Multilingual Natural Language Processing 2024

Report Homework 1b

Stefano D'Urso

1 Task 27 - HaSpeeDe 3

Task 27, Hate Speech Detection, aims to classify some tweets as hateful or not using two `.jsonl` files. Each line in the files is an object with the attributes:

- **"id"**, that is the tweet identifier,
- **"text"** that contains the tweet content,
- **"choices"** which can be "Non contiene odio" or "Contiene odio", if the tweet is not hateful or if it is.
- **"label"** that in the first case would be set as 0, 1 otherwise.

The dataset contains 5.600 training samples and 4.400 test samples. As no validation set is provided, the training set was split to allocate the 20% of its samples for validation, maintaining the same distribution of the original samples. The resulting distributions for each set are shown in Fig.1, resulting in 4480 training samples, 1120 for validation and the unvaried 4400 test samples. As it is possible to see, the dataset is unbalanced, with the 60-70% of samples belonging to class "0". To execute the notebook, it's enough to run all the cells in `hw1b.ipynb`

1.1 Model

The model is a BiLSTM one, whose architecture is shown in Fig.2. The model takes as input a sequence of words from a text, these then pass through an Embedding layer, transforming them into a dense vector representation. The output then passes through 4 BiLSTM layers, each LSTM layer has 256 hidden units. After each layer, a droput of 0.4 is applied. The output then passes through a fully-connected layer that produces an output belonging to one of the two possible classes. The model is trained for 10 epochs. The parameters choices have been done through a Grid Search, whose code is shown in the `src/` folder. Many hyperparameters are tested: the number of hidden dimensions is chosen between `[128,256,512]`, the number of BiLSTM layers among `[3,4,5]`, the dropout within `[0.3,0.4,0.5]` and the number of epochs between `[5,10,15,20,25]`.
The model uses a vocabulary to manage the words in the text, it is extracted from the train dataset and is used to index the words. For each dataset a dataloader is used to manage the loading of the data.

1.2 Baselines

To evaluate the model effectiveness, a Random Baseline has been implemented. Baselines serve as a reference model used to compare its performance to the one of more complex models. They are just a starting point from which is possible to decide a performance standard that is meant to be surpassed by more advanced approaches. The Random Baseline chooses uniformly at random between one of the two labels, resulting in an accuracy of 51.96% on validation set and 50.45% on the test set.

1.3 Results

The model named `haspeede3_tagger` is trained using the Adam optimizer with a learning rate of 0.0001 and log_steps set to 100. The accuracy on the validation set and the losses on both the train and the validation sets are reported in Fig.3.
An evaluation on the test set is then performed, taking into account not only the accuracy, but also metrics like precision, recall and f1-score. As a matter of fact, as previously stated, the dataset is very unbalanced and the accuracy may not be the best metric to evaluate the model. To better visualize the unbalance, a confusion matrix is plotted in Fig.4: the model successfully predicts the class "0" 3118 times and misclassifies it 95 times. As for class "1", it is well predicted 371 times and misclassified 816. Infact, considering the Recall metric, it's possible to see that in the first class it's 0.97, while for the second one only 0.31, meaning

that many of the "1" instances are not correctly classified.
Comparing the performance of the model to the baseline's, it appears anyway that the first one outperforms the second, indeed the accuracy of the model on test set is 0.79, while the one of the baseline is 0.5.

1.4 Word2Vec

Another simpler model has been implemented using Word2Vec, which is a word embedding technique that learns the embeddings to show semantics similarities between words based on the context of the data.
The model is trained on a set of sentences, learning dense vector representations for words in the text. An average vector is computed for each text in the training set, serving then as input features for the train of the logistic regression classifier that is used afterwards. Similarly, for each text of the test set the average vector is computed, but if a word in the test set is not present in the learned vocabulary, it is replaced with a vector of zeros. Subsequently, a logistic regression classifier is computed, making predictions on the test set. The model performs in a good way, obtaining 0.73 in accuracy, but still being outperformed by the haspeede3_tagger model.
A visual representation of the word embeddings in two dimensions is visualized in Fig.5

