UNIVERSITÀ DI PISA

Master Degree in Artificial Intelligence and Data Engineering

Computational Intelligence and Deep Learning

CyberBullying Classification

Lorenzo Bianchi, Edoardo Morucci

Github repository:
https://github.com/EdoardoMorucci/CyberBullying-Classification

Contents

1 Abstract

Cyberbully and its impact have occurred around the world and now the number of cases is increasing. The detection of cyberbullying is very important because the amount of information on internet is too large and humans are not able to track all this information.

The purpose of this research is to create a classification model with high accuracy, we built different models like LSTM, CNN and BERT to address the identification and classification problem of recognize 6 classes of cyberbullying. Two word embeddings are used as input of the models, one from spaCy and one created by us using only the dataset.

Finally we obtained the best results with an Ensemble of our best models, with an accuracy of 0,8633.

2 Introduction

Cyberbullying is the manifestation in the Internet of a wider phenomenon better known as bullying. The latter is characterized by violent and intimidating actions exercised by a bully on a victim.

Nowadays technology allows bullies to infiltrate victims' homes by manifesting themselves at any time of their life via internet. The most used ways for cyberbullying are social networks, through them in fact it is very simple to send message discriminatory and offensive to the victims.

Cyberbullying detection is very important because the online information is too large, so it is not to be tracked by humans. For this reason, we decided to deepen this topic by focusing on the famous social network Twitter. The purpose of this research is to create a model to identify and classify cyberbully conversations.

After preprocessing the dataset, we created two different types of word embedding, for the first we used the pre-trained spaCy library while for the second we created a custom word embedding using only our dataset. Later we used multiple neural networks for classification.

- Long Short-Term Memory (LSTM)
- Convolutional Neural Network (CNN)
- Bidirectional Encoder Representations from Transformers (BERT)

Finally, we joined the various models in an Ensemble, first using an average of the predictions of the models extracting the ensemble classified label and then using a weighted average of the predictions so that best classifiers have higher weights than the worst classifiers.

3 Related Work

With the rise of automatic text classifications techniques, in recent years, automatic cyberbullying detection in social media has attracted the attention of the scientific community.

The earlier works are from Dinakar et al.[1] and Yin et al.[2], respectively from 2011 and 2009, in which they demonstrate the researchers' interest in detecting cyberbullying events in social media textual data using supervised machine learning tools. On 2017, Davidson et al.[3], aimed to identify different types of cyberbullying on Twitter data via a multiclass classifier, carrying on a similar task with a similar objective with respect to ours.

During the COVID-19 pandemic, cyberbullying has become an even more serious threat, and Wang et al.[5] work aims to investigate the viability of an automatic multiclass cyberbullying detection model that is able to classify whether a cyberbully is targeting a victim's age, ethnicity, gender, religion, or other quality. The dataset used for this work has been uploaded on Kaggle[4] on February 2022, given the opportunity to many users, including us, to use it. Among the solutions posted on the website, most of them use Word2Vec embeddings and uses techniques as CNN and Bert. From these works we can see how the higher difficulties are in classifying other type of cyberbullying, and most of the works remove these category in order to achieve cleaner results.

4 Dataset

The dataset we chose consists of 47k tweets and is available on Kaggle. Each tweet is labelled according to the classes of cyberbullying that are: age, ethnicity, gender, religion, other type and not cyberbullying.

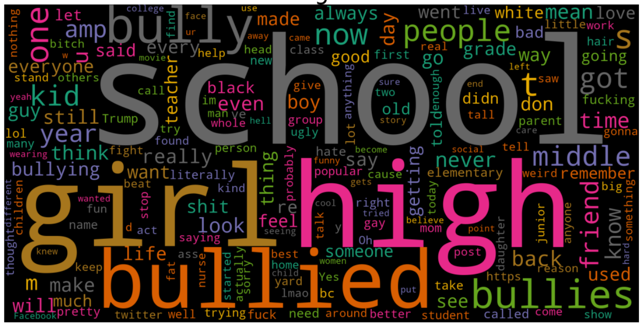
The data are balanced and contain about 8'000 rows per class. Each line of the dataset consists of two columns, one for tweet text and the other for the associated cyberbullying category.

In analyzing the dataset we asked ourselves the question about the elimination or not of the class "other type of cyberbullying", altough we will see (post data preprocessing) that this class will be in a clear minority compared to the others, however, we decided to keep it because we believe that every tweet that represents some form of cyberbullying should be recognized and classified as such. Just think about what the ultimate goal of this project would be and it is clear the importance in recognizing all kinds of cyberbullying.

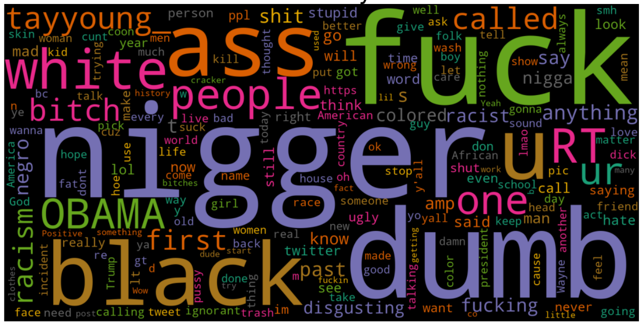
4.1 Dataset Visualization

Below we show the most used word for each type of cyberbullying.

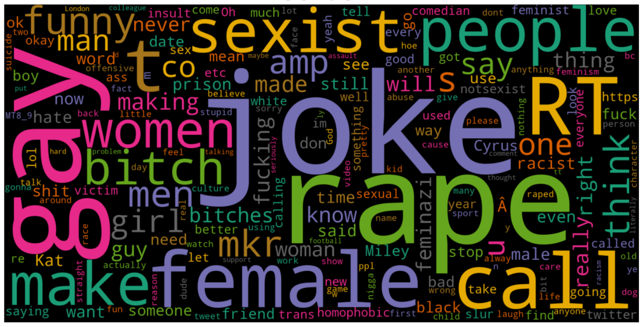
Age



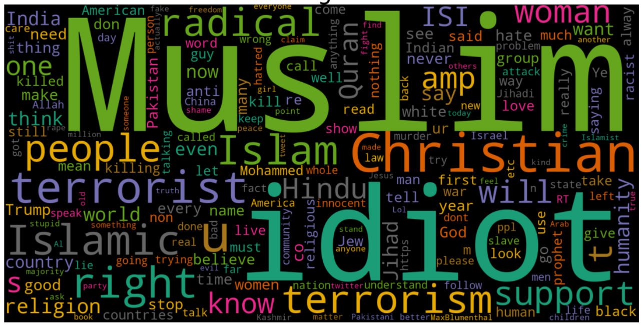
Ethnicity



Gender



Religion



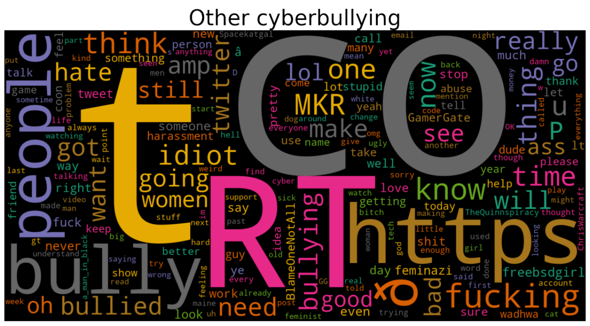


Figure 2: Word Cloud of each class

From the word clouds we can see how the word in other cyberbullying are more generic and less dense of meaning than the words present in other categories.

4.2 Data preprocessing

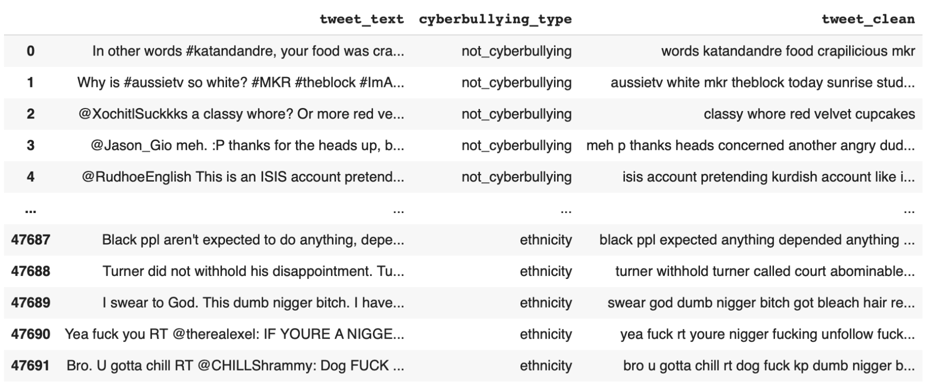
Once we downloaded the dataset from Kaggle we started the preprocessing phase, first of all we have to clean the text of the tweets. To do this we removed:

- Punctuation
- Links
- Emoji
- Stopwords
- Mentions
- New line character
- Do not ascii character
- Special symbols like $, &, #
- Multi sequential spaces

The impact of punctuation symbols on the effectiveness depends on our task. Since we are doing a semantic analysis there is no need to keep the punctuation, in fact we decided to remove it because it is considered as Noisy data.

Later we removed the contractions of the words, such as "can’t" in "can not", in such a way as to standardize these pairs of words by dividing them and leaving the extended version as standard.

Below we can see an example of the result of what we have just described:



	tweet_text	cyberbullying_type	tweet_clean
0	In other words #katandandre, your food was cra...	not_cyberbullying	words katandandre food crapilicious mkr
1	Why is #aussietv so white? #MKR #theblock #ImA...	not_cyberbullying	aussietv white mkr theblock today sunrise stud...
2	@XochitlSuckkks a classy whore? Or more red ve...	not_cyberbullying	classy whore red velvet cupcakes
3	@Jason_Gio meh. :P thanks for the heads up, b...	not_cyberbullying	meh p thanks heads concerned another angry dud...
4	@RudhoeEnglish This is an ISIS account pretend...	not_cyberbullying	isis account pretending kurdish account like i...
...	...	...	...
47687	Black ppl aren't expected to do anything, depe...	ethnicity	black ppl expected anything depended anything ...
47688	Turner did not withhold his disappointment. Tu...	ethnicity	turner withhold turner called court abominable...
47689	I swear to God. This dumb nigger bitch. I have...	ethnicity	swear god dumb nigger bitch got bleach hair re...
47690	Yea fuck you RT @therealexel: IF YOURE A NIGGE...	ethnicity	yea fuck rt youre nigger fucking unfollow fuck...
47691	Bro. U gotta chill RT @CHILLShrammy: Dog FUCK ...	ethnicity	bro u gotta chill rt dog fuck kp dumb nigger b...

Figure 3: Representation of the dataset with the new column of cleaned tweed.

After removing the duplicate tweets we decided to remove the tweets with a number of words less than or equal to 3, since they are too short to be considered relevant for our analysis. At the end of this process the dataset is unbalanced, with the minority class (other_cyberbullying) being just over half the size of the majority class.

We have therefore adopted date augmentation which we will see in the next chapter. But before we splitted the dataset, we decided to use 60% for the training set, 20% for the validation set and 20% for the test set. It's important to split the dataset before doing data augmentation because we want to leave the test and validation sets as they are to have a more truthful evaluation.

