MNLP Homework 1 Report: Event Detection

Ludovico Comito
Sapienza University of Rome
comito.1837155@studenti.uniroma1.it

Abstract

This report describes the work involved in solving the first homework of the MNLP class. The task proposed is Event Detection, where the goal is to identify and categorize events in a sequence of text. The proposed work provides a solution based on the use of GloVe word embeddings and a Bidirectional LSTM, as well as providing empirical observations on various experiments involving different RNN-based architectures.

1 Introduction

Event Detection can be considered a particular instance of Sequence Labelling where the task is about identifying and categorizing events in a sequence of text. In this task, an event is considered a specific occurrence involving participants. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are two types of Recurrent Neural Networks (RNNs) that fit well this kind of tasks, given their ability to capture long-term dependencies in sequential data. The work presented in this report describes a series of experiments based on LSTM and GRU architectures, involving hyperparameter tuning and the choice of different ways of representing word vectors using word embedding models.

1.1 LSTM based architecture

LSTM (Long Short-Term Memory) [2] networks are a type of recurrent neural network (RNN) that can capture long-term dependencies in sequential data. An LSTM unit is composed of a cell state, input gate, forget gate, and output gate. The cell state represents the memory of the LSTM, which can be modified by the gates, that regulate what information to remember or to forget. The proposed LSTM architecture consists takes as input the batched inputs in the shape [batch_size, sequence_length, embedding_dimension] and feeding it to an LSTM. Dropout is then applied to the output hidden state of the LSTM in order to regularize the network and prevent overfitting [4]. Lastly, the output of the dropout is fed to a fully connected layer that will output the logits corresponding to the probabilities for each label.

1.2 GRU based architecture

Gated Recurrent Unit (GRU) [1] networks are a type of RNN that were born to solve the vanishing gradient problem, which is a typical flaw of classic RNNs. GRUs are based on the reset and update gate which directly affect their hidden state to determine how much of the previous hidden state and how much of the new input to incorporate into the new hidden state. The GRU based architecture keeps the same structure as the one proposed above, by substituting the LSTM layer with a GRU-layer.

1.3 GloVe word embeddings

Global Vectors (GloVe) [3] is an unsupervised learning algorithm for obtaining vector representations for words. The main intuition behind GloVe is that semantic relationships between words can be derived from the co-occurrence matrix. Given a corpus having N words, the co-occurrence matrix X will be an NxN matrix, where the i-th row and j-th column of X, X_{ij} denotes how many times word i has co-occurred with word j. GloVe is a global, count-based word embedding approach that combines co-occurrence statistics of words in a corpus with a matrix factorization method to generate dense, low-dimensional word vectors. The presented experiments use the glove-wiki-gigaword 200 and 300 managed using the Gensim library.

2 Methodology

2.1 Data pre-processing

The train, validation and test datasets are provided in the jsonl format, which are imported using the Pandas library. The first pre-processing step consists in converting all the token characters to low-

ercase, while labels are translated to indices by indexing them through a dictionary. The lowercasing step helps by reducing the vocabulary size and ensuring consistency by reducing the variability of the tokens. This is because the same word may appear in different forms (uppercase, lowercase, title case etc.) in different parts of the text. Subsequently, each sentence and the corresponding labels are padded according to the length of the longest sequence present in the dataset using a special <PAD> token. This is done because both LSTMs and GRUs are designed to process fixed-length sequences. The last step involves creating the word embedding for each token. Unknown words are handled by assigning them a vector of the same dimensionality as the model's embeddings but initialized using random values. Moreover, the padding tokens are assigned to an all-zeros vector, which is believed to discriminate better the padding token from the others.

2.2 Model training

The training of the model is handled by the trainer class, which computes the model's loss, accuracy and f1 score for each epoch at the training and validation steps. The chosen loss criterion was Cross-Entropy, which is performed by ignoring the padding label. The chosen optimizer was Adam.

2.3 Model selection

The key performance metric chosen for this task is the f1 score, which is computed using the seqeval library. The selection was performed by storing the model at the epoch where it performed the best f1 score.

2.4 Model testing

The models were tested on the test set by computing their f1-score, accuracy score and classification report (provided by seqeval). Moreover, a confusion matrix is plotted to better investigate the model's performances.

3 Experimental setup

Exploratory Data Analysis on the training set revealed some key properties of the dataset. In particular an analysis on the labels frequencies shows that the 'O' labels make up the 90.35% of all the labels present in the dataset, as shown in Table 1 and in ?Plot 1. By contrast, the seven least frequent labels out of the 11 total labels make just the 1,39% of the total, with the last four being below the 0.1%. The extremely unbalanced nature of this dataset consists in a challenge for the classification tasks for two main reasons. The first one is the risk of overfitting, since the model could rapidly learn to predict only 'O' labels and achieve reasonable results, especially in terms of loss and accuracy. The second reason being that the features of extremely underrepresented labels are very challenging to extract, given their poor frequency.

