Ministry of Higher Education and Research
Higher School of Computer Science 08 May 1945 - Sidi Bel Abbes

Second Year Second Cycle - Artificial Intelligence and Data Science

Lab 04 : Named entity recognition with sequence to sequence models

Presented By : FELLAH Abdelnour.

Date: May 3, 2024

1 Introduction

Named entity recognition is a subtask of information retrieval and natural language processing that involves detecting and classifying named entities such as organizations, locations, and persons. In this lab report, we will explain our approach and discuss the obtained results of using sequence-to-sequence models, mainly Bidirectional LSTM, for such tasks, the source code as well as the generated models can be found at : https://github.com/abdelnour13/Natural-Language-Processing-Labs/tree/main/TP-4.

2 Dataset

For this task we will be using Conll2003 dataset available for download via this link : https://huggingface.co/datasets/conll2003, this dataset contains approximately 21 thousands sentences with there corresponding POS,Chunk and NER tags splitted into three subsets (train,validation and test),we are only interested in the NER tags and the sentecnes,this dataset contains nine different NER tags :

- O: Represents the ”Outside” tag, indicating that the token is not part of any named entity.
- B-PER: Represents the ”Begin-Person” tag, indicating the beginning of a named entity that is a person.
- I-PER: Represents the ”Inside-Person” tag, indicating a token inside a named entity that is a person.
- B-ORG: Represents the ”Begin-Organization” tag, indicating the beginning of a named entity that is an organization.
- I-ORG: Represents the ”Inside-Organization” tag, indicating a token inside a named entity that is an organization.
- B-LOC: Represents the ”Begin-Location” tag, indicating the beginning of a named entity that is a location.
- I-LOC: Represents the ”Inside-Location” tag, indicating a token inside a named entity that is a location.
- B-MISC: Represents the ”Begin-Miscellaneous” tag, indicating the beginning of a named entity that is miscellaneous (i.e., not a person, organization, or location).
- I-MISC: Represents the ”Inside-Miscellaneous” tag, indicating a token inside a named entity that is miscellaneous.

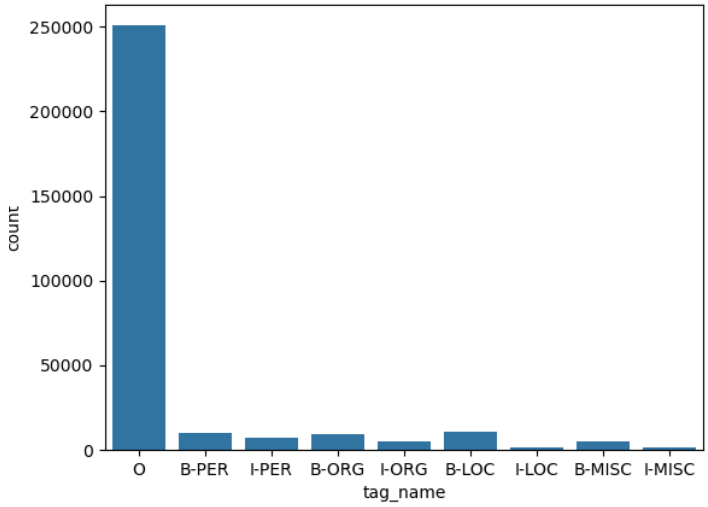


Figure 1: The distribution of the NER tags in the Conll2003 dataset.

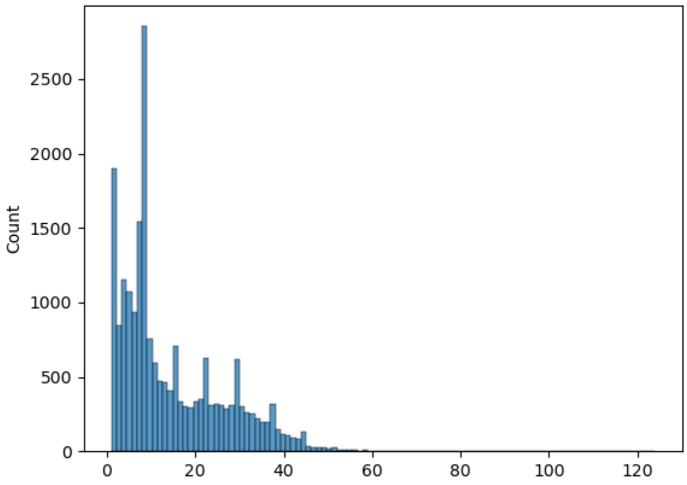


Figure 2: The histogram of sentences lengths.