4.3 Dataset balancing

After the preprocessing, the duplicate removal and the dataset splitting we have the following distributions of classes for the the training and test sets:

```
religion               6309
age                    6279
ethnicity              6143
gender                 5875
not_cyberbullying      5138
other_cyberbullying    3695
Name: cyberbullying_type, dtype: int64
```

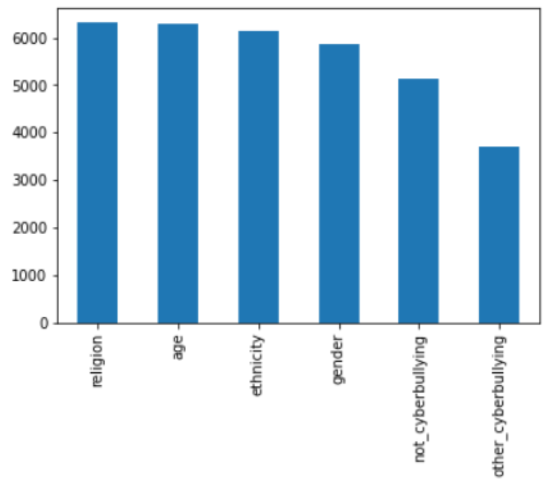
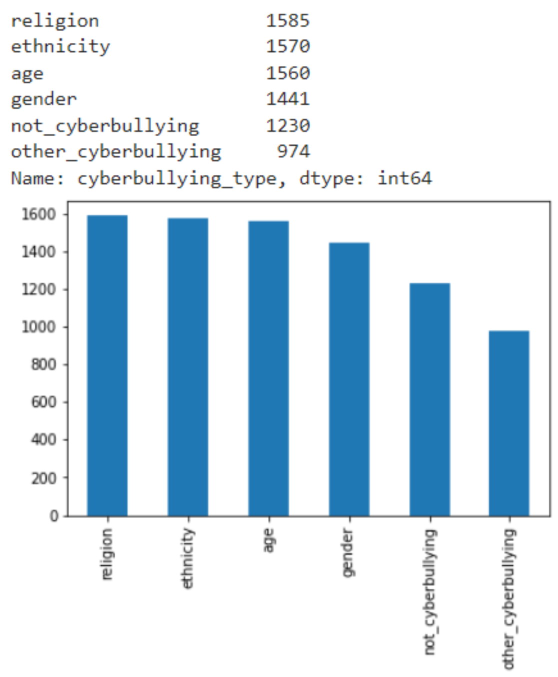


Figure 4: Train dataset class distribution



```
religion               1585
ethnicity              1570
age                    1560
gender                 1441
not_cyberbullying      1230
other_cyberbullying     974
Name: cyberbullying_type, dtype: int64
```

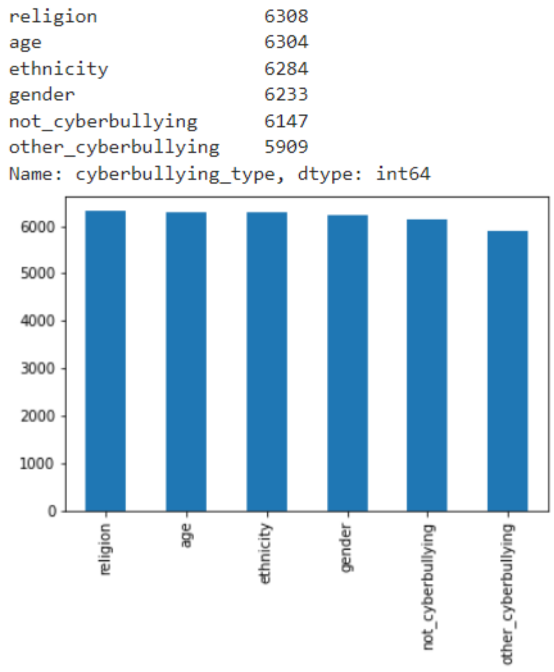
Figure 5: Test dataset class distribution

As we can see the dataset is extremely unbalanced, with the other type of cyberbullying which has almost the half of elements from the the religion and age classes. Though it is not a problem an unbalanced test set, a training set unbalanced can lead to problems during the training phase: feeding imbalanced data to a classifier can make it biased in favor of the majority class, because it did not have enough data to learn about the minority. To solve this issue we initially considered two options: undersampling and oversampling.

Since we did not have so much data, we thought that undersampling could led to a very poor dataset. Considering also that the differences between the occurrences per classes is big, balancing the classes removing tweets in order to obtain the number of occurrences for each class equal to the one of other type of cyberbullying would have led to remove about $\frac{1}{3}$ of the dataset. Instead oversampling could increase the overfitting of our model, making it more capable to recognize tweets duplicated.

Even if a combination of the two techniques could be a good tradeoff between the information loss and the overfitting problem, we opted for another solution which seemed to be better: the Data Augmentation. We used the library **nlpaug**[6] in order to replace 1 or 2 words of a tweet with a synonym, and add the new tweet to the dataset. This technique, beyond balancing the dataset, differently from the oversampling improve the capa-

bility to generalize of our models making it capable of recognizing different tweets, reducing overfitting. An example of data augmentation, with the preprocessed tweets, is the following:
Original tweet: boy snapchat calling black girl n****r saying disgusting things
Augmented tweet: male child snapchat calling black girl n****r saying disgusting things
We balanced the dataset randomly sampling tweets from the minority classes and applying the data augmentantation, adding the new generated tweet to the dataset, and repeating this procedure until we reached an equal distribution of classes (6309 elements for each class). During this process some duplicated tweets are created and so duplicate removal is redone, obtaining the following final distribution:



```
religion               6308
age                    6304
ethnicity              6284
gender                 6233
not_cyberbullying      6147
other_cyberbullying    5909
Name: cyberbullying_type, dtype: int64
```

Figure 6: Train dataset class distribution after data augmentation

As we can see the dataset is almost balanced and ready to be used for training, and after this procedure the validation set is created.

5 Word Embeddings

Now we have to create numerical representations of words in a predefined vector space, so our neural network will be able to interpret words. With one-hot enconding we would have for each word a numerical vector as large as the vocabulary where each cell is at 0 except that of the word itself which is at 1. This representation is sparse and is too large for the neural network which will not generalize.

We then used word embedding which is a learned representation for text where words that have the same meaning have a similar representation. The key to the approach is the idea of using a dense representation distributed for each word, this is learned according to the use of words, naturally capturing their meaning.

First of all we Created our Vocabulary from all tweets via TextVectorization of Keras.

We decided to create two different word embeddings, one using the spacy library and the other using a custom approach by towing the network with only our dataset.

5.1 Spacy Word Embedding

Then we created an embedding matrix where each line corresponds to the dense representation (dim_embedding=300) of a word.

Not all words in our dictionary will be present in the spaCy dictionary, so many words will be represented by a vector containing only zeros. These "missing words" are about 55% of our vocabulary, although almost all of them occur less than 5 times in the various tweets. This means that they are uncommon and rarely used words, so we do not worry about this fact.

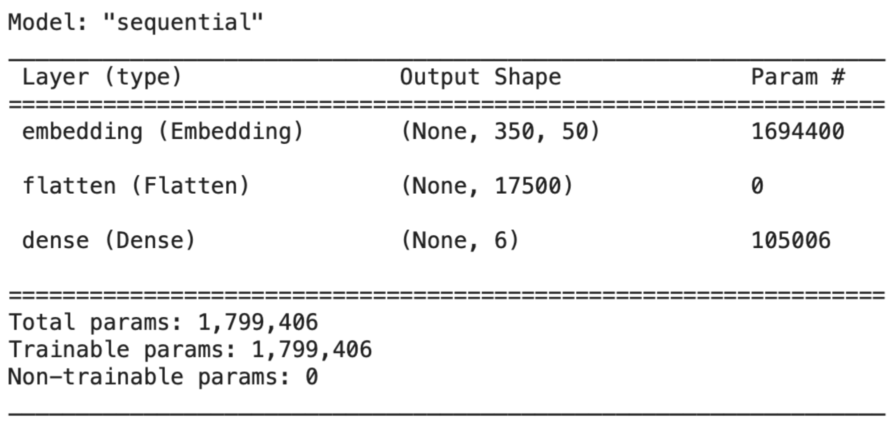
At this point we save the embedding matrix that we will use later in the chapter of Experiments [6].

5.2 Custom Word Embedding

Now we go to create our custom word embedding, to do so we start with the creation of the dictionary using all the tweets contained in the train set.

To build our network we leveraged Keras, we created a sequential model, and in the layer stack we embedded an embedding layer, a flatten one and finally a dense layer.

Below is shown the simple architecture used:



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 350, 50)           1694400

 flatten (Flatten)           (None, 17500)             0

 dense (Dense)               (None, 6)                 105006

=================================================================
Total params: 1,799,406
Trainable params: 1,799,406
Non-trainable params: 0
_________________________________________________________________
```

Figure 7: Custom word embeddings NN.

6 Experiments

In order to find the model that performs better on our dataset we tried different solutions with different hyperparameters. We tried LSTMs, CNNs, and BeRT.

In order to fight overfitting we used the **Early Stopping**. This technique will evaluate after each epoch of training the loss on validation set, if there is no improvement for a determinate number of epochs, the training will stop since the model is fitting too much to the training set. Then the model kept is the one obtained after the epoch with the best loss on validation set. The parameter that determine the number of epochs of no improvement before stopping the training is called patience, and in our models we will use a patience of 3.

6.1 Evaluation techniques

Each model trained will be tested on our test set, and we will plot all the statistics of this evaluation aside from the Confusion Matrix. Other interesting informations to plot are accuracy and history on training and validation set after each epoch. This will let us know interesting information on the training phase.

Besides **accuracy** and **loss** on the test set, there is another particular ratio where we will focus on: the **precision on not_cyberbullying** class. This index is defined as:

$$Precision\ on\ not\_cyberbullying = \frac{TP}{TP + FP}$$

where:

$$TP = Number\,of\,correctly\,classified\,as\,not\_cyberbullying$$

$$FP = Number\,of\,incorrectly\,classified\,as\,not\_cyberbullying$$

The importance of this index is to be charged to the consequences to which a misclassification by our model will lead to. In a scenario where our model will be used to automatically detects cyberbullying tweets in order to take some actions about it (for example obscuring the tweet or banning the user author of that tweet), it is a less serious mistake misclassify the type of offensive tweet (for example labelling a religion cyberbullying tweet as an age cyberbullying tweet) than classifying an offensive tweet as non offensive. In fact in the first case the platform that use our model could take in any case the actions to punish the user for an offensive tweet, but in the second case no action will be taken, underestimating the threaten of that tweet.

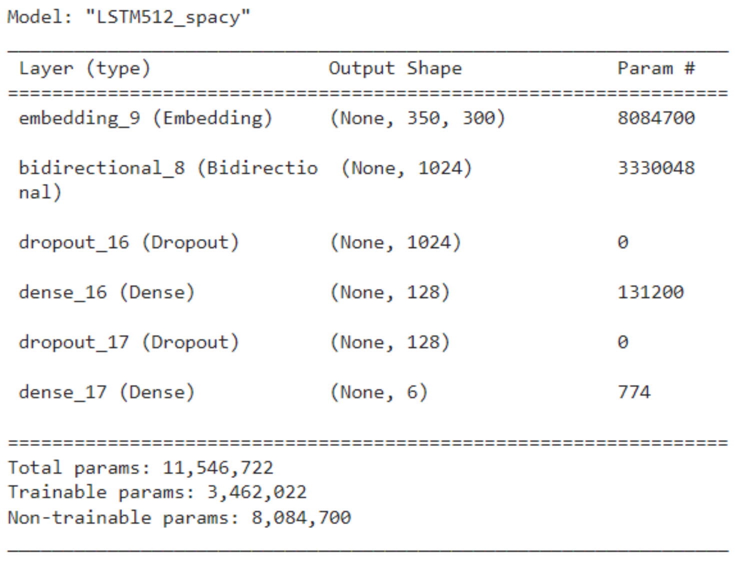
The precision on not_cyberbullying will be fundamental in evaluate the capability of a model to not cause this type of problems, and we will focus our attention on this ratio. In particular when we will do the ensemble of our models we will try to build the best model according to this parameter.

6.2 LSTM

Our first experiment was to build a model using LSTMs. We tried, for all the two embeddings three different configurations. The simplest one is with 64 LSTMs and a 30 Dense Neurons layer, then we increased the capacity of the network creating a model with 512 LSTMs, 128 Dense Neurons and a dropout layer. Finally we evaluated the effect of dropout proposing again the last model but decreasing the dropout rate from 0.5 to 0.2.

6.2.1 Spacy Embeddings

The model which performed better with the pretrained Spacy embedding is the one with 512 LSTMs, 128 Dense Neurons and a dropout layer with dropout rate 0.5. The training history of this model are shown below.



