HW1

Good morning, the first homework talks about Named Entity Recognition or simply NER.

What is NER? NER is a tagging problem. I used an IOB scheme to tag: I stands for inside, O for outside, and B for beginning.

The classes are these and for example in the sentence “Jhon went to California to visit Google”, Jhon is a person, California is a location, and Google a corporation.

This is another example showing the use of IOB scheme. My, name and is do not belong to any class and that is why they are categorized as O, instead Robin Hood is a name, so class person, composed of two words, the First one has B and the following ones have I.

The model used takes a sequence of words as input and passes them to a bidirectional LSTM with two layers and then classifies everything with a classifier layer.

Another model used is the same as before with a CRF layer at the end.

CRF is a conditional random field that encourages correct prediction of sequences.

Now let’s talk about of the preprocess

First, I augmented the dataset. To do this, I used the spacy library lemmatizer to find the lemma version of each word within the sentence. The idea is to give the model more general sentences in addition to the normal ones.

Doing this some words were split like the one in the example

In this case an exception is made to the IOB scheme, in fact a B is assigned for each split of this word, obviously if this word belongs to a class

GloVe is a word embedding, there are several implementations, using twitter, wikipedia and so on as corpus.

I used the one based on twitter

With an embedding size of two hundred

Before word emebdding, it is necessary to make a vocabulary.

First, I calculated the frequencies of each word in the dataset

Then, I assigned each word an id

And in case a word is not present, the id of the <unk> token is assigned

Now let's talk about the model input

We need to find the length of the input, which can be a small number or a large number. With a large number I got better results since the whole sentence was almost always processed together

This is the size of a window that scrolls over the sentence, and each frame is an input

For small sentences, padding was added at the end

I have trained for fifteen epochs on average, and these are the hyperparameters of the two models used. As you can see, they are the same except for the min frequency and GloVe embedding parameters. Min frequency discards and puts to unk all words that don’t appear at least this value of times. For the base model it was necessary to eliminate this noise, instead using CRF, the use of GloVe increased the results, and since glove is a very large and pretrained embedding, it was not necessary to exclude words based on frequency.

These are the loss functions, and we can see how the second model lowered it more

These is the results and also here the second model have the better results

And these are the confusion matrices. We can see how the O tag generate a lot of noise in the dataset since is the more frequent class.

HW2

The second homework talks about Semantic Role Labeling or simply SRL. The SRL is the ability to assign the semantic role that a single word plays within a sentence.

We need to answer this question. who did what to whom where and when.

The algorithm is as follows. In step 1 we have to identify predicates, in step 2 we have to disambiguate them, in step 3 we have to identify the arguments of a predicate, and in step 4 we have to classify them

Let’s talk about of the different architectures

I start with the model for steps three and four of the algorithm. I start here because the main goal of the project was to implement these steps, and the model described in this slide is the starting point from which I made the subsequent models, in fact I named it the "base model." This model takes as input the tokenized sentence for the transformer and a tensor that identifies the location of the predicate using an embedding layer. The outputs are then concatenated and passed to a bidirectional LSTM with two layers and then a classifier

I improved this model by adding an embedding of the part-of-speech tagging of the sentence. This emebedding if we consider the algorithm from step two or one, is removed since the pos\_tag "VERB" is in position of the predicates that we don't know

For step one the model is the base model removed predicate embedding, since we must find them

Instead for step two is exactly the base model

Now let’s talk about of the preprocess

The starting preprocess is to use the tokenizer of the specific transformer offered by HuggingFace. The input is the tokenized sentence and if we aren't in step one of the algorithm there is the input of predicate embedding. In this input there is one corresponding to words that are not predicates, two corresponding to predicates, and zero corresponding to initial character, separators, padding, and parts following the first caused by a word split. A split can be seen in the word Transformer which is split in two. This is because the tokenizer splits unfamiliar words in such a way that the first part is a very frequent word in the corpus.

If we need pos embedding, its added to the input by inserting indices for each type of pos\_tag and zero in the cases explained earlier. The zero corresponds to the padding index and in fact those tokens will be ignored by the loss function

A great improvement was achieved by using verbatlas. A separator character was added to the end of the sentence and, in addition to the lemmatized version of the predicate, the two arguments recommended by verbatlas. In these new positions both the pos input and predicate embedding are set to zero

I have trained on average for 30 epochs and these are the hyperparameters used. There are different combinations but what I would like to focus on is the type of transformer used. Both bert-base-uncased and bert-base-cased were tested, however, bert-base-case each time gave worse results and that is why only bert-base-uncased was used.

These are the loss functions, and in all the steps it was minimized a lot

These are the results

As can be seen, using the entire algorithm gives satisfactory results to prove the fact that it works. Of course, in the presence of ground truth of the previous steps the scores increase

I also tried testing in Spanish and French, and these are the hyperparameters

Three tests were conducted for the Spanish dataset in search of the best combination. The first was to use the model used with the English dataset and gave the worst results since the transformer is focused on the English language. Next the same model was tried but with a multilingual transformer and there were good improvements but the best result was fine tuning from the English checkpoint. Most likely even if an English transformer is used, in the checkpoint, many words have been trained enough to the point that the split Spanish words are recognized well. This last technique was also used for the other models and languages

These are the scores, and again the whole algorithm worked reasonably well even though good results for predicate disambiguation were not achieved

that' s all, thanks for your attention