4 Results

Each model was trained for 30 epochs and tested on its f1-score and accuracy using the functions provided by the seqeval library. The examples reported in this documents all share the following hyperparamers, which have proven to yeld the best results: dropout=0.2, batch size=50, learning rate=0.001. As shown in Table 3, the best performing model was a 5 layer Bidirectional LSTM using the glove-wiki-gigaword-300 with an f1-score of 0.71 and an accuracy of 0.95. A noticeable impact in performance is given by the choice of the word embeddings dimensionality. In fact, as shown in tables 3 and 2, in most cases we see an improvement in performances when going from a dimensionality of 200 to one of 300 on the same model. Moreover, with the same hyperparameters, bidirectional models have been shown to perform better than their non-bidirectional counterparts only when increasing the number of hidden layers. Noticeably the GRU models achieved comparable performances to the LSTMs with the same hyperparameters, but using an average of 28.5% fewer parameters. It is interesting noticing that, when observing the confusion matrix (4) of the model that scored the best and the one that scored the worst (5), the best one was able to predict more consistently the labels different from 'O' especially, still with bad performances in the labels that where least frequent in the training dataset. Lastly, an interesting result was given by the Bidirectional GRU with 5 hidden layers and embedding dimensionality of 300 (Confusion Matrix 6), where it was able to predict correctly 'I-SCENARIO' and 'I-POSSESSION' labels which appeared in the training dataset with a frequency respectively of 0.09% and 0.005%. A possible hypothesis might be that fewer parameters can make GRUs less prone to overfitting, thus being more efficient in the memorization of rare events

Label	Frequency percentage
O	90.357%
B-ACTION	4.379%
B-CHANGE	2.584%
B-SCENARIO	1.238%
B-SENTIMENT	0.589%
B-POSSESSION	0.528%
I-CHANGE	0.113%
I-SCENARIO	0.098%
I-ACTION	0.089%
I-POSSESSION	0.005%
I-SENTIMENT	0.004%

Table 1: Frequency in percentage of the labels in the training dataset in descending order.

Model	Accuracy	F1-Score	Number of trainable parameters
Bi-LSTM 5 layers	0.95	0,70	5.794.860
Bi-LSTM 1 layer	0.94	0,69	394.908
LSTM 1 layer	0.95	0,66	2.065.692
LSTM 5 layers	0.94	0,66	2.065.692

Table 2: Performances results for the LSTM models trained using word embeddings of dimension 200.

Model	Accuracy	F1-Score	Number of trainable parameters
Bi-LSTM 5 layers	0.95	0.71	12.010.060
LSTM 1 layer	0.95	0.70	830.508
Bi-LSTM 1 layer	0.95	0.70	1.661.004

Table 3: Performances results for the LSTM models trained using word embeddings of dimension 300.

Model	Accuracy	F1-Scotr	Number of trainable parameters
gru_BI_glove_300_5-layers	0.95	0.70	9.009.516
gru_BI_glove_300_1-layer	0.94	0.70	1.247.724
gru_glove_300_1-layer	0..95	0.69	623.868
gru_glove_300_5-layers	0.95	0.68	3.213.756

Table 4: Performances results for the GRU models trained using word embeddings of dimension 300.

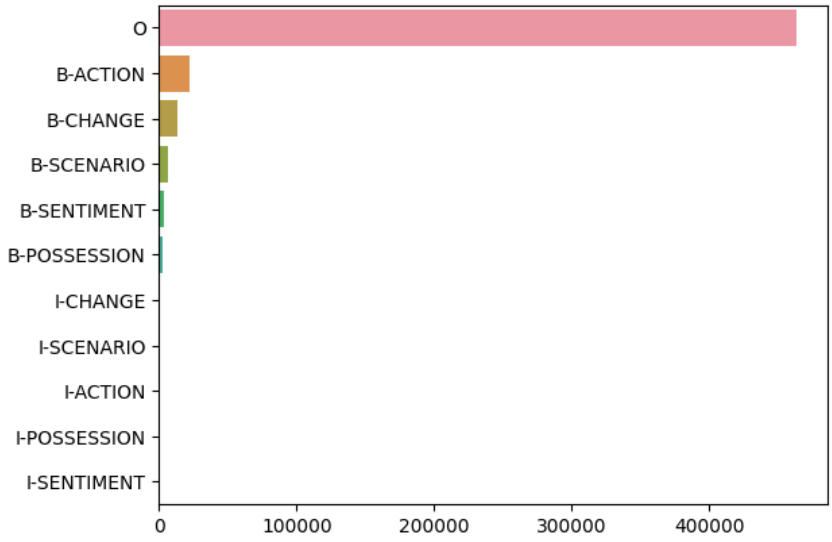

Figure 1: Plot showing the presence of labels in the traing set.