3 Preprocessing

For preprocessing we only converted all the tokens to lower case and then we replaces all the tokens with their index in the vocabulary and then the sequences were padded to have all the same length.

4 Vectorization

For vectorization we tried the following approaches :

- Use an embedding layer to learn word embeddings with back-propagation (end-to-end approach).
- Use pret-rained GLoVe vectors,we experimented with different dimensions : 50,100,200 and 300.
- Use pre-trained fast text with skip-gram on ‘English Wikipedia Dump of February 2017‘ corpus, the dimensions of vectors is equal to : 300.

5 Architecture

The architecture used is composed of an embedding layer that either learns the word vectors or is initialized with pre-trained vectors and then freezed,tow biderctional LSTM layers and final fully connected layer below is a visual representation of the architecture :

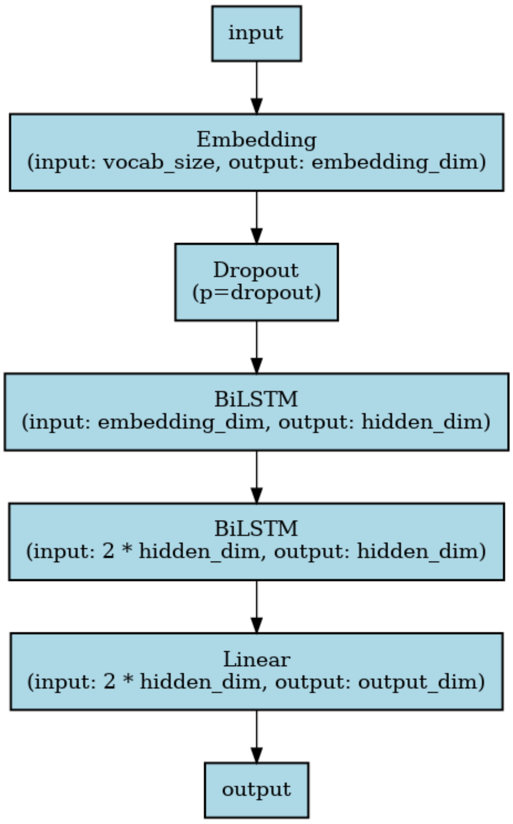

input
Embedding
(input: vocab_size, output: embedding_dim)
Dropout
(p=dropout)
BiLSTM
(input: embedding_dim, output: hidden_dim)
BiLSTM
(input: 2 * hidden_dim, output: hidden_dim)
Linear
(input: 2 * hidden_dim, output: output_dim)
output

Figure 3: A visual representation of the architecture.

6 Training

All the different models were trained using AdamW optimizer and cross entropy loss with the following hyper-parameters :

Learning rate	0.00025
Weight decay	0.0025
Batch size	32
Embedding dimension	64
Hidden dim	32
Dropout rate	0.3
Epochs	100

Table 1: Hyper-parameters

NOTE : the specified value for the embedding dimension concerns the end-to-end approach.

During training, the average loss, accuracy, and macro F1 score were tracked for each epoch :