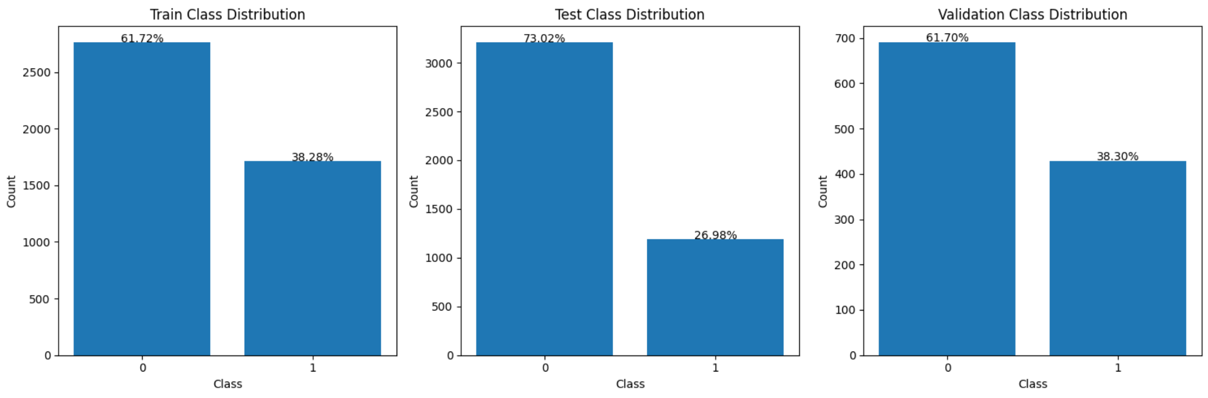


Figure 1: Classes Distribution. Train: "0": 61.72%, "1":38.28%. Test: "0":73.02%, "1":26.98%. Validation: "0":61.70%, "1":38.30%

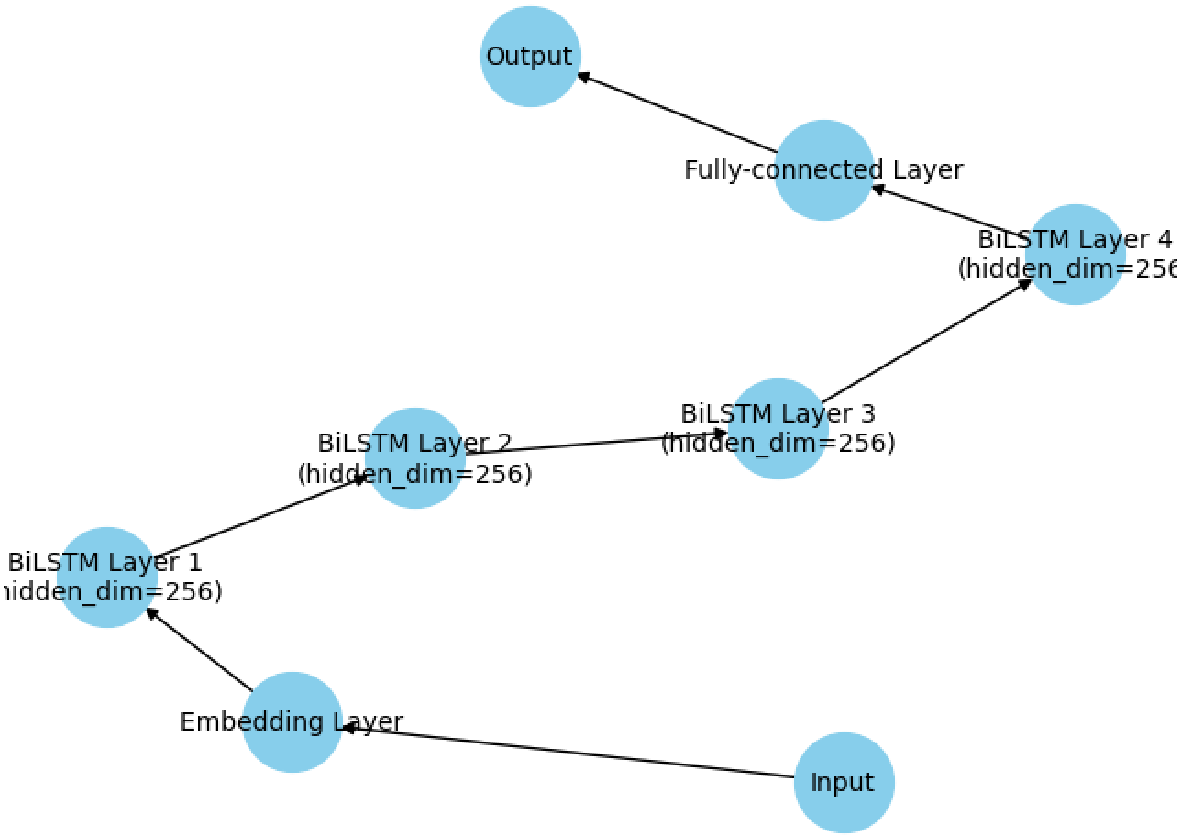
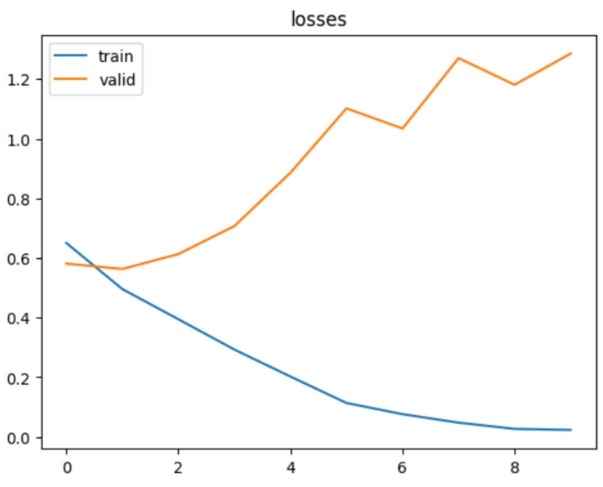
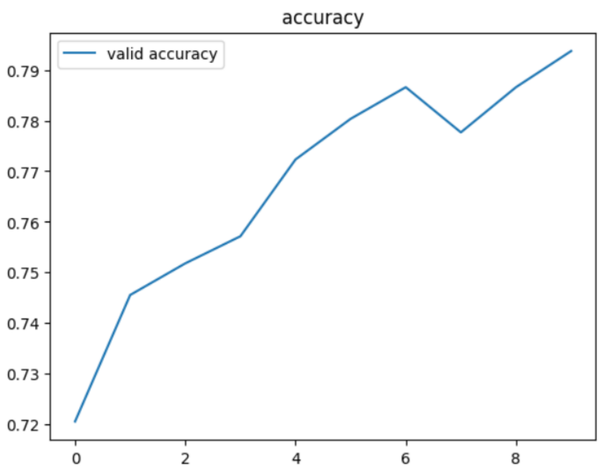


