Assignment 1

Due to: 11/01/2022 (dd/mm/yyyy)

If you deliver it by 11/12/2021 your assignment will be graded by 11/01/2022.

Credits: Andrea Galassi, Federico Ruggeri, Paolo Torroni

Summary: Part-of Speech (POS) tagging as Sequence Labelling using Recurrent Neural Architectures

Please, read ALL the following instructions 😃

Intro

In this assignment we will ask you to perform POS tagging using neural architectures

You are asked to follow these steps:

- Download the corpora and split it in training and test sets, structuring a dataframe.
- Embed the words using GloVe embeddings
- Create a baseline model, using a simple neural architecture
- Experiment doing small modifications to the baseline model, choose hyperparameters using the validation set
- Evaluate your two best model
- Analyze the errors of your model

Task: given a corpus of documents, predict the POS tag for each word

Corpus: Ignore the numeric value in the third column, use only the words/symbols and its label. The corpus is available at: https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/packages/corpora/dependency_treebank.zip

Splits: documents 1-100 are the train set, 101-150 validation set, 151-199 test set.

Features: you MUST use GloVe embeddings as the only input features to the model.

Splitting: you can decide to split documents into sentences or not, the choice is yours.

I/O structure: The input data will have three dimensions: 1-documents/sentences, 2-token, 3-features; for the output there are 2 possibilities: if you use one-hot encoding it will be 1-documents/sentences, 2-token labels, 3-classes, if you use a single integer that indicates the number of the class it will be 1-documents/sentences, 2-token labels.

Baseline: two layers architecture: a Bidirectional LSTM layer and a Dense/Fully-Connected layer on top; the choice of hyper-parameters is yours.

Architectures: experiment using a GRU instead of the LSTM, adding an additional LSTM layer, and adding an additional dense layer; do not mix these variantions.

Training and Experiments: all the experiments must involve only the training and validation sets.

Evaluation: in the end, only the two best models of your choice (according to the validation set) must be evaluated on the test set. The main metric must be F1-Macro computed between the various part of speech. DO NOT CONSIDER THE PUNCTUATION CLASSES. [What is punctuation?]{https://en.wikipedia.org/wiki/English_punctuation}

Metrics: the metric you must use to evaluate your final model is the F1-macro, WITHOUT considering punctuation/symbols classes; during the training process you can use accuracy because you can't use the F1 metric unless you use a single (gigantic) batch because there is no way to aggregate "partial" F1 scores computed on mini-batches.

Discussion and Error Analysis : verify and discuss if the results on the test sets are coherent with those on the validation set; analyze the errors done by your model, try to understand which may be the causes and think about how to improve it.

Report: you are asked to deliver the code of your experiments and a small pdf report of about 2 pages; the pdf must begin with the names of the people of your team and a small abstract (4-5 lines) that sums up your findings.

Out Of Vocabulary (OOV) terms

How to handle words that are not in GloVe vocabulary? You can handle them as you want (random embedding, placeholder, whatever!), but they must be STATIC embeddings (you cannot train them).

But there is a very important caveat! As usual, the element of the test set must not influence the elements of the other splits! If you use random embeddings or placeholder embeddings (e.g., everything is 0) this is automatically solved.

If you want to use other techniques: when you compute new embeddings for train+validation, you must forget about test documents. The motivation is to emulate a real-world scenario, where you select and train a model in the first stage, without knowing nothing about the testing environment.

For implementation convenience, you CAN use a single vocabulary file/matrix/whatever. The principle of the previous point is that the embeddings inside that file/matrix must be generated independently for train and test splits.

Basically in a real-world scenario, this is what would happen:

1. Starting vocabulary V1 (in this assignment, GloVe vocabulary)
2. Compute embeddings for terms out of vocabulary V1 (OOV1) of the training split
3. Add embeddings to the vocabulary, so to obtain vocabulary V2=V1+OOV1
4. Training of the model(s)
5. Compute embeddings for terms OOV2 of the validation split
6. Add embeddings to the vocabulary, so to obtain vocabulary V3=V1+OOV1+OOV2
7. Validation of the model(s)
8. Compute embeddings for terms OOV3 of the test split
9. Add embeddings to the vocabulary, so to obtain vocabulary V4=V1+OOV1+OOV2+OOV3
10. Testing of the final model

In this case, where we already have all the documents, we can simplify the process a bit, but the procedure must remain rigorous.