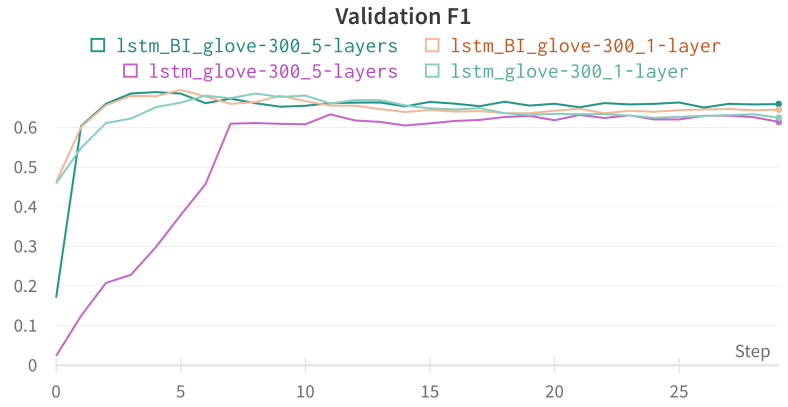

Figure 2: Evolution of the F1-score during training. All the plots refer to LSTM-based architectures using embeddings of dimensionality 300

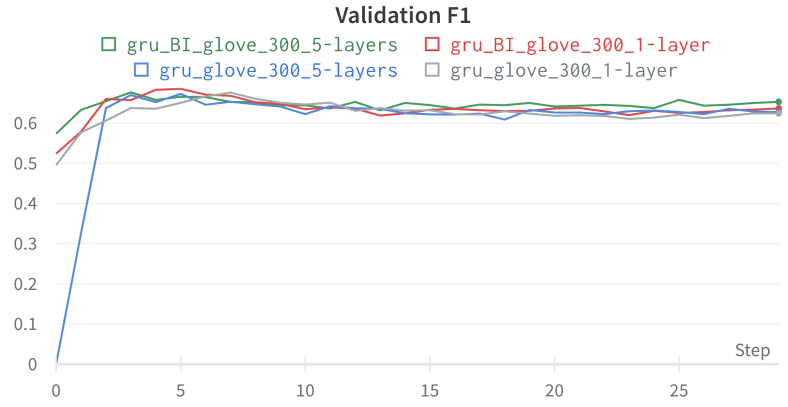

Figure 3: Evolution of the F1-score during training. All the plots refer to GRU-based architectures using embeddings of dimensionality 300

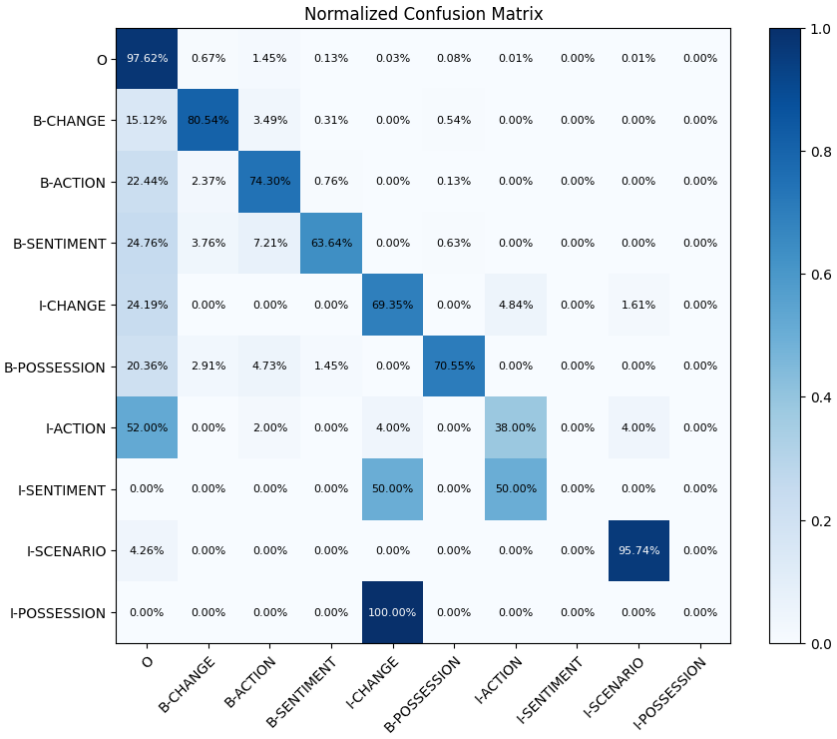

Figure 4: Confusion matrix relative to the performance with the best F1-score (Bi-LSTM 5 layers, GloVe 300)