```
Model: "LSTM512_spacy"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_9 (Embedding)     (None, 350, 300)          8084700

 bidirectional_8 (Bidirectio  (None, 1024)             3330048
 nal)

 dropout_16 (Dropout)        (None, 1024)              0

 dense_16 (Dense)            (None, 128)               131200

 dropout_17 (Dropout)        (None, 128)               0

 dense_17 (Dense)            (None, 6)                 774

=================================================================
Total params: 11,546,722
Trainable params: 3,462,022
Non-trainable params: 8,084,700
_________________________________________________________________
```

Figure 8: Summary of the model

The results of the training are shown below.

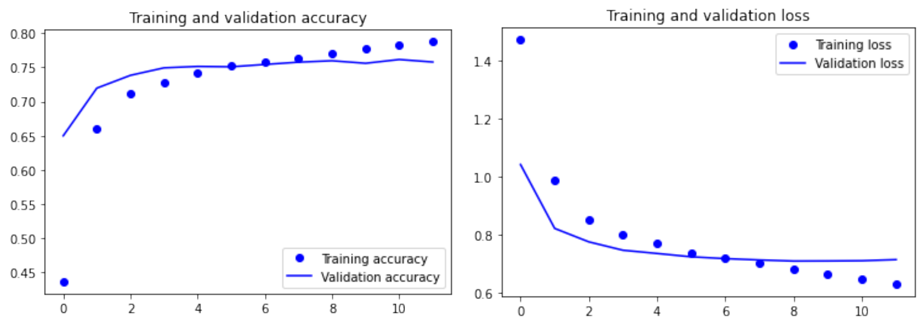
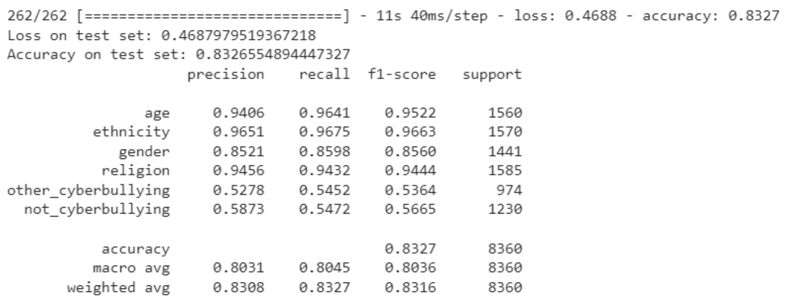
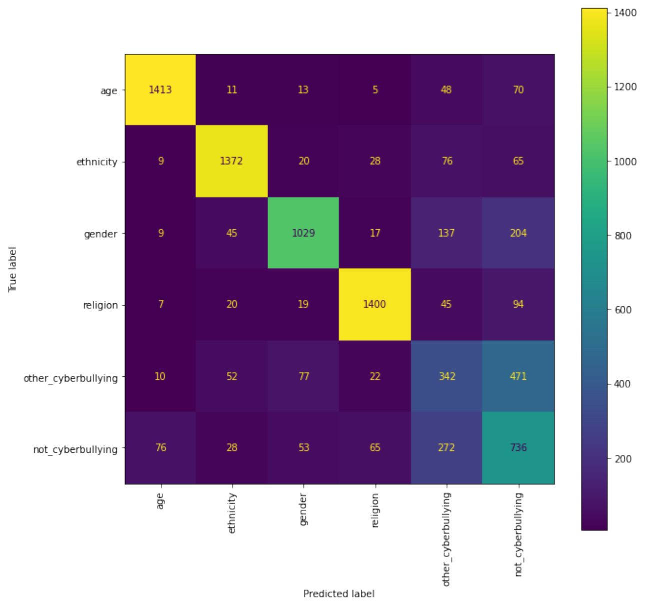


Figure 9: Accuracy and **Loss** on the training and validation set



```
262/262 [==============================] - 11s 40ms/step - loss: 0.4688 - accuracy: 0.8327
Loss on test set: 0.4687979519367218
Accuracy on test set: 0.8326554894447327
                     precision    recall  f1-score   support

                age     0.9406    0.9641    0.9522      1560
          ethnicity     0.9651    0.9675    0.9663      1570
             gender     0.8521    0.8598    0.8560      1441
           religion     0.9456    0.9432    0.9444      1585
  other_cyberbullying     0.5278    0.5452    0.5364       974
    not_cyberbullying     0.5873    0.5472    0.5665      1230

           accuracy                         0.8327      8360
          macro avg     0.8031    0.8045    0.8036      8360
       weighted avg     0.8308    0.8327    0.8316      8360
```

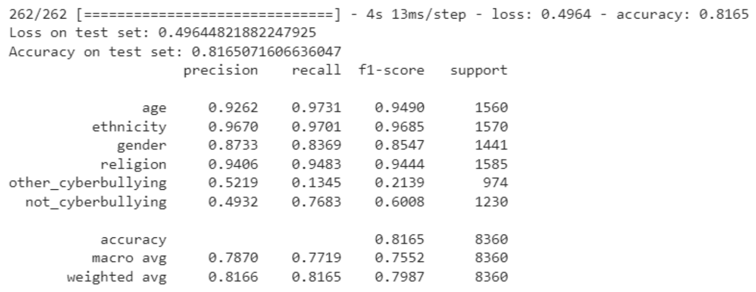


True label \ Predicted label	age	ethnicity	gender	religion	other_cyberbullying	not_cyberbullying
age	1413	11	13	5	48	70
ethnicity	9	1372	20	28	76	65
gender	9	45	1029	17	137	204
religion	7	20	19	1400	45	94
other_cyberbullying	10	52	77	22	342	471
not_cyberbullying	76	28	53	65	272	736

Figure 10: Evaluation of the model against the test set.

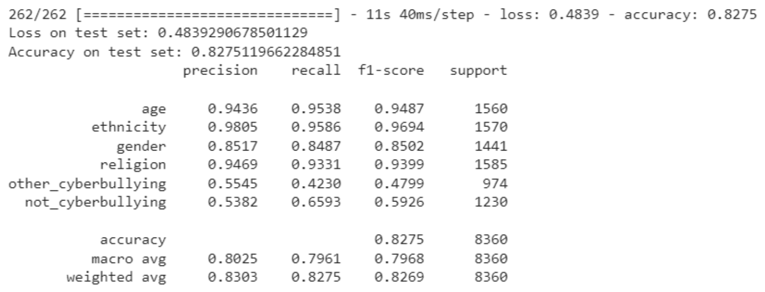
The results on the test set are better than the validation because there is no data augmentation, so the percentages of the dataset of other_cyberbullying tweets, which cause most of the problem, is lower.

While the results on the accuracy and loss are quite satisfying, the precision on not_cyberbullying is not good. From the confusion matrix we can notice how the most common error is labelling other cyberbullying as not cyberbullying. This error shows why some other utilizers of this dataset removed this category, since it is most difficult to recognize. Apart from this particular category, another error made by our model is labelling gender discrimination tweets as not_cyberbullying, demonstrating a greater difficulty in recognizing this category than the other. This could be due to the fact that most of the more used word in this category, as we can see from the word clouds, are less offensive than the ones in the other category, making the recognizing of these tweets more difficult.
The results of the other models are shown below.



```
262/262 [==============================] - 4s 13ms/step - loss: 0.4964 - accuracy: 0.8165
Loss on test set: 0.49644821882247925
Accuracy on test set: 0.8165071606636047
                     precision    recall  f1-score   support

                age     0.9262    0.9731    0.9490      1560
          ethnicity     0.9670    0.9701    0.9685      1570
             gender     0.8733    0.8369    0.8547      1441
           religion     0.9406    0.9483    0.9444      1585
other_cyberbullying     0.5219    0.1345    0.2139       974
  not_cyberbullying     0.4932    0.7683    0.6008      1230

           accuracy                         0.8165      8360
          macro avg     0.7870    0.7719    0.7552      8360
       weighted avg     0.8166    0.8165    0.7987      8360
```

Figure 11: Results of the model with 64 LSTMs and 30 Dense Neurons



```
262/262 [==============================] - 11s 40ms/step - loss: 0.4839 - accuracy: 0.8275
Loss on test set: 0.4839290678501129
Accuracy on test set: 0.8275119662284851
                     precision    recall  f1-score   support

                age     0.9436    0.9538    0.9487      1560
          ethnicity     0.9805    0.9586    0.9694      1570
             gender     0.8517    0.8487    0.8502      1441
           religion     0.9469    0.9331    0.9399      1585
other_cyberbullying     0.5545    0.4230    0.4799       974
  not_cyberbullying     0.5382    0.6593    0.5926      1230

           accuracy                         0.8275      8360
          macro avg     0.8025    0.7961    0.7968      8360
       weighted avg     0.8303    0.8275    0.8269      8360
```

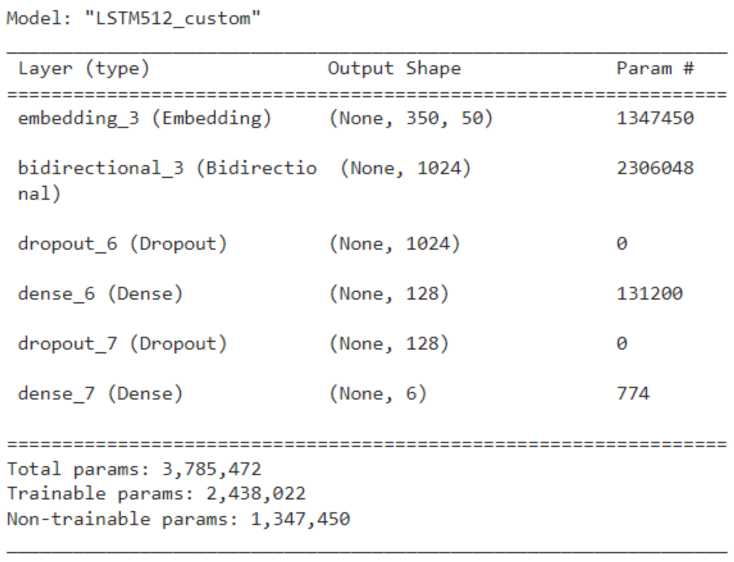
Figure 12: Results of the model with 512 LSTMs and 128 Dense Neurons with reduced dropout

As we can see the simpler model doesn't achieve the results of the more complex one. But since we use a complex model, the effect of the dropout is important in order to fight overfitting, and the model with the reduced dropout rate is worst.

An important advantage of the simpler model is the time of training: in addition of having results not so far from the complex model, each epoch of training took in average 12 seconds against the 42 seconds of the other two models. With a quite simple dataset as the one used the training time is not so relevant, but with a more massive database this model can let us saving a lot of time, achieving anyway good results.

6.2.2 Contextual Embeddings

The situation for the contextual embeddings is similar to the one seen before. The model which performs better is the one with 512 LSTMs and 128 Dense Neurons and a dropout rate of 0.5.