Figure 2: Model Architecture



(a) Loss on train and validation.



(b) Accuracy on validation.

Figure 3

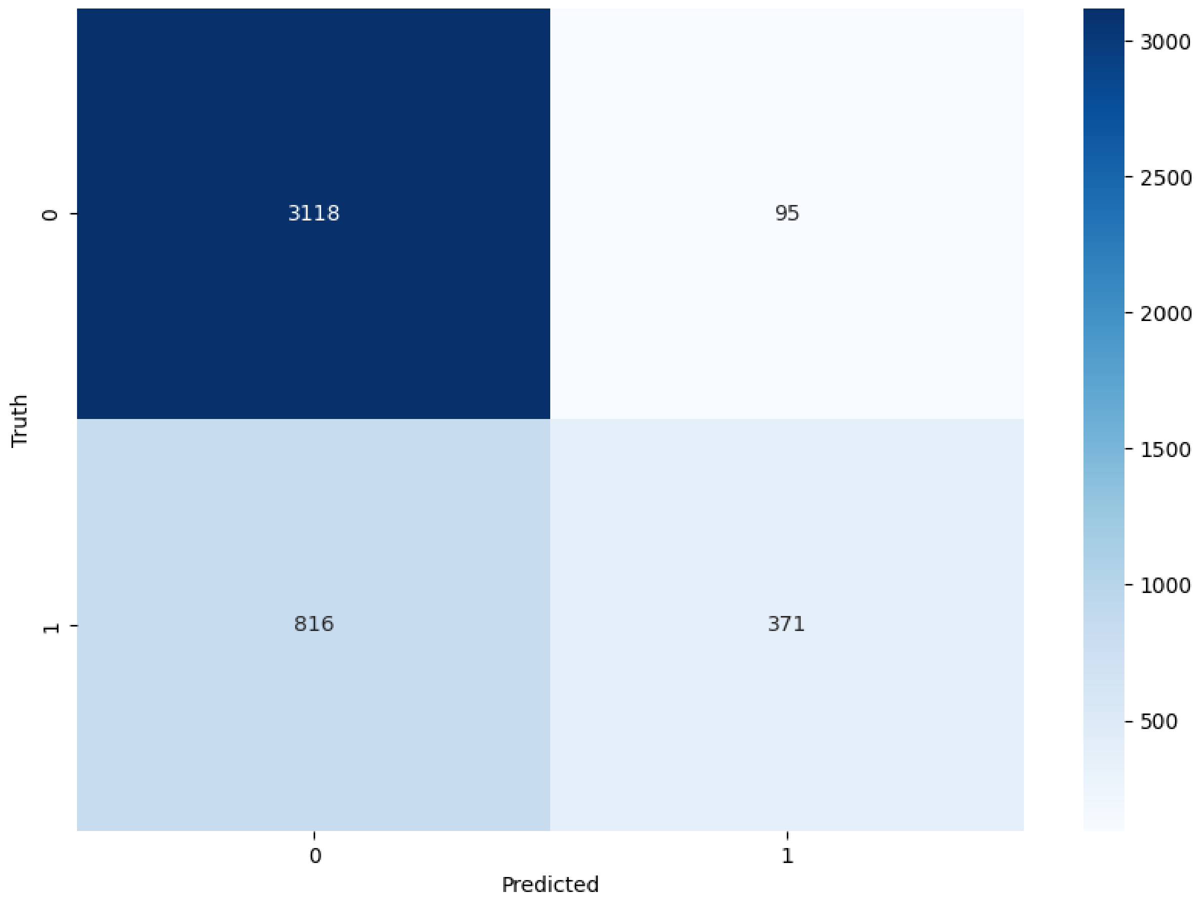