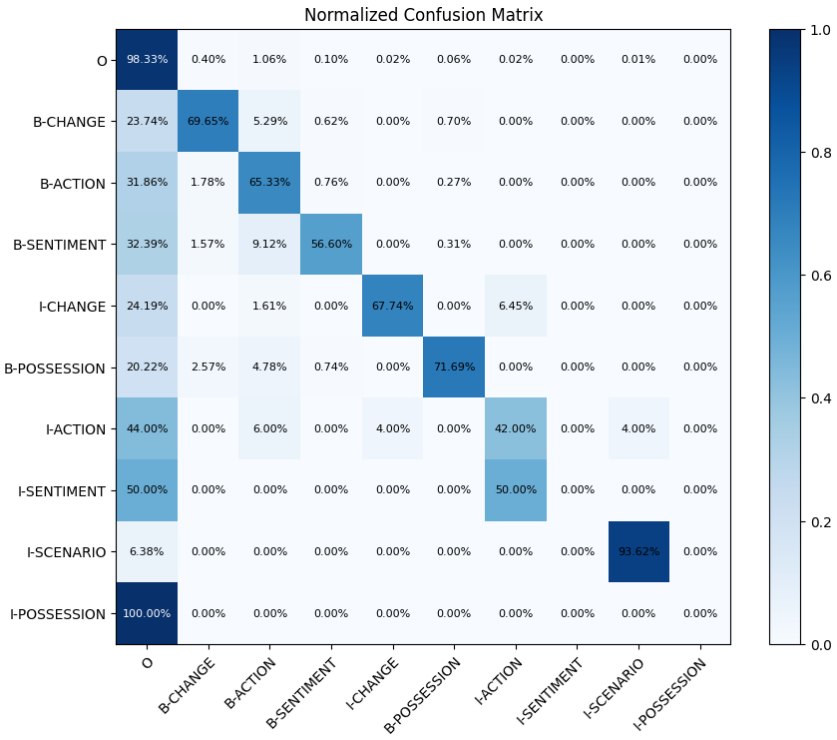

Figure 5: Confusion matrix relative to the performance with the word F1-score (Bi-LSTM 2 layers, GloVe 200)

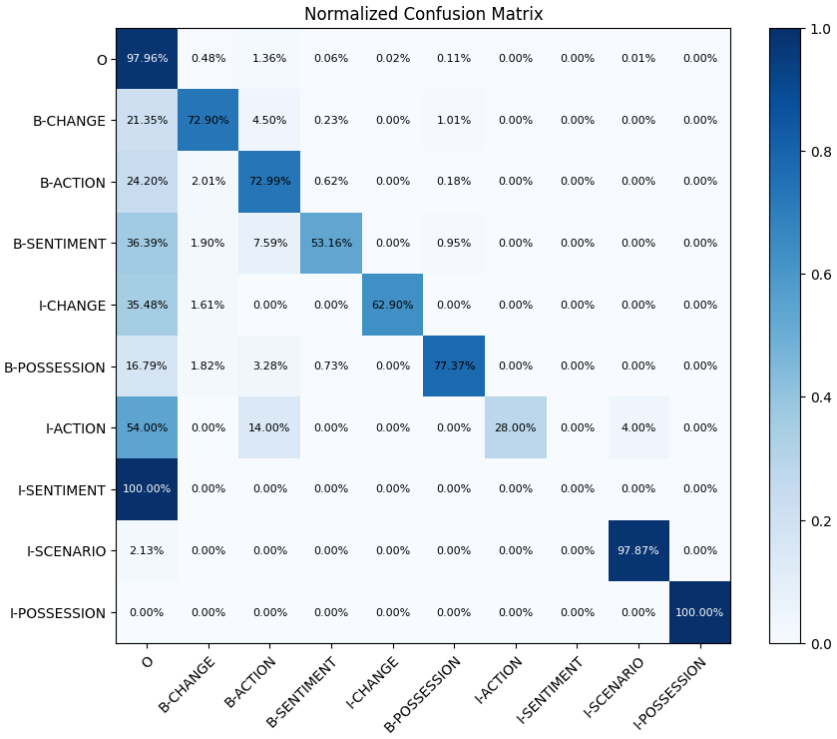

Figure 6: Confusion matrix relative to the performance with the best F1-score (Bi-GRU 5 layers, GloVe 300)

References

[1] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[2] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[3] Jeffrey Pennington, Richard Socher, and Christopher D Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[4] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.