Task-8 SadiStance Classifier

Nemish Murawat-2056142

Abstract

Today, understanding the text through Machine Learning methods, especially models capable of explicit sequential data relation modelling, has been widely adopted in various use cases. This work focuses on creating such models in the form of a sentiment(sentence) classifier for the SardiStance Dataset. Further, the aim is to compare modelling ability between non-recurrent and recurrent-based models. As a result, the model based on the combination of 2D-CNN Character-level embedding+ Spacy Word MultiHash Embedding +BILSTM performed the best among the model architectures considered with test accuracy **64.8 %** and Macro-F1 score: **0.50**

1 Dataset Description

In this work tweets in Italian were scrapped from Twitter regarding the Sardinian Movement to evaluate LLM's ability for Stance detection. This dataset was created for the EVALITA 2020 challenge. The training data is composed of 2132 tweets with over 1028 in Against,589 in favour and 515 being neutral towards the Sardinian Movement. These are organised in **.csv** file with **tweet-id,user-id,text and label** being the fields present in it. Further based on the work in HW-1 two files **train.jsonl** and **test.jsonl** were created. Lastly, to perform Hyperparameter Search, the train set was split into **80/20**, to get the validation set via Stratified Sampling.

2 Architectures-Main Model

2.1 Spacy-MutliHash Embedding

MultiHash embedding [1] is a memory-efficient technique used in natural language processing to generate unique word vectors without storing separate vectors for each word. It leverages a hashing trick inspired by Bloom filters to summarize word form, subword information, and shape, reducing memory footprint while maintaining effective representations for known and unknown words. I used the it_core_news_lg spacy Italian pipeline to retrieve **static word embedding** with 300 dimensions.

2.2 Character-Based Word Embedding

To make word representation more informative, I attempted to obtain token embedding via character-based embedding. This is established by performing 2D convolution and Max Pooling operations on learned character embedding to obtain token-level embedding.This architecture is motivated by previous works of [2].Figure 1 shows the schematic view of the CNN-based architecture used. This is final concatenated with MultiHash embedding.

2.3 Parts-of-Speech word embedding

Since the spacy pipeline provided additional informative artefacts such as POS tags, I thought of incorporating this information via the concatenation of learned POS tag embeddings to the other two embeddings for the token.Though after Hyperparameter Search, this turned out to be the less optimal.

2.4 Recurrent Layer:Sentence Representation

To leverage the sequential nature of the data aka text, I explored pipelines using architectures involving 2 layers of BI-RNN,BI-LSTM and BI-GRU. To obtain sentence representation, I used two approaches. The first focussed on using the final hidden layer states(Forward and Backward) while the latter averages over the token output vectors.

2.5 Non-Linearity and Classification Head

After getting the Sentence Embedding from the Recurrent Layer, I explored adding Non-Linearity to increase the separation of classes. For this, I

utilised 1D convolutions over the embedding dimensions with the activation function. However, the best-performing model architecture did not utilise it, hence pointing to suboptimal addition. Finally, the classification head is a simple Linear layer.

3 Hyperparameter Search

I performed quite an exhaustive search of over **1440** combinations requiring nearly about 20 hours on Kaggle GPU T4. In Table 1 all the Hyperparameters and their search space are defined. Based on the highest Validation Accuracy obtained, the highlighted values are the chosen combination.

4 Simpler Model & Baseline

In building the Simpler Model, I focussed on experimentation with Static Spacy MutliHash embedding(not fine-tuned) and Feed-Forward Network. Sentence representation was obtained by averaging over the tokens and fed to FFNs. I have also included two dropout layers,one after the token embedding and the other on the sentence embedding. I performed a much smaller Hyperparameter search(due to fewer Hyperparameters). The search space and the best-performing Hyperparameters are shown in Table 2 and chosen values are in highlighted. Finally, the baseline is created through **Stratified Sampling**.

5 Results

The performance of the three models on the Validation and Test dataset is shown in Table 3. One can observe from the table that simple and main models are clear improvements over Baseline. However, the performance metric of Accuracy does not differ much between the simple and main models. On further investigation based on the Macro F1 score and Confusion matrix(shown in Fig 2,3), I can state the main model is a better and balanced classifier. However, in case of less computation complexity, a simple model would be selected. The main model results show test accuracy of **64.8 %** and Macro-F1 score of **0.50**

6 Code Instructions

The code for the task is present in 'mnlp-hw1b-2056142.ipynb' and the required files are in the data folder. Also alternatively, you can use the following Kaggle Notebook link and use **run all** to execute.