Figure 4: Confusion Matrix

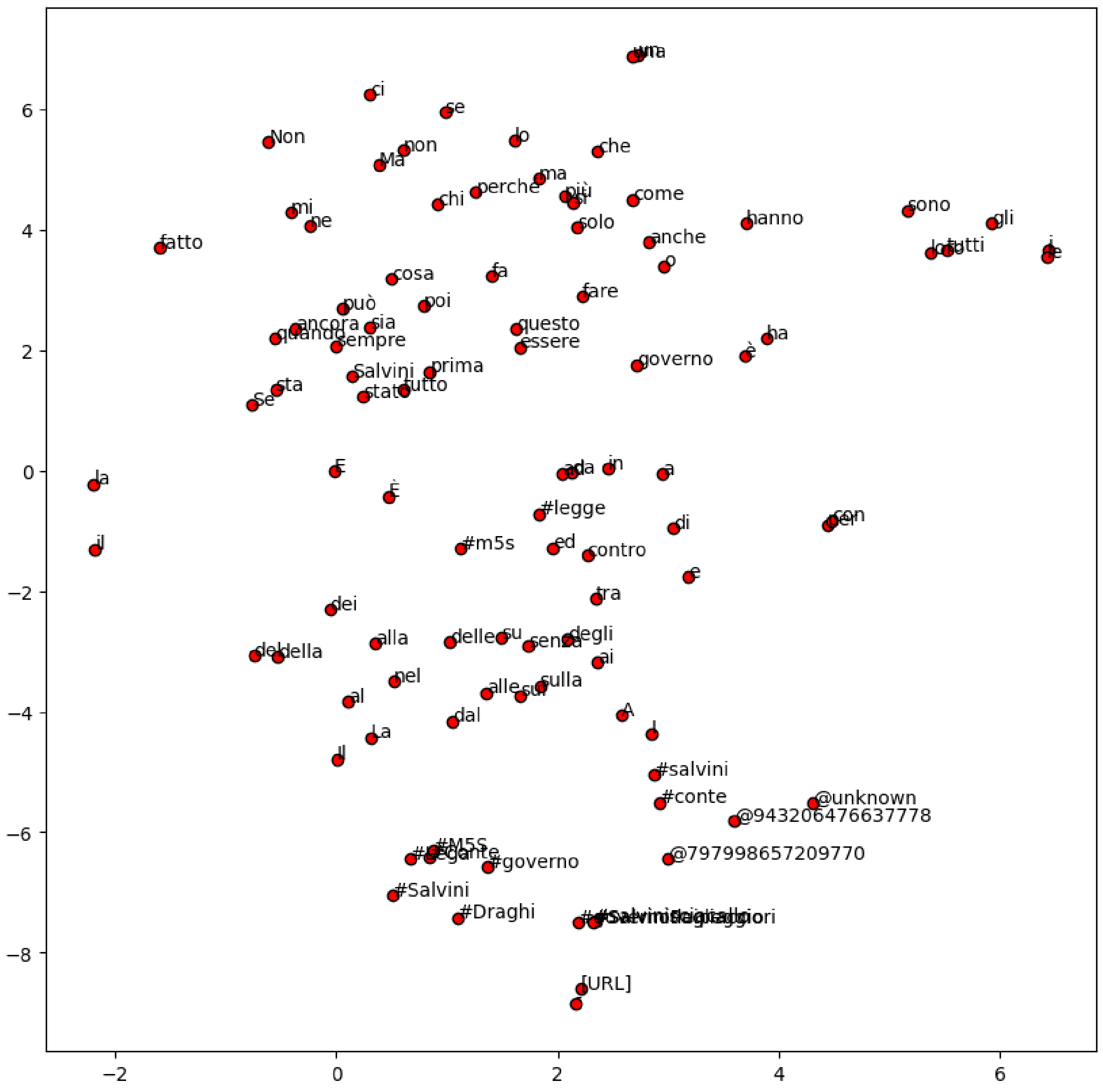


Figure 5: Word Embeddings visualization: similar words in a semantically way are near