Universitat Politécnica de Catalunya

Master's degree in Advanced Telecommunication Technologies

Active learning algorithms for multi topic classification

Guillem Bonafonte Pardàs

Supervised by:
M. Asunción Moreno Bilbao
Marta Ruiz Costa-Jussà

Abstract

In this master thesis we develop a model that surpasses previous studies to be able to detect cyberbullying and other disorders that are a common behaviour in teenagers. We analyze short sentences in social media with new techniques that haven't been studied in depth in language processing in order to be able to detect these problems.

Deep learning is nowadays the common approach for text analysis. However, struggling with dataset size is one of the most common problems. It is not optimal to dedicate thousands of hours to label data by humans every time we want to create a new model. Different techniques have been used over the years to solve or at least minimize this problem, for instance transfer learning or self-learning. One of the most known ways to solve this is by data augmentation.

In this thesis we make use of active learning and self-training to address having restrictions of labelled data. We have used data that has not been labeled to improve the performance of our models. The architecture of the model is composed of a Bert model plus a linear layer that projects the Bert sentence embedding into the number of classes we want to detect. We take advantage of this already functional model to label new data that we will use afterwards to create our final model. Using noise techniques we modify the data so the final model has to predict less structured data and learn from difficult scenarios.

Thanks to this technique we were able to improve the results in some of the classes, for instance the F-score modified increases by 7% for substance abuse (drugs, alcohol, etc) and 3% in disorders (anxiety, depression and distress) while keeping the performance of the other classes.

Acknowledgements

I would like to express my special thanks to my supervisors Marta Ruiz and Asuncion Moreno for their patience and giving me the strength for dealing with all the problems we found in this journey. Also to Montse Pardàs for her perseverance and trust. This whole project wouldn't exist without all your help and advice.

I've learnt much more than what I expected when I enrolled in this topic and I am grateful for all the knowledge acquired and all the process I've taken. Of course this would have been impossible if it weren't for my father Antonio Bonafonte, who proposed to me this topic and taught me uncountable hours of deep learning materia.

Revision history and approval record

Revision	Date	Purpose
0	**17/02/2021**	**Document creation**
1	**12/06/2021**	**Document revision**
2	**29/06/2012**	**Final draft**

Written by:		**Reviewed and approved by:**	
Date	29/06/2021	Date	01/07/2021
Name	Guillem Bonafonte Pardàs	Name	Marta Ruiz Costa-Jussà
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables

1. Introduction

1.1 Requirements and specifications

This project was born with an initial dataset of short sentences extracted from forums and web pages that were mainly used by young people. Each of these sentences was labeled according to 4 different topics: violence, substance, sex and disorders (which include anxiety, distress and depression). As we can see they are all related with behaviours that we want to prevent from young people, or at least detect to apply some measure to delicate situations. Each of this topic was labeled carefully by a group of professionals from 1 to 5 to get a more accurate degree on how strong this topic appears in the sentence and to have an initial dataset to work with.

The project goal is to achieve a final model that is able to classify these sentences and identify the presence of each topic in that sentence, all at once. Being able to detect if any of those topics is present in the sentence or not, with a notable confidence. We want to discover if there is some kind of toxic behaviour in a non-labeled sentence and which of the topics is the one that is being talked about.

Mobile World Capital created in 2017 a challenge to fight against cyberbullying [URL 3]. They wanted an application that could be installed at kids' phones (with parents consent) that would analyze the messages they receive in order to be aware if the kid is receiving messages that could lead to depression, anxiety or bullying and create an alert that will alert the parents about it.

The UPC participated in that project creating a database and introduced this topic in its research lines. With this project we want to improve the results obtained in the initial proposal that was developed at UPC.

1.2 Objectives

There are multiple models that we can generate in order to accomplish the same task. But sometimes it is not enough to create a really good model. Especially when you are dealing with a limited dataset. One of the main problems that we found when trying to solve this problem is the limited resources or labelled texts we have. Most of the classification problems need data labeled by people to get a decent model that can learn the problem specifically with good results. This data is expensive to generate, and even though a lot of texts were labeled for our case there isn't a big external labeled database that we could use as well combined with the initial data to have more samples or a simple way to create or extract more labeled data.

An additional and important problem is that most of the labelled data available is actually from correct behaviours. In other words, most of the data collected had all labels marked as negative in all the different topics, so it is hard for the model to learn efficiently how to identify them with a high precision.

The objective of this project is to improve the conflict detector system from text messages by including into the whole process non labeled data, which is easier to obtain.

So our main focus in this project was to find different ways to expand this data or to use the data we already have to improve the results of an initial model. The method that we decided to investigate is active learning. Active learning is a special case of machine learning where we use an external source to label our data with the purpose of keeping improving the results of the model. A typical way to achieve this is by using a first model (that is commonly called teacher model) to label unlabeled data, so we can extend the database. Using the extended database a posterior model, called student, can be trained to get better results. Most of the time we use human help to label data that the first model isn't certain about its content. In this way we can correct the behaviour of the first model by labeling the things it was not able to detect and the student will learn of these carences.

This technique has been successfully applied by many, all differing in the type of actively labeling this data and the type of the data we are trying to identify.

In our case our goal is to apply these methods to text data without any human help, only with the first created model and a huge amount of unlabeled data.

The labeled data created for the MWC challenge had already been exploited in a previous thesis "*Sentiment analysis on short Spanish and Catalan texts using contextual word embeddings*" [Cumalat 2020], where a first conflict detector from text was created. Our initial idea was to use their already built models as a starting point for the teacher model. We wanted to focus on the improvement of the dataset, but finally we decided that the results would be better if we created our own model from scratch, which I would have more control of and knowledge in how to improