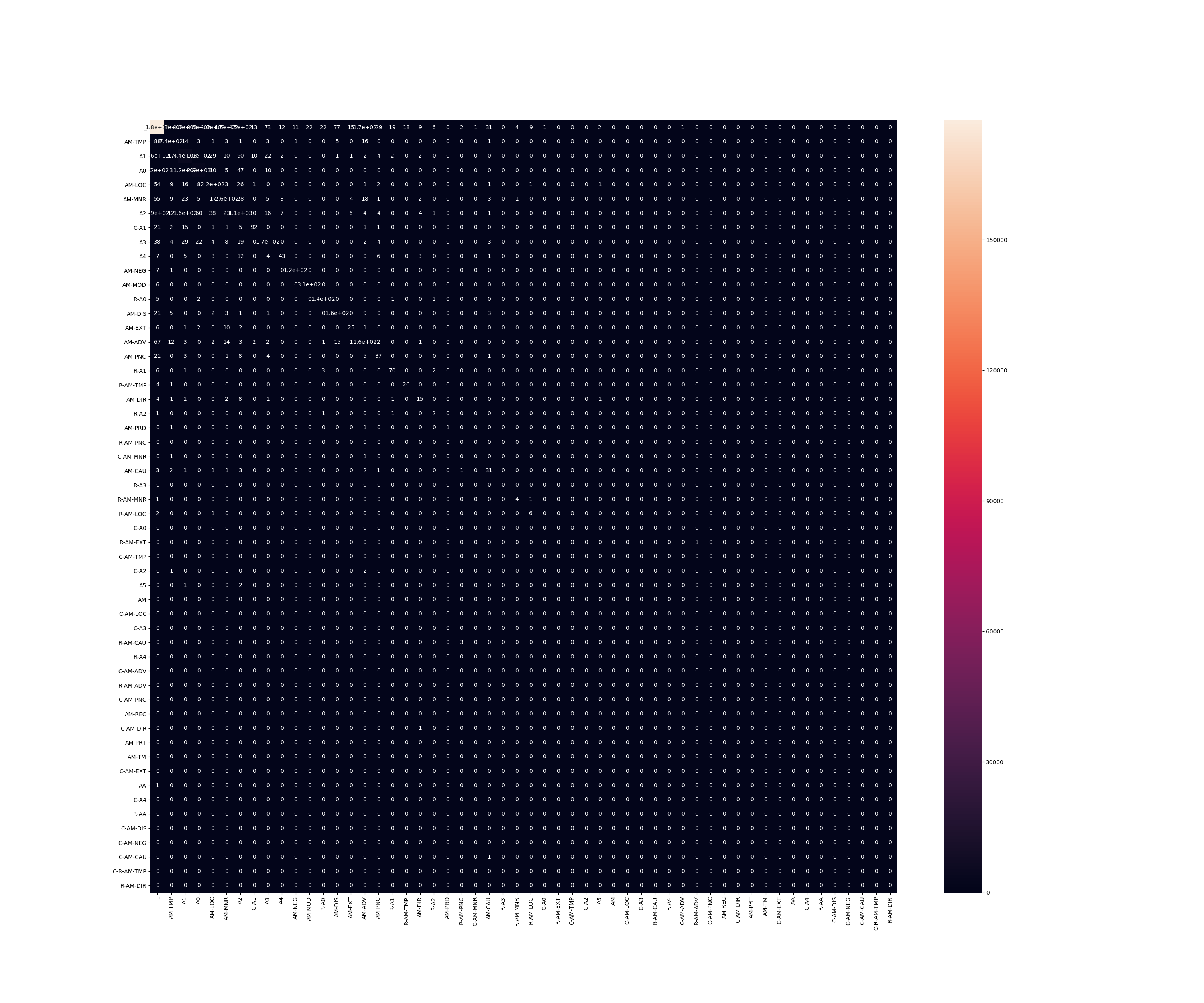
He L., Lee K., Lewis M., Zettlemoyer L. – Deep semantic role labeling: what works and what’s next

Marcheggiani D., Frolov A., Titov I. – A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling

Palmer M., Gildea D., Xue N. – Semantic role labeling



1 - Neural architecture structure that I used



2 - Confusion matrix of the mandatory task