```
Model: "LSTM512_custom"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 350, 50)           1347450

 bidirectional_3 (Bidirectio  (None, 1024)             2306048
 nal)

 dropout_6 (Dropout)         (None, 1024)              0

 dense_6 (Dense)             (None, 128)               131200

 dropout_7 (Dropout)         (None, 128)               0

 dense_7 (Dense)             (None, 6)                 774

=================================================================
Total params: 3,785,472
Trainable params: 2,438,022
Non-trainable params: 1,347,450
_________________________________________________________________
```

Figure 13: Summary of the model

The results of the training are shown below.

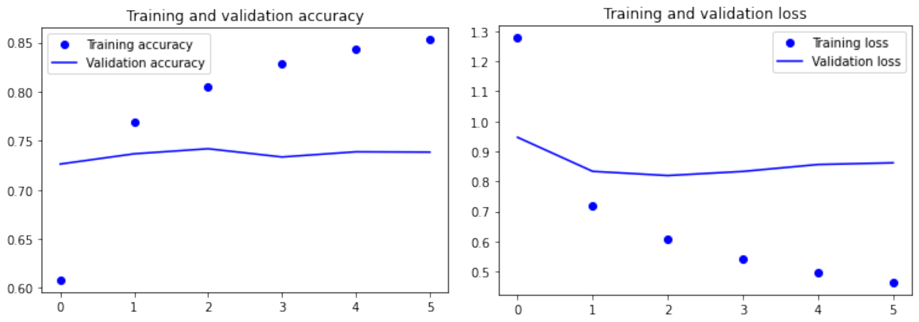
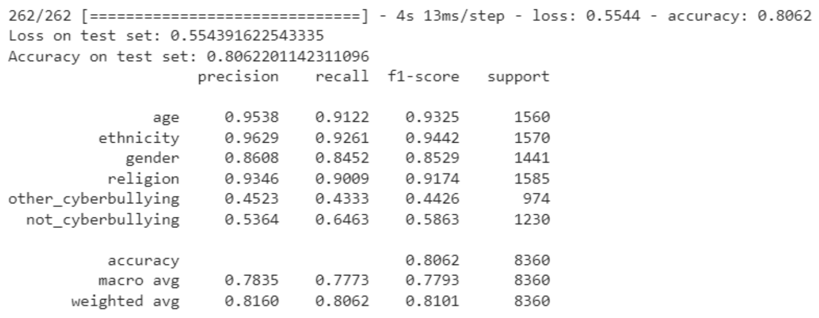


Figure 14: Accuracy and **Loss** on the training and validation set



```
262/262 [==============================] - 4s 13ms/step - loss: 0.5544 - accuracy: 0.8062
Loss on test set: 0.554391622543335
Accuracy on test set: 0.8062201142311096
                     precision    recall  f1-score   support

                age     0.9538    0.9122    0.9325      1560
          ethnicity     0.9629    0.9261    0.9442      1570
             gender     0.8608    0.8452    0.8529      1441
           religion     0.9346    0.9009    0.9174      1585
other_cyberbullying     0.4523    0.4333    0.4426       974
  not_cyberbullying     0.5364    0.6463    0.5863      1230

           accuracy                         0.8062      8360
          macro avg     0.7835    0.7773    0.7793      8360
       weighted avg     0.8160    0.8062    0.8101      8360
```

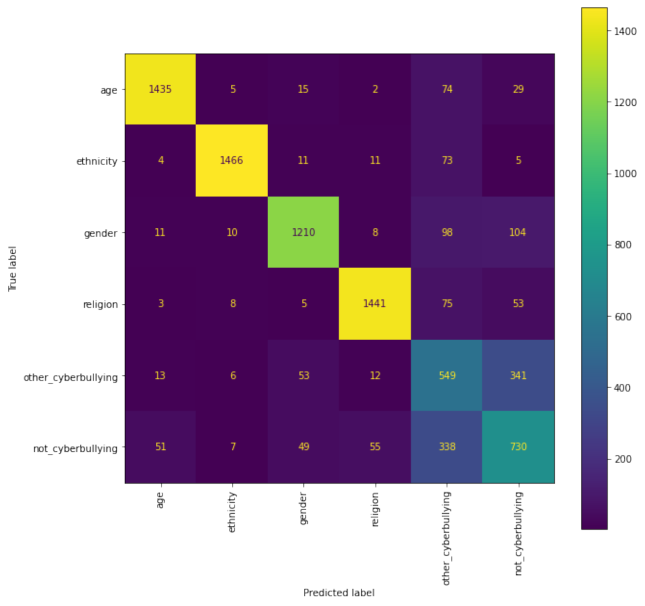
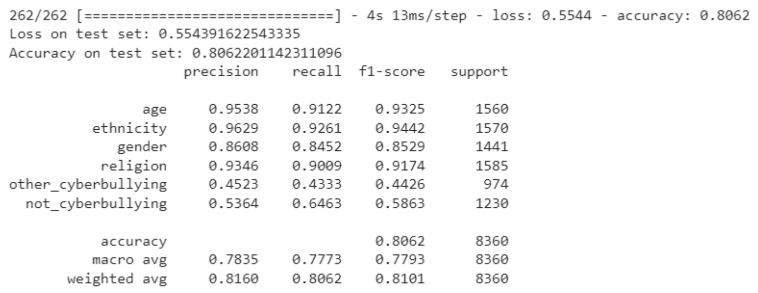


Figure 15: Evaluation of the model against the test set.

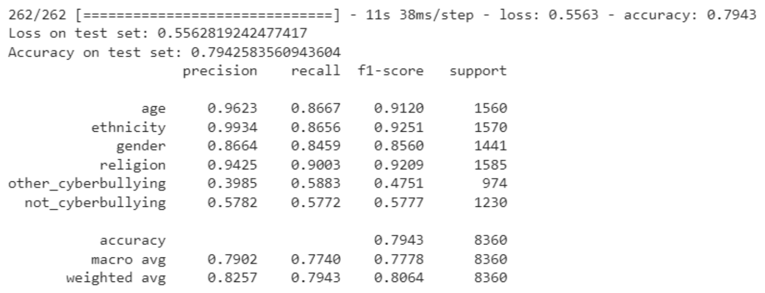
The results with custom embeddings are a little bit lower of the one with Spacy, both for the loss and accuracy and for precision on not cyberbullying. Aside from that we can see a slightly different confusion matrix, with about half of the errors of the custom model in misclassifying gender tweets as not cyberbullying.

The results of the other models are shown below.



```
262/262 [==============================] - 4s 13ms/step - loss: 0.5544 - accuracy: 0.8062
Loss on test set: 0.554391622543335
Accuracy on test set: 0.8062201142311096
                    precision    recall  f1-score   support

               age     0.9538    0.9122    0.9325      1560
         ethnicity     0.9629    0.9261    0.9442      1570
            gender     0.8608    0.8452    0.8529      1441
          religion     0.9346    0.9009    0.9174      1585
other_cyberbullying     0.4523    0.4333    0.4426       974
  not_cyberbullying     0.5364    0.6463    0.5863      1230

          accuracy                         0.8062      8360
         macro avg     0.7835    0.7773    0.7793      8360
      weighted avg     0.8160    0.8062    0.8101      8360
```

Figure 16: Results of the model with 64 LSTMs and 30 Dense Neurons



```
262/262 [==============================] - 11s 38ms/step - loss: 0.5563 - accuracy: 0.7943
Loss on test set: 0.5562819242477417
Accuracy on test set: 0.7942583560943604
                    precision    recall  f1-score   support

               age     0.9623    0.8667    0.9120      1560
         ethnicity     0.9934    0.8656    0.9251      1570
            gender     0.8664    0.8459    0.8560      1441
          religion     0.9425    0.9003    0.9209      1585
other_cyberbullying     0.3985    0.5883    0.4751       974
  not_cyberbullying     0.5782    0.5772    0.5777      1230

          accuracy                         0.7943      8360
         macro avg     0.7902    0.7740    0.7778      8360
      weighted avg     0.8257    0.7943    0.8064      8360
```

Figure 17: Results of the model with 512 LSTMs and 128 Dense Neurons with reduced dropout

6.3 CNN

The properties of Convolutional Neural Networks are the most suitable architecture to handle and analyze images, but we can also exploit them to extract knowledge from text data as well.

The idea of using a CNN to classify text was first presented in the paper Convolutional Neural Networks for Sentence Classification [8].

The central concept of this idea is to see our documents as images, to do that we create a matrix of numbers with the shape max_length of a tweet multiplied by the embedding size (350 in our case). This just explained is how the CNN sees a sentence in input, where in each row there is the word embedding associated to the word of that row.

Given this embedding matrix of a tweet we go to apply filters as wide as the representation of words and 3, 4 or 5 high depending on the number of words we want to take together (n-gram). Filters only slide vertically and not even horizontally as we are used to when processing images. For each filter we get a vector to which we apply a max pooling layer, concatenate the outputs of this layer for each filter and from it we get a single vector that will be used for classification.

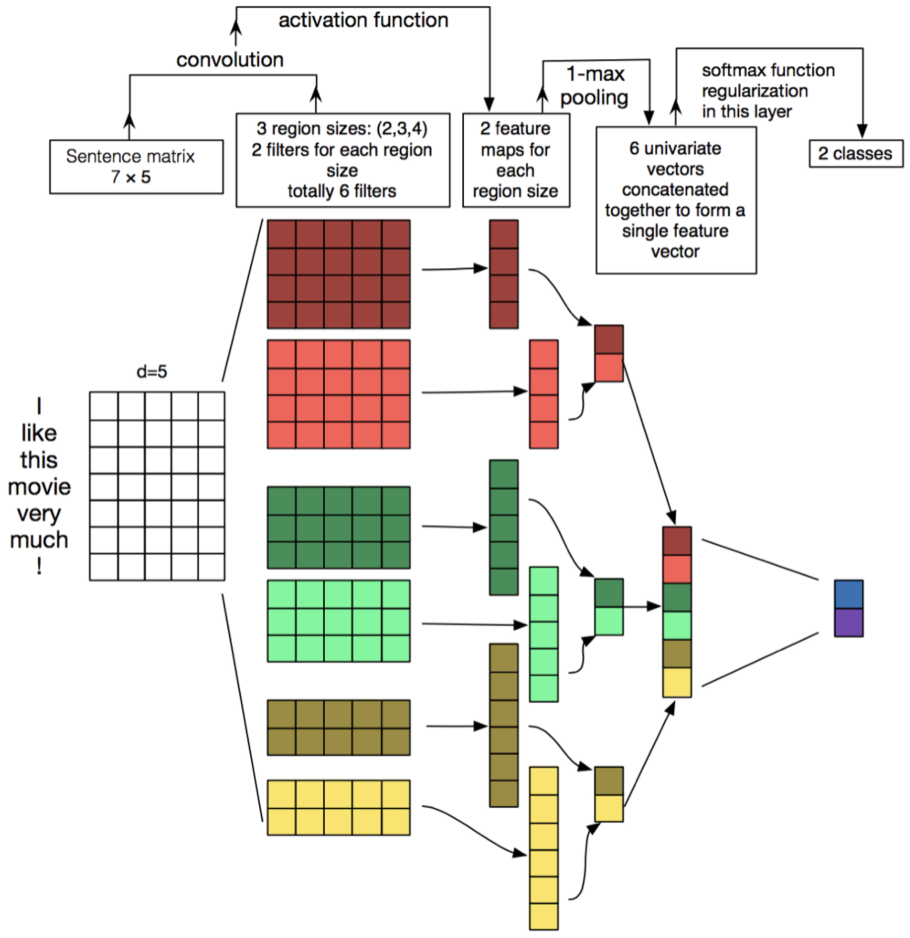
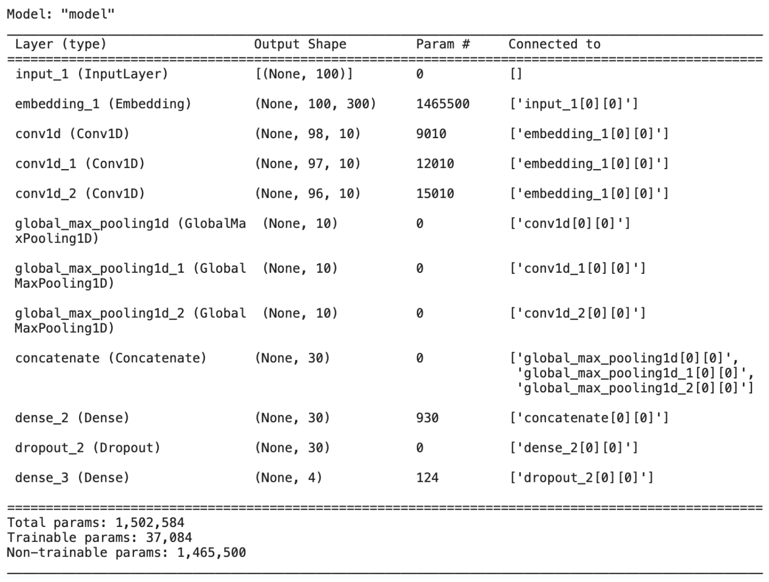

activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I like this movie very much !

Figure 18: CNN for text classification explained.

In other words, in each 1D-convolution with filter_size=n , the output is a concatenation of the convolution operator over all possible windows of n words. By varying n we regulate the size of such windows and focus on smaller or larger regions of the input sentence.

The downstream max-pooling layer extracts the most important feature for each convolution, independently of where it is located in the input sentence. Since each filter evaluates a particular elaboration of the embedding space, the max-pooling operation highlights the most relevant sequence of n vectors given that particular elaboration. Furthermore, max-pooling allows to considerably reduce the number of trainable parameters.

Below is shown the architecture of the CNN used:



```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 100)]        0           []

 embedding_1 (Embedding)        (None, 100, 300)     1465500     ['input_1[0][0]']

 conv1d (Conv1D)                (None, 98, 10)       9010        ['embedding_1[0][0]']

 conv1d_1 (Conv1D)              (None, 97, 10)       12010       ['embedding_1[0][0]']

 conv1d_2 (Conv1D)              (None, 96, 10)       15010       ['embedding_1[0][0]']

 global_max_pooling1d (GlobalMa  (None, 10)          0           ['conv1d[0][0]']
 xPooling1D)

 global_max_pooling1d_1 (Global  (None, 10)          0           ['conv1d_1[0][0]']
 MaxPooling1D)

 global_max_pooling1d_2 (Global  (None, 10)          0           ['conv1d_2[0][0]']
 MaxPooling1D)

 concatenate (Concatenate)      (None, 30)           0           ['global_max_pooling1d[0][0]',
                                                                  'global_max_pooling1d_1[0][0]',
                                                                  'global_max_pooling1d_2[0][0]']

 dense_2 (Dense)                (None, 30)           930         ['concatenate[0][0]']

 dropout_2 (Dropout)            (None, 30)           0           ['dense_2[0][0]']

 dense_3 (Dense)                (None, 4)            124         ['dropout_2[0][0]']

==================================================================================================
Total params: 1,502,584
Trainable params: 37,084
Non-trainable params: 1,465,500
__________________________________________________________________________________________________
```

Figure 19: CNN architecture.

6.3.1 Spacy Embeddings

Here is reported the accuracy and validation during the training process.

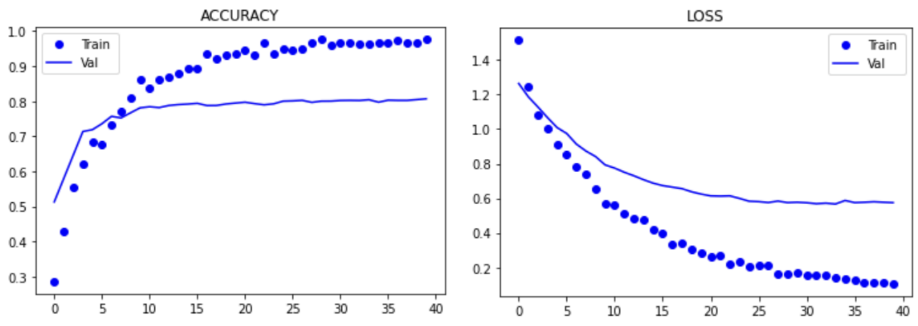
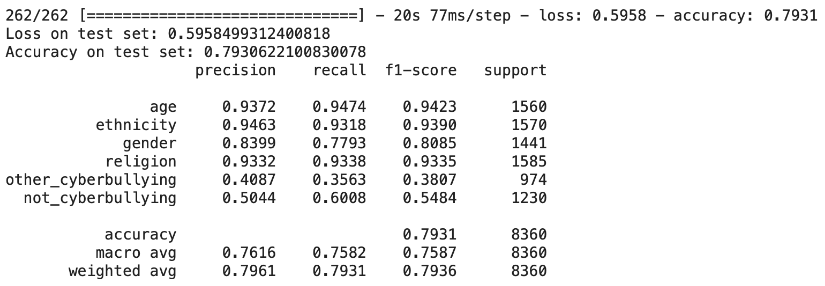


Figure 20: Accuracy and **Loss** on the training and validation set.

Below instead is reported the overall results and the confusion matrix.



```
262/262 [==============================] - 20s 77ms/step - loss: 0.5958 - accuracy: 0.7931
Loss on test set: 0.5958499312400818
Accuracy on test set: 0.7930622100830078
                     precision    recall  f1-score   support

                age     0.9372    0.9474    0.9423      1560
          ethnicity     0.9463    0.9318    0.9390      1570
             gender     0.8399    0.7793    0.8085      1441
           religion     0.9332    0.9338    0.9335      1585
other_cyberbullying     0.4087    0.3563    0.3807       974
  not_cyberbullying     0.5044    0.6008    0.5484      1230

           accuracy                         0.7931      8360
          macro avg     0.7616    0.7582    0.7587      8360
       weighted avg     0.7961    0.7931    0.7936      8360
```

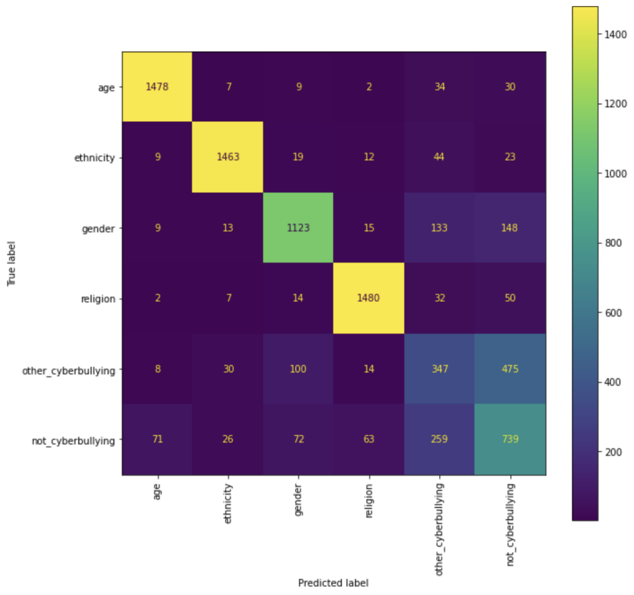


Figure 21: Overall results and **Confusion matrix**.

We obtain an overall accuracy on the test set equal to 0,79. From the results we can see that the precision, recall and f1-score are very good for the first 4 classes instead for classes *other_cyberbullying* and *not_cyberbullying* the results is not so good. As already said in the previous section someone who used this dataset deleted the class *other_cyberbullying* but in our opinion also this class must be included in the analysis.

We can see easily from the confusion matrix that the bad performances are due to the wrong prediction of the two classes *other_cyberbullying* and *not_cyberbullying*.

6.3.2 Contextual Embeddings

Now we report the result we obtain using again the CNN but with our contextual word embedding.

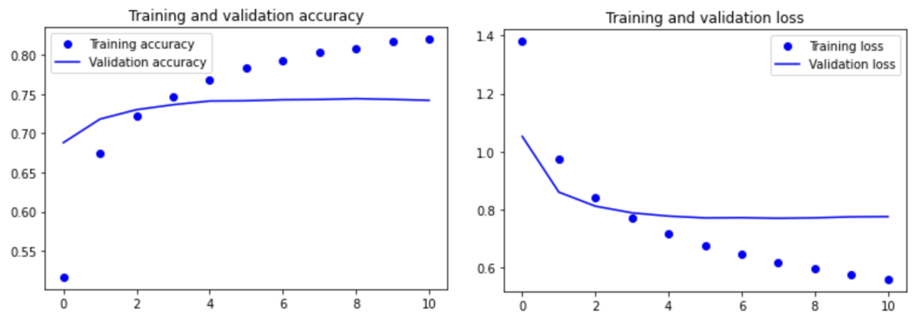
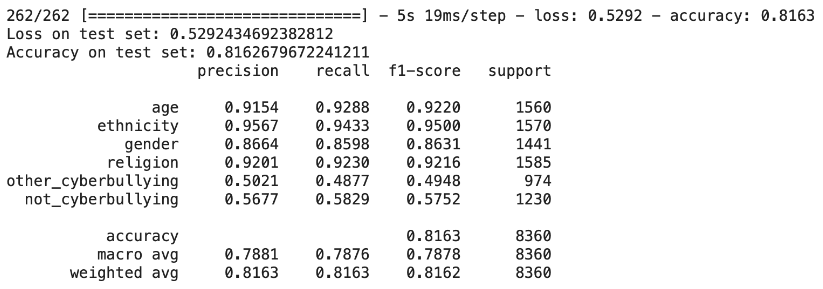


Figure 22: Accuracy and **Loss** on the training and validation set.



```
262/262 [==============================] - 5s 19ms/step - loss: 0.5292 - accuracy: 0.8163
Loss on test set: 0.5292434692382812
Accuracy on test set: 0.8162679672241211
                     precision    recall  f1-score   support

                age     0.9154    0.9288    0.9220      1560
          ethnicity     0.9567    0.9433    0.9500      1570
             gender     0.8664    0.8598    0.8631      1441
           religion     0.9201    0.9230    0.9216      1585
other_cyberbullying     0.5021    0.4877    0.4948       974
  not_cyberbullying     0.5677    0.5829    0.5752      1230

           accuracy                         0.8163      8360
          macro avg     0.7881    0.7876    0.7878      8360
       weighted avg     0.8163    0.8163    0.8162      8360
```

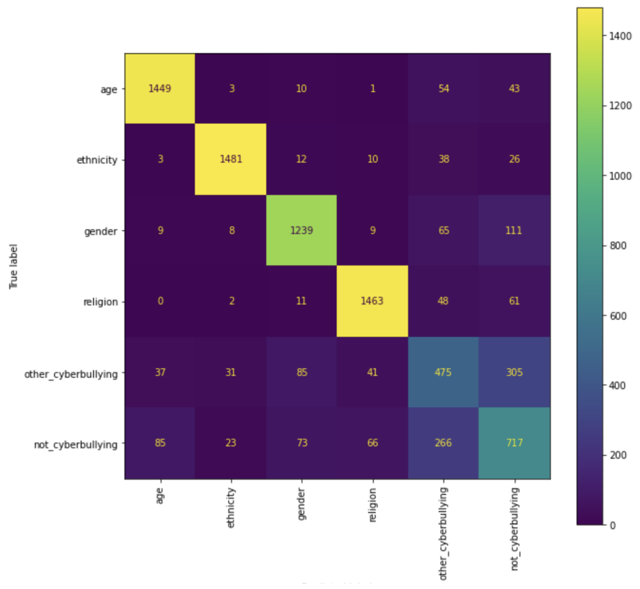


Figure 23: Overall results and **Confusion matrix**.

The results are slightly better respect to the spaCy embeddings, we obtain an increment of 0,2 in the overall accuracy. We have still some problems with the two classes $other\_cyberbullying$ and $not\_cyberbullying$ although we have an increment on the precision and recall on the $other\_cyberbullying$ class we have a little decrement on $not\_cyberbullying$.

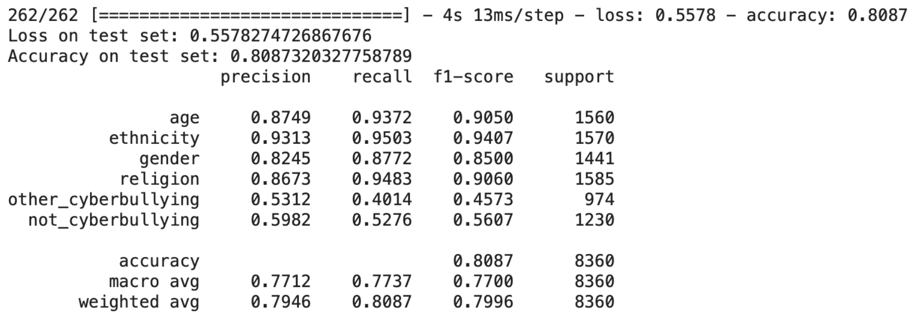
Despite this the overall performances of our custom embeddings are better.

We tried also another configuration of the convolutional neural network , that uses 20 filters instead of 10 and a dense layer with 45 neurons instead of 30.

We obtained an accuracy equal to 0,8145 that is slightly worst than the previous model but we had a little improvement of the precision on not_cyberbullying.

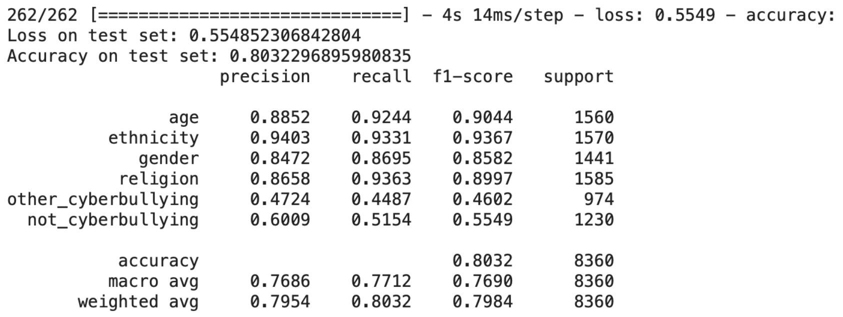
The last trial we made uses 50 filters and the dense layer has 45 neurons like before. The results with this trial are not so good, the accuracy is even worst than before and same happen for the precision on the not_cyberbullying class.

Below are reported the results of the two models:



```
262/262 [==============================] - 4s 13ms/step - loss: 0.5578 - accuracy: 0.8087
Loss on test set: 0.5578274726867676
Accuracy on test set: 0.8087320327758789
                     precision    recall  f1-score   support

                age     0.8749    0.9372    0.9050      1560
          ethnicity     0.9313    0.9503    0.9407      1570
             gender     0.8245    0.8772    0.8500      1441
           religion     0.8673    0.9483    0.9060      1585
other_cyberbullying     0.5312    0.4014    0.4573       974
  not_cyberbullying     0.5982    0.5276    0.5607      1230

           accuracy                         0.8087      8360
          macro avg     0.7712    0.7737    0.7700      8360
       weighted avg     0.7946    0.8087    0.7996      8360
```

Figure 24: Results of the model with 20 filters and dense layer with 45 neurons.



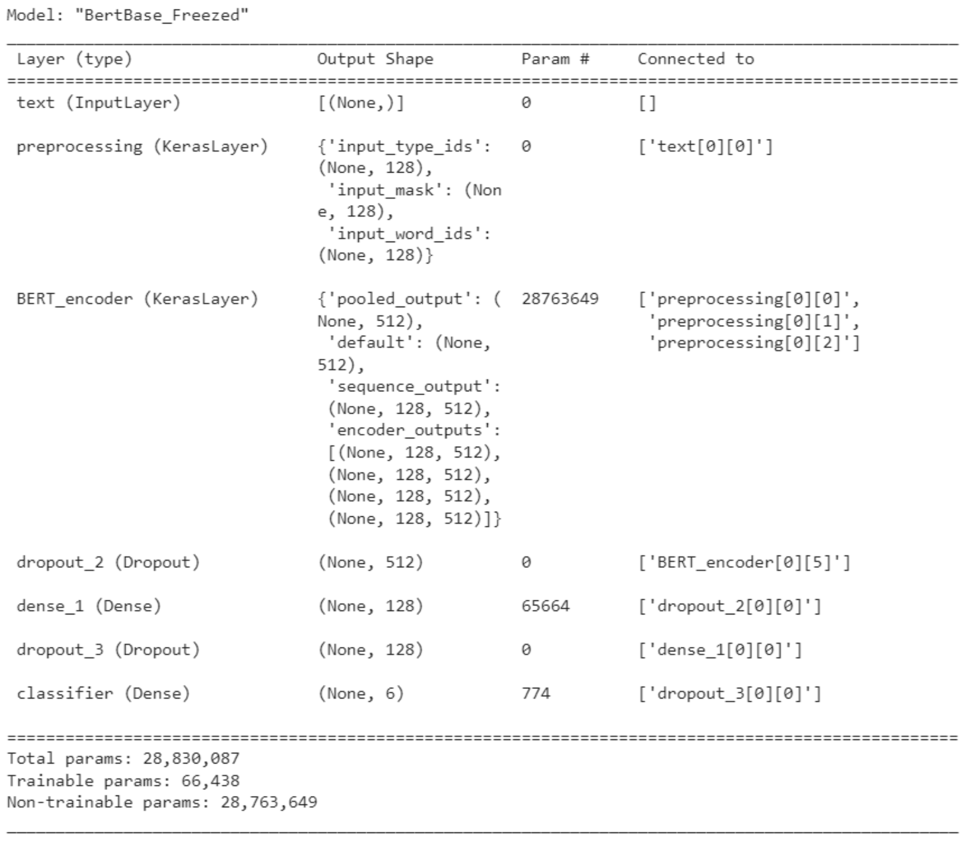
```
262/262 [==============================] - 4s 14ms/step - loss: 0.5549 - accuracy:
Loss on test set: 0.554852306842804
Accuracy on test set: 0.8032296895980835
                     precision    recall  f1-score   support

                age     0.8852    0.9244    0.9044      1560
          ethnicity     0.9403    0.9331    0.9367      1570
             gender     0.8472    0.8695    0.8582      1441
           religion     0.8658    0.9363    0.8997      1585
other_cyberbullying     0.4724    0.4487    0.4602       974
  not_cyberbullying     0.6009    0.5154    0.5549      1230

           accuracy                         0.8032      8360
          macro avg     0.7686    0.7712    0.7690      8360
       weighted avg     0.7954    0.8032    0.7984      8360
```

Figure 25: Results of the model with 50 filters and dense layer with 45 neurons.

An idea could be to tune the hyperparameter but the running time is to high and the free version of Colab can not do this heavy job.

6.4 BeRT

The State Of The Art model for Natural Language Processing is BeRT (Bidirectional Encoder Representations from Transformers), a transformer-based pre-trained model developed by Google. We built a model with a 128 MLPs network on top of a Base BeRT model, with a dropout rate of 0.1. Initially we kept the weight of BeRT freezed and we only trained the MLPs. Since the training time of this model is massive, and our computational power was low (for this experiments we used the free version of Colab), we used a smaller Bert model[7], which achieved satisfying results.



```
Model: "BertBase_Freezed"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to                     
==================================================================================================
 text (InputLayer)              [(None,)]            0           []                               
                                                                                                  
 preprocessing (KerasLayer)     {'input_type_ids':   0           ['text[0][0]']                   
                                (None, 128),                                                      
                                 'input_mask': (Non                                               
                                e, 128),                                                          
                                 'input_word_ids':                                                
                                (None, 128)}                                                      
                                                                                                  
 BERT_encoder (KerasLayer)      {'pooled_output': (  28763649    ['preprocessing[0][0]',          
                                None, 512),                       'preprocessing[0][1]',          
                                 'default': (None,                'preprocessing[0][2]']          
                                512),                                                             
                                 'sequence_output':                                               
                                 (None, 128, 512),                                                
                                 'encoder_outputs':                                               
                                 [(None, 128, 512),                                               
                                 (None, 128, 512),                                                
                                 (None, 128, 512),                                                
                                 (None, 128, 512)]}                                               
                                                                                                  
 dropout_2 (Dropout)            (None, 512)          0           ['BERT_encoder[0][5]']           
                                                                                                  
 dense_1 (Dense)                (None, 128)          65664       ['dropout_2[0][0]']              
                                                                                                  
 dropout_3 (Dropout)            (None, 128)          0           ['dense_1[0][0]']                
                                                                                                  
 classifier (Dense)             (None, 6)            774         ['dropout_3[0][0]']              
                                                                                                  
==================================================================================================
Total params: 28,830,087
Trainable params: 66,438
Non-trainable params: 28,763,649
__________________________________________________________________________________________________
```

Figure 26: Summary of the model

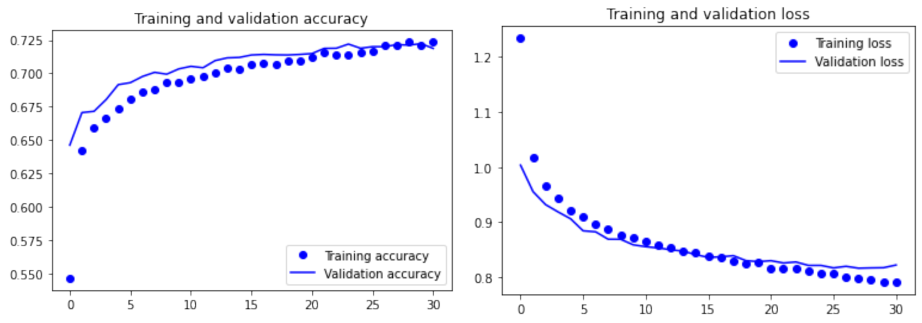
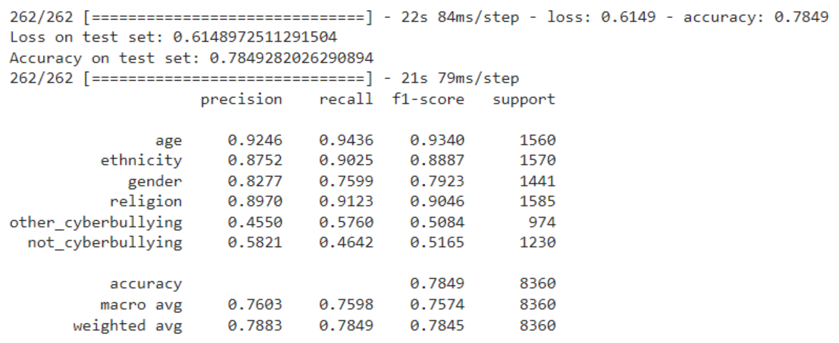


Figure 27: Accuracy and **Loss** on the training and validation set

As we can see it took a lot of time for the training to converge, considering also that each epoch lasted about 290 second, almost 7 time slower than the complex LSTMs models.