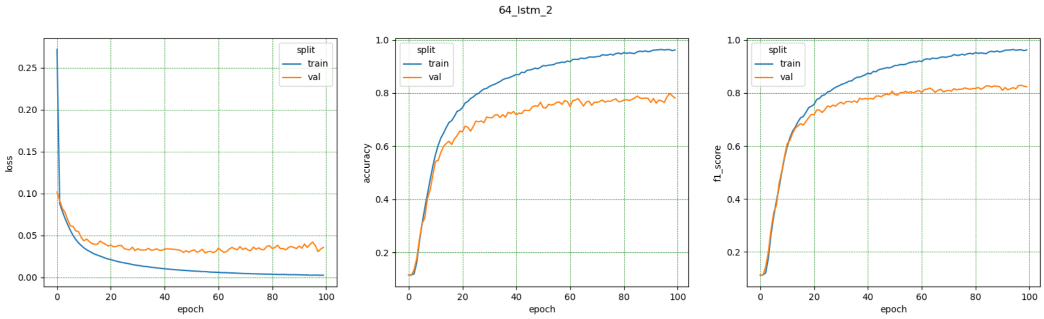


Figure 4: Model with trained word embedding

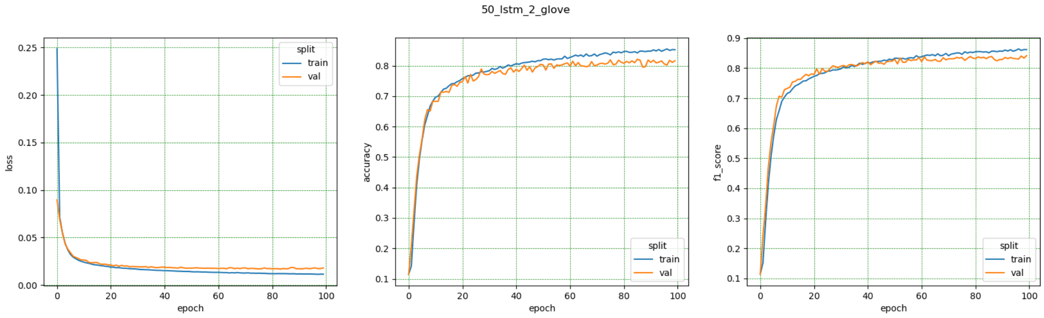


Figure 5: Model with pre-trained GLoVe-50 word embedding

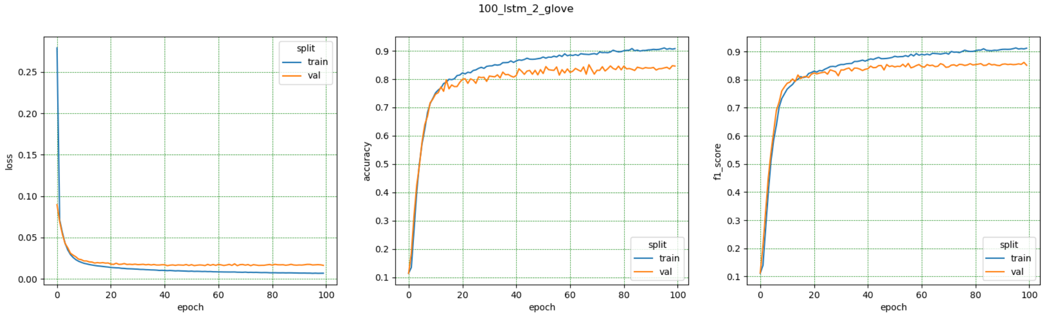


Figure 6: Model with pre-trained GLoVe-100 word embedding

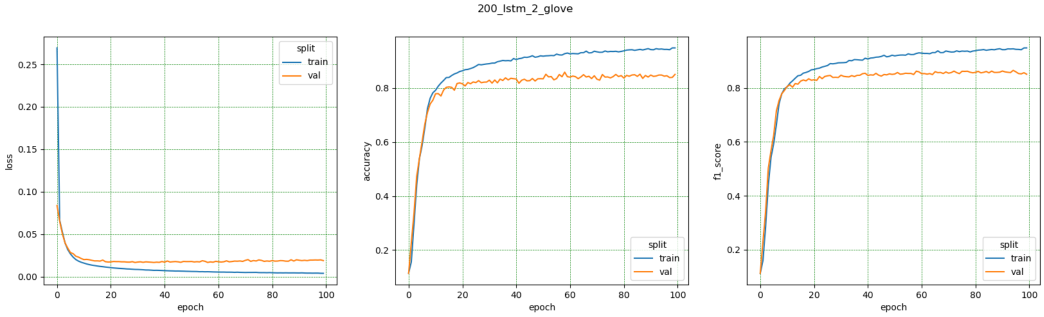


Figure 7: Model with pre-trained GLoVe-200 word embedding

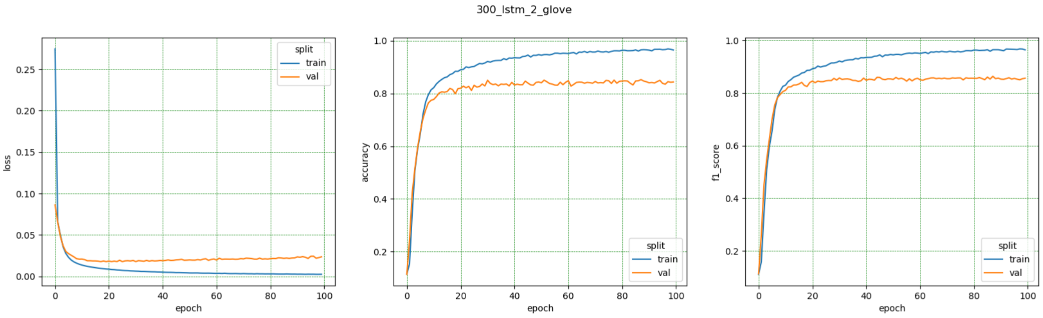


Figure 8: Model with pre-trained GLoVe-300 word embedding

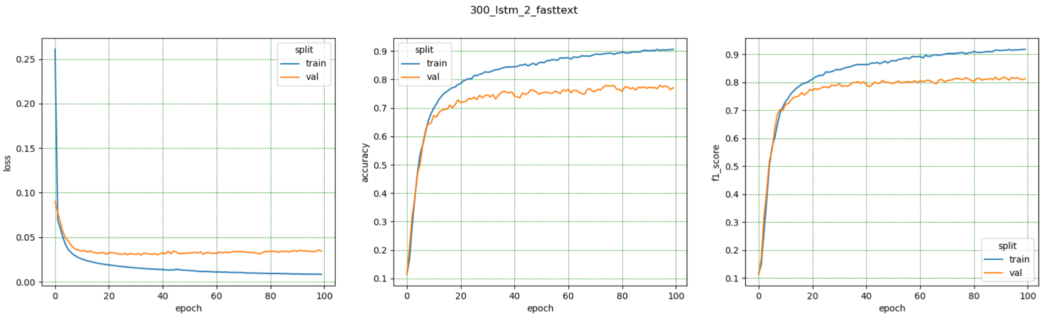
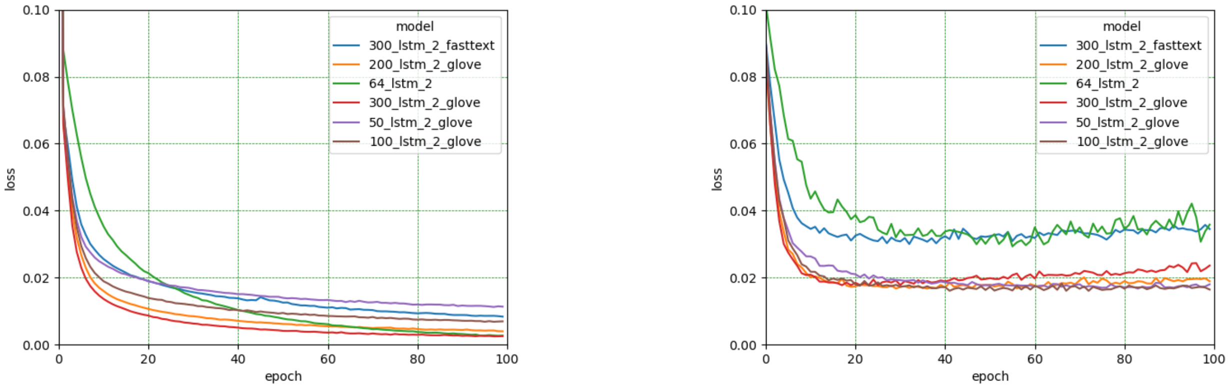


Figure 9: Model with pre-trained FastText word embedding

Comments :