A Appendix

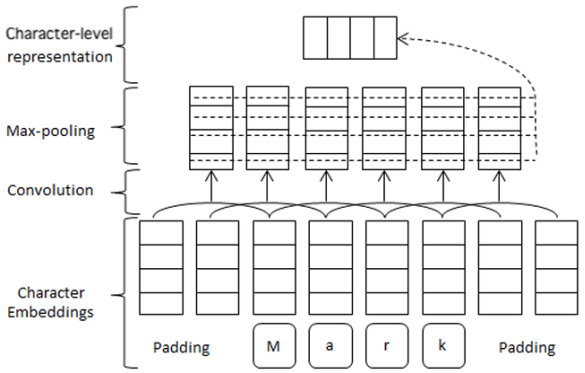

Figure 1: CNN based Architecture to obtain Character-Level Representations

Hyperparameter Name	Search Space
char_emb_space	**True**, False
char_emb_dim_space	**N**one,30, 50
char_kernel_size_space	2, **5**
pos_emb_space	True, **False**
pos_emb_dim_space	**None**,30, 50
dropout_space	**0.3**, 0.5
rec_space	Rnn, Gru, **Lstm**
lr_space	1e-3, **1e-4**
weight_decay_space	**1e-4**, 1e-3
average_tokens_space	**True**, False
cnn_lstm_space	True, **False**

Table 1: Hyperparameter Search Space Main Model

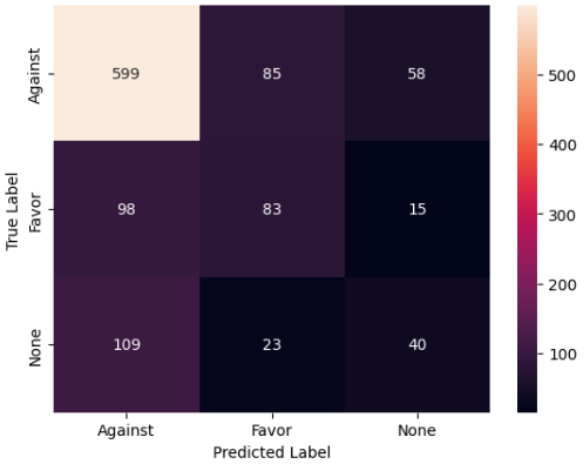

Figure 2: Confusion Matrix of Main Model

Hyperparameter Name	Search Space
dropout	**0.3**, 0.5
lr	1e-3, **1e-4**
weight_decay	1e-4, **1e-3**
add_linear	True, **False**
linear_hid_fac	**None**,0.5, 1, 1.5, 2

Table 2: Hyperparameter Search Space for Simpler Model

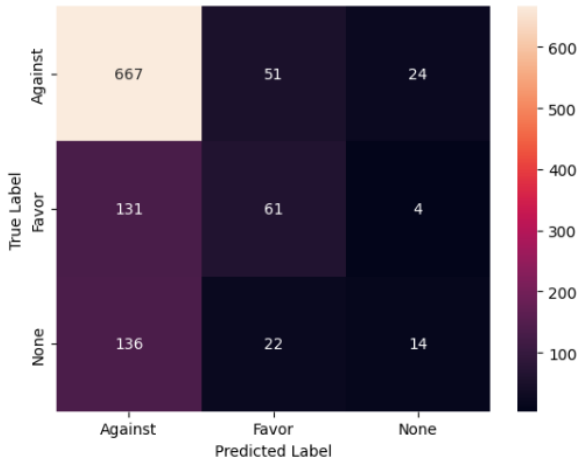

Figure 3: Confusion Matrix of Simple Model

Model	Validation Accuracy	Validation Macro F1	Test Accuracy	Test Macro F1
Baseline	0.252	0.285	0.270	0.262
Simple Model	0.537	0.369	0.669	0.432
Main Model	0.572	0.492	0.649	0.495

Table 3: Model Performance

B References

References

[1] Kádár Á. Boyd A. Van Landeghem S. Søgaard A. Honnibal M Miranda, L. J. 2022. Multi hash embeddings in spacy. *Arxiv*.

[2] D. Q. Zenan Zhai. 2018. Comparing cnn and lstm character-level embeddings in bilstm-crf models for chemical and disease-named entity recognition. *Journal of the Association for Computing Machinery*.