```
262/262 [==============================] - 22s 84ms/step - loss: 0.6149 - accuracy: 0.7849
Loss on test set: 0.6148972511291504
Accuracy on test set: 0.7849282026290894
262/262 [==============================] - 21s 79ms/step
                    precision    recall  f1-score   support

               age     0.9246    0.9436    0.9340      1560
         ethnicity     0.8752    0.9025    0.8887      1570
            gender     0.8277    0.7599    0.7923      1441
          religion     0.8970    0.9123    0.9046      1585
other_cyberbullying     0.4550    0.5760    0.5084       974
  not_cyberbullying     0.5821    0.4642    0.5165      1230

          accuracy                         0.7849      8360
         macro avg     0.7603    0.7598    0.7574      8360
      weighted avg     0.7883    0.7849    0.7845      8360
```

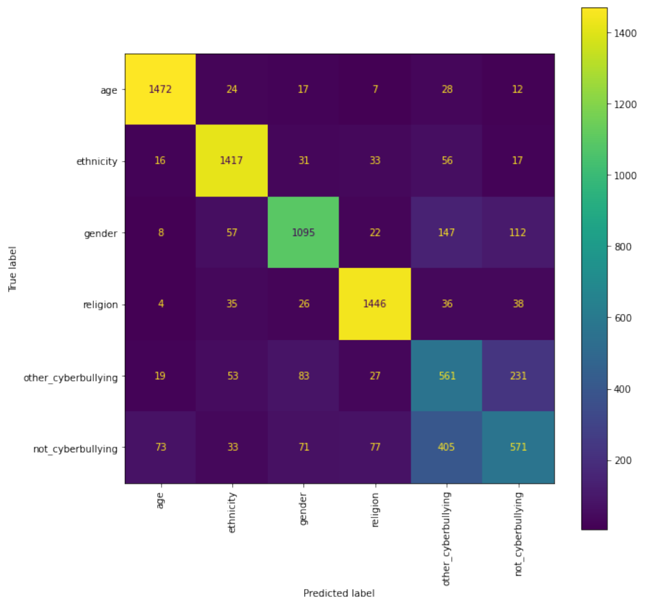


Figure 28: Evaluation of the model against the test set.

In this phase the results are a little bit lower than the models seen before, but unfreezing BeRT and fine-tuning the whole model the perfomance drastically increase.

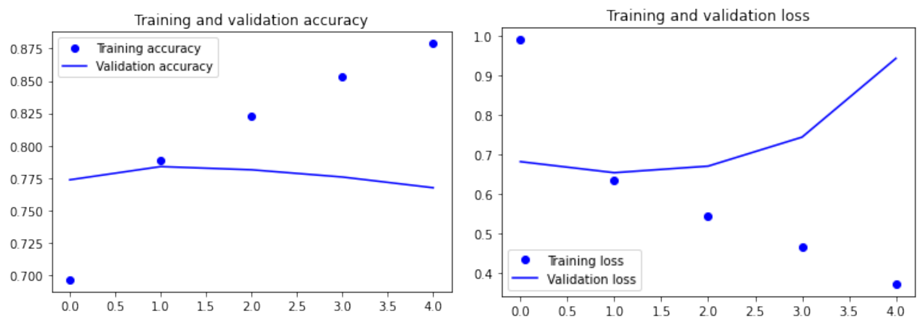
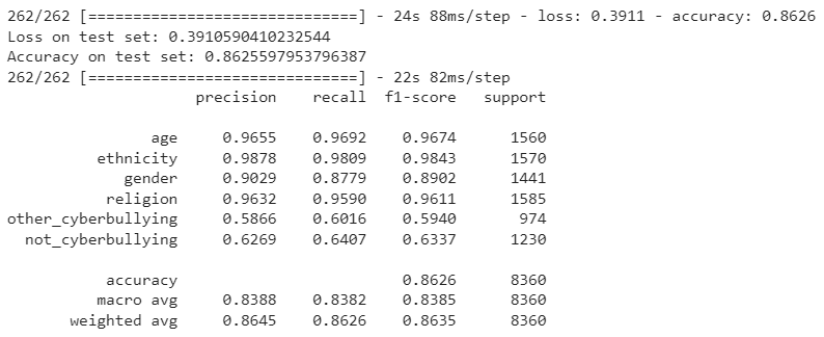


Figure 29: Accuracy and **Loss** on the training and validation set



```
262/262 [==============================] - 24s 88ms/step - loss: 0.3911 - accuracy: 0.8626
Loss on test set: 0.3910590410232544
Accuracy on test set: 0.8625597953796387
262/262 [==============================] - 22s 82ms/step
                     precision    recall  f1-score   support

                age     0.9655    0.9692    0.9674      1560
          ethnicity     0.9878    0.9809    0.9843      1570
             gender     0.9029    0.8779    0.8902      1441
           religion     0.9632    0.9590    0.9611      1585
other_cyberbullying     0.5866    0.6016    0.5940       974
  not_cyberbullying     0.6269    0.6407    0.6337      1230

           accuracy                         0.8626      8360
          macro avg     0.8388    0.8382    0.8385      8360
       weighted avg     0.8645    0.8626    0.8635      8360
```

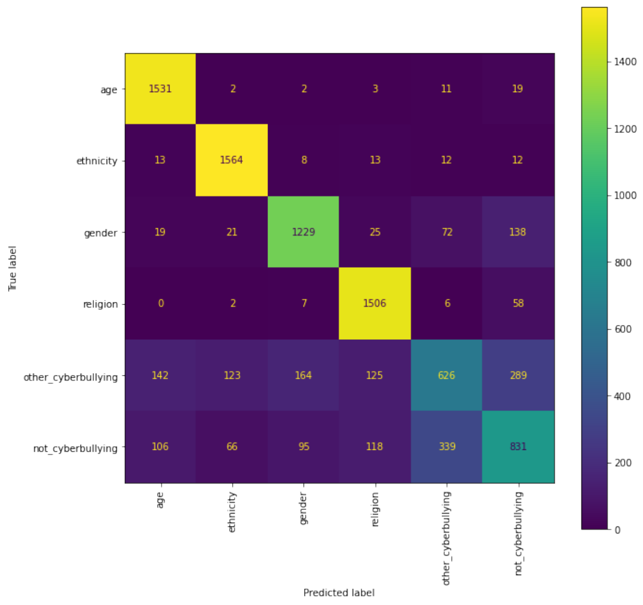
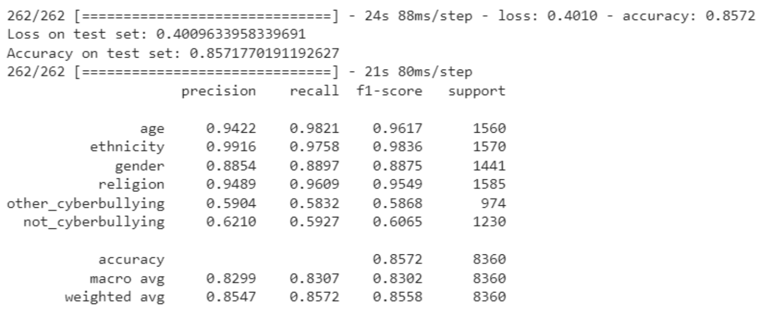


Figure 30: Evaluation of the model against the test set.

We can see how this model outperforms the results achieved before.

In the end we redone the whole procedure with a model with an increased dropout rate from 0.1 to 0.35, but the results slightly decreased.



```
262/262 [==============================] - 24s 88ms/step - loss: 0.4010 - accuracy: 0.8572
Loss on test set: 0.4009633958339691
Accuracy on test set: 0.8571770191192627
262/262 [==============================] - 21s 80ms/step
                     precision    recall  f1-score   support

                age     0.9422    0.9821    0.9617      1560
          ethnicity     0.9916    0.9758    0.9836      1570
             gender     0.8854    0.8897    0.8875      1441
           religion     0.9489    0.9609    0.9549      1585
other_cyberbullying     0.5904    0.5832    0.5868       974
  not_cyberbullying     0.6210    0.5927    0.6065      1230

           accuracy                         0.8572      8360
          macro avg     0.8299    0.8307    0.8302      8360
       weighted avg     0.8547    0.8572    0.8558      8360
```

Figure 31: Results of the model with increased dropout

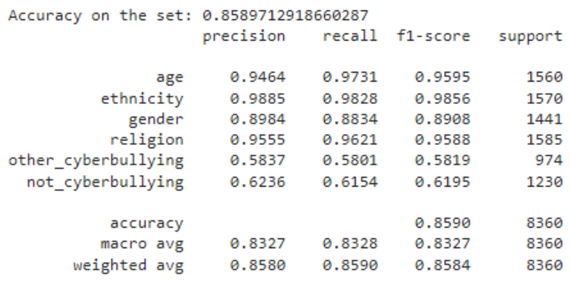
7 Ensemble

In the Experiments chapter we built models which achieved very satisfying results, in this chapter we are gonna combine the predictions of this models with the aim of improving the performances and the robustness of the ensembled classifier with respect to the single ones.

In particular we are gonna use the best model found for each type of network (LSTMs, CNNs and BeRT), with an exception for LSTMs models: since we spotted some relevant differences in the confusion matrices of the two best models with different embeddings, we are gonna keep all the two models with the hope that having more eterogenous predictions will make a positive contribution to the ensembled model.

7.1 Average model

The easiest way to aggregate the predictions of a set of classifiers is to average their predictions and to extract the ensemble classified label from this prediction.



```
Accuracy on the set: 0.8589712918660287
                     precision    recall  f1-score   support

                 age     0.9464    0.9731    0.9595      1560
           ethnicity     0.9885    0.9828    0.9856      1570
              gender     0.8984    0.8834    0.8908      1441
            religion     0.9555    0.9621    0.9588      1585
   other_cyberbullying     0.5837    0.5801    0.5819       974
     not_cyberbullying     0.6236    0.6154    0.6195      1230

            accuracy                         0.8590      8360
           macro avg     0.8327    0.8328    0.8327      8360
        weighted avg     0.8580    0.8590    0.8584      8360
```

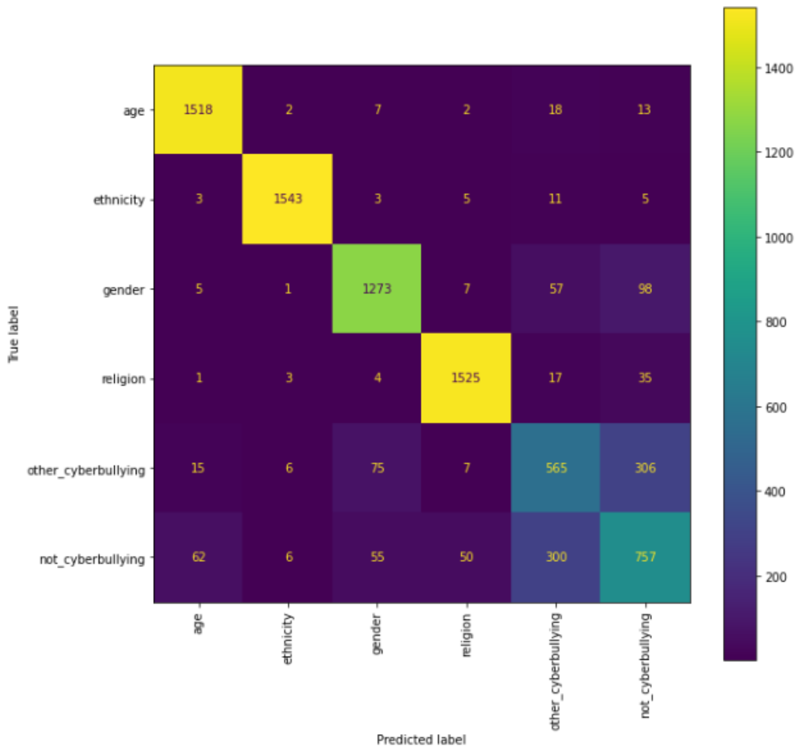


Figure 32: Average ensembled model results

The results of this ensembled model are better than the one obtained with LSTMs and CNNs, but not good as the better model obtained before, the one with BeRT. This is due to the fact that the BeRT model has performances too much better than the other models, and giving it the 25% of importance reduce the importance of the best model.

7.2 Weighted Average model

Another possibility to ensembling our models is to give a weight to each model prediction, in order to improve the importance of better models in the final result. Given 4 models and a preprocessed tweet, the prediction of the ensembled model will be:

$ensemble\_prediction = a * model1\_prediction + b * model2\_prediction + c * model3\_prediction + d * model4\_prediction$

where a, b, c, d, represent the weights associated to the models and the model predictions are vector with the element *i* representing the confidence with which the model believes the tweet belongs to class *i*. If the weights are all equal to $\frac{1}{4}$, we obtain the model equal to the Average Model seen in the last chapter.

We are gonna use two different techniques in order to find the best weights of the model. In all the two experiments we will build two models: one which will maximize the accuracy on the validation set and one that will maximize the precision on not_cyberbullying tweets.

One way to found a good set of weights is to try all the weights within a certain step (respecting the constraint that the sum of all the weights needs to be 1) and try all the combinations keeping only the models with the best results on the validation set. Since we have 4 model, by choosing a not so small step size (0.01), this approach is feasible in a moderate time, but since the algorithmic complexity of this solution is $O((\frac{1}{step})^n)$, where n is the number of models, this approach becomes unfeasible if the number of model is higher or we want an higher precision decreasing the step size.

```
Model with best accuracy:
Weights: [0.14, 0.17, 0.09, 0.6]
Accuracy on the set: 0.8632775119617225
                    precision    recall  f1-score   support

               age     0.9456    0.9814    0.9632      1560
         ethnicity     0.9936    0.9815    0.9875      1570
            gender     0.8933    0.8945    0.8939      1441
          religion     0.9573    0.9609    0.9591      1585
other_cyberbullying     0.5951    0.5975    0.5963       974
  not_cyberbullying     0.6375    0.6106    0.6238      1230

          accuracy                         0.8633      8360
         macro avg     0.8371    0.8377    0.8373      8360
      weighted avg     0.8616    0.8633    0.8623      8360
```

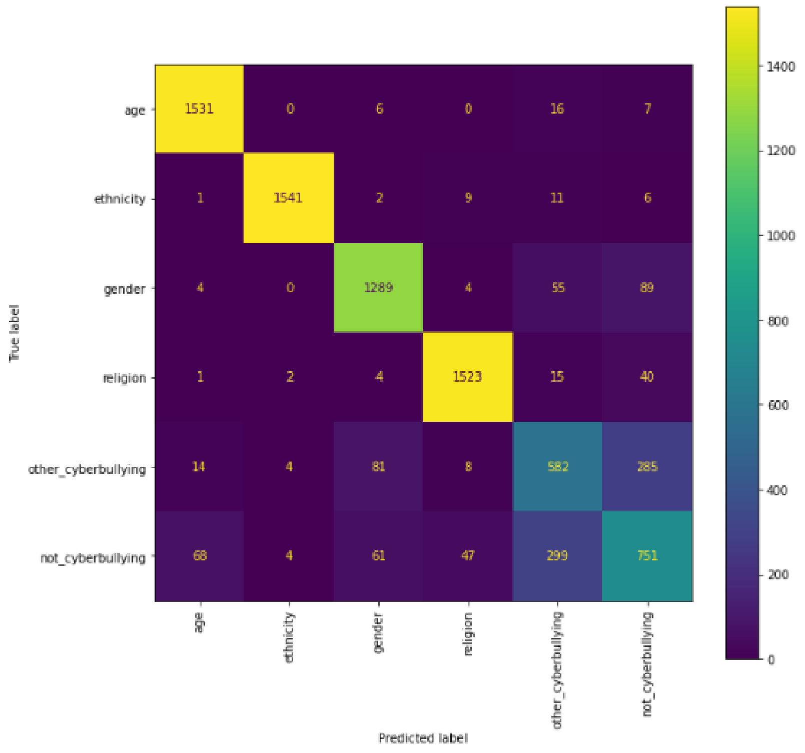
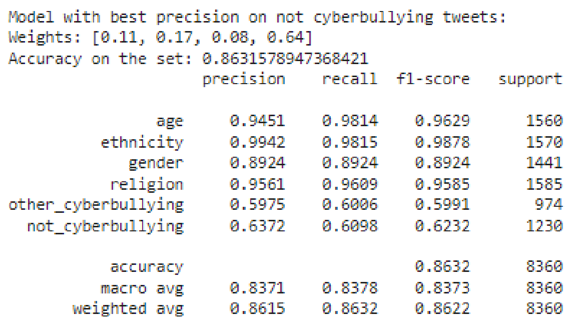


Figure 33: Best model found wrt **accuracy on validation set**



```
Model with best precision on not cyberbullying tweets:
Weights: [0.11, 0.17, 0.08, 0.64]
Accuracy on the set: 0.8631578947368421
                    precision    recall  f1-score   support

               age     0.9451    0.9814    0.9629      1560
         ethnicity     0.9942    0.9815    0.9878      1570
            gender     0.8924    0.8924    0.8924      1441
          religion     0.9561    0.9609    0.9585      1585
other_cyberbullying     0.5975    0.6006    0.5991       974
  not_cyberbullying     0.6372    0.6098    0.6232      1230

          accuracy                         0.8632      8360
         macro avg     0.8371    0.8378    0.8373      8360
      weighted avg     0.8615    0.8632    0.8622      8360
```

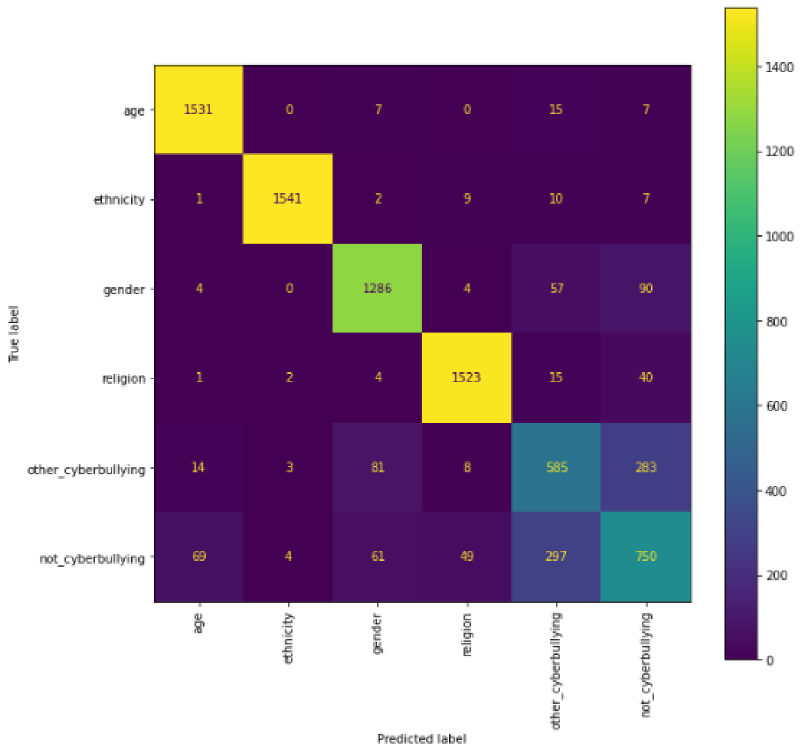


Figure 34: Best model found wrt **precision on not_cyberbullying**

The two models found has similar weights, this is index that maximize the precision on not_cyberbullying maximize also the accuracy of the model and viceversa, making our use of two different techniques of ensembling not so meaningful.

As we could expect, the model with a more relevant weight is BeRT, having the weight in the two model respectively equal to 0.6 and 0.64. Differently from the Average model, the results are better than BeRT, improving both accuracy and precision on not_cyberbullying.

To found more accurate weights we could decrease the step size, or implementing a genetic algorithm which could find the best weights in polynomial time.

8 Conclusions

In this report we analyzed the different techniques of Natural Processing. Languages applied to cyberbulying tweets. The results of our best models are summarized below.

Model	Word embeddings	Accuracy	Loss	Precision on not_cyberbullying
LSTM	spaCy	0,8327	0,4688	0,5873
LSTM	custom	0,8171	0,5192	0,5784
CNN	spaCy	0,7931	0,5958	0,5044
CNN	custom	0,8163	0,5292	0,5677
BeRT		0,8626	0,3911	0,6269
Ensemble		0,8633	-	0,6375

We observed an odd behaviour comparing the performances of SpaCy Embeddings with our customized one: LSTMs network worked better with SpaCy embeddings, while CNNs worked better with customized ones.

As we could expected, the solution which worked better was BeRT, the State Of The Art model for NLP problems, and to improve that result, we had to combine it with other models.

The final ensembled model achieved satisfying results regarding loss and accuracy. We could see how there are some types of cyberbullying tweets which our model recognizes better: the recall on age and ethnicity tweets is more than 98.1%, that means that less than 2% of these tweets are misclassified. This outcome is even more strong considering that for the the precision for ethnicity is about 99.4%, achieving an almost perfect detector for this type of cyberbullying tweets. We obtained also good results in recognizing religion discriminations, and for the gender cyberbullying, we obtained a small decrease of the performance, with recall and precision values near to 90%.

Most of the problems of our classifier comes from the other_cyberbullying class, which tend to be confused with the not_cyberbullying class and biasing the results for this category. This problem decrease the accuracy of out model and stop us from obtaining satisfying results for the precision on notumor. Despite this we think that keeping that class inside our dataset, differently from most of the solutions proposed by other users on Kaggle, was the right choice, since removing it it is just a trick for bypassing the problem without solving it, and if the model is going to be used in a real life context, the other 4 types of classes are not sufficient to describe all the cyberbullying types. Furthermore, keeping this class on the dataset still allowed us to achieve optimal results in recognizing other types of cyber-bullying.

References

[1] K. Dinakar, R. Reichart, and H. Lieberman, “Modeling the detection of textual cyberbullying,” Proc. Fifth International AAAI Conference on Weblogs and Social Media (SWM’ 11), 2011

[2] D. Yin, Z. Xue, L. Hong, and B. Davison, “Detection of harassment on Web 2.0,” in Proceedings of the Content Analysis in the WEB 2.0, 2009, vol. 2

[3] T. Davidson, D. Warmsley, M. Macy, and I. Weber, “Automated hate speech detection and the problem of offensive language,” in Proceedings of the Eleventh International AAAI Conference on Web and Social Media (ICWSM 2017), 2017

[4] `https://www.kaggle.com/datasets/andrewmvd/cyberbullying-classification`

[5] J. Wang, K. Fu, C.T. Lu, “SOSNet: A Graph Convolutional Network Approach to Fine-Grained Cyberbullying Detection,” Proceedings of the 2020 IEEE International Conference on Big Data (IEEE BigData 2020), December 10-13, 2020.

[6] `https://github.com/makcedward/nlpaug`

[7] Turc, Iulia and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina, "Well-Read Students Learn Better: On the Importance of Pre-training Compact Models", 2019

[8] Yoon Kim: Convolutional Neural Networks for Sentence Classification, 2014