1. Starting vocabulary V1 (in this assignment, GloVe vocabulary)
2. Compute embeddings for terms out of vocabulary V1 (OOV1) of the training split
3. Add embeddings to the vocabulary, so to obtain vocabulary V2=V1+OOV1
4. Compute embeddings for terms OOV2 of the validation split
5. Add embeddings to the vocabulary, so to obtain vocabulary V3=V1+OOV1+OOV2
6. Compute embeddings for terms OOV3 of the test split
7. Add embeddings to the vocabulary, so to obtain vocabulary V4=V1+OOV1+OOV2+OOV3
8. Training of the model(s)
9. Validation of the model(s)
10. Testing of the final model

Step 2 and step 6 must be completely independent of each other, for what concerns the method and the documents. But they can rely on the previous vocabulary (V1 for step 2 and V3 for step 6) THEREFORE if a word is present both in the training set and the test split and not in the starting vocabulary, its embedding is computed in step 2) and it is not considered OOV anymore in step 6).

Report

The report must not be just a copy and paste of graphs and tables!

The report must not be longer than 2 pages and must contain:

- The names of the member of your team
- A short abstract (4-5 lines) that sum ups everything
- A general (brief!) description of the task you have addressed and how you have addressed it
- A brief analysis of the data (class distribution, lenght, etc)
- A short description of the models you have used
- Some tables that sum up your findings in validation and test and a discussion of those results
- The most relevant findings of your error analysis

Evaluation Criterion

The goal of this assignment is not to prove you can find best model ever, but to face a common task, structure it correctly, and follow a correct and rigorous experimental procedure. In other words, we don't care if you final models are awful as long as you have followed the correct procedure and wrote a decent report.

The score of the assignment will be computed roughly as follows

- 1 point for the correctness of approach/methodology
- 1 point for the handling of OOV terms
- 1 point for the correct implementation of models (they must run and do the right thing)
- 1 point for train-validation-test procedure
- 2 point for the discussion of the results, error analysis, and report

The evaluation will not be based on the performance of the models!

We also reserve the right to assign a small bonus (0.5 points) to any assignment that is particularly worthy.

Deliver

- Two files: a pdf file for the report, and a python notebook
- Put the names of the member of the team at the beginning of both files
- What about additional files, for example models or weights? You can upload them in a private cloud and insert the link in the report.
- Please, submit a notebook with clear sections, text boxes, and comments and indications about what is going on in the code. When we do not understand what is going on we get annoyed, when we are annoyed we tend to be more severe in our evaluation.

Contacts

In case of any doubt, question, issue, or help we highly recommend you to check the course useful material for additional information, and to use the Virtuale forums to discuss with other students.

You can always contact us at the following email addresses. To increase the probability of a prompt response, we reccomend you to write to both the teaching assistants. If we do not reply within 3-4 days, please send it again!

Teaching Assistants:

- Andrea Galassi -> a.galassi@unibo.it
- Federico Ruggeri -> federico.ruggeri6@unibo.it

Professor:

- Paolo Torroni -> p.torroni@unibo.it

Additional notes and FAQ

- You can use a non-trainable Embedding layer to load the glove embeddings
- For the baseline, it must have only two trainable layers: the BiLSTM and the Dense/FC one. The Dense layer is the "classification head" with softmax activation. You must not add an additional dense layer on top of the baseline. You can use the embedding layer before the BiLSTM, but it must be not trainable.
- You can use any library of your choice to implement the networks. Two options are tensorflow/keras or pythorch. Both these libraries have all the classes you need to implement these simple architectures and there are plenty of tutorials around, where you can learn how to use them.
- For the application of the Dense Layer, it is recommended to use a Time-Distributed Dense. In any case, doing otherwise is NOT considered an error.
- Some examples of things you can analyze in your discussion and error analysis are: the performances on the most frequent classes and the less frequent classes, precision and recall, the confusion tables, specific misclassified samples.
- Punctuation: you must keep the punctuation in the documents, since it may be helpful for the model, you simply must ignore it when you perform the evaluation of the model, not considering the

punctuation classes among the ones you use to compute F1 macro score. If you are curious, you can run ADDITIONAL experiments where you remove punctuation to see its impact.