- The model with learned word embedding shows severe overfitting and relatively low validation accuracy / f1-score,due to the model learning the words representationsfrom the small train set it has.
- The models trained using GLoVe have higher validation accuracy/f1-score.
- The models trained using GLoVe-50 and GLove-100 vectors shows less overftitting then the other models,while GLoVe-300 shows overfitting due to the high dimentiality of the embeddings.
- FastText shows both overfitting due to the high dimentiality and and low validation accuracy due the large number of OOV (out of vocabulary) tokens.

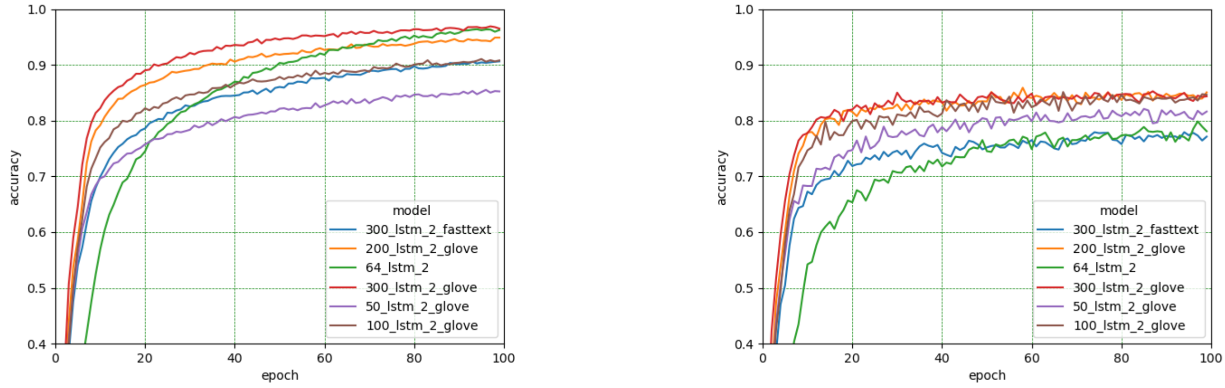
Here below are some comparative plots to help further understand the differences in performance between the different models :



(a) Training loss

(b) Validation loss

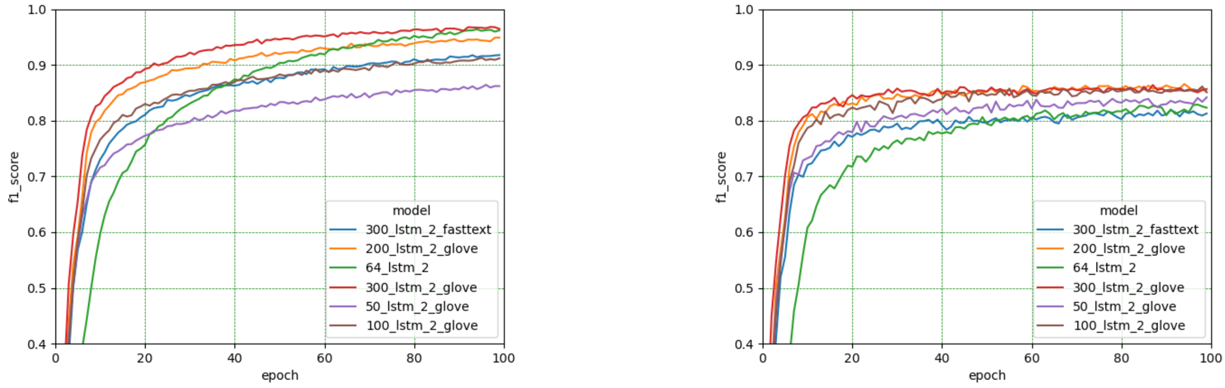
Figure 10: Validation and training loss



(a) Training accuracy

(b) Validation accuracy

Figure 11: Validation and training accuracy



(a) Training f1 score

(b) Validation f1 score

Figure 12: Validation and training f1 score

7 Results :

Here we will present the obtained results of the different models on the training,validation and test sets :

Embedding	GLoVe-50	GLoVe-300	GLoVe-200	Embedding Layer	FastText	GLoVe-100
accuracy	0.916	**0.996**	0.991	0.994	0.958	0.972
f1_score	0.938	**0.997**	0.990	0.996	0.970	0.974
auc	0.999	**1.000**	0.999	**1.000**	0.999	**1.000**
precision	0.962	**0.997**	0.988	0.998	0.983	0.977
recall	0.916	**0.996**	0.992	0.994	0.958	0.972

Table 2: Training set results

Embedding	GLoVe-50	GLoVe-300	GLoVe-200	Embedding Layer	FastText	GLoVe-100
accuracy	0.803	0.836	**0.845**	0.773	0.763	0.842
f1_score	0.850	**0.868**	0.863	0.837	0.825	0.865
auc	**0.999**	**0.999**	**0.999**	0.998	0.995	**0.999**
precision	0.914	0.907	0.886	**0.919**	0.905	0.892
recall	0.803	0.836	**0.845**	0.773	0.763	0.842

Table 3: Validation set results

Embedding	GLoVe-50	GLoVe-300	GLoVe-200	Embedding Layer	FastText	GLoVe-100
accuracy	0.759	0.797	0.798	0.695	0.698	**0.799**
f1_score	0.797	**0.820**	0.807	0.763	0.762	0.812
auc	**0.999**	0.998	**0.999**	0.997	0.995	**0.999**
precision	0.841	0.846	0.819	**0.851**	0.846	0.827
recall	0.759	0.797	0.798	0.695	0.698	**0.799**

Table 4: Test set results

8 Conclusion :

In conclusion, this lab report explored the application of sequence-to-sequence models, specifically Bidirectional LSTMs, in the task of Named Entity Recognition (NER). The experiment involved training and evaluating several models with different configurations by varying embedding dimensions and different types of word embeddings (GloVe and FastText).

Our results indicate that sequence-to-sequence models, especially those utilizing Bidirectional LSTMs, can effectively capture the contextual information necessary for accurate NER. The models showed strong performance in terms of accuracy, F1-score, AUC, precision, and recall across different embedding dimensions and types.

Furthermore, our findings suggest that the choice of word embeddings can significantly impact the performance of the NER model. While GloVe embeddings performed well, FastText embeddings showed a higher number of out-of-vocabulary (OOV) tokens, indicating the need for further investigation into preprocessing steps such as lemmatization.

Overall, this study highlights the effectiveness of sequence-to-sequence models in NER tasks and emphasizes the importance of selecting appropriate word embeddings for optimal performance. Future research could explore other advanced architectures and preprocessing techniques to further improve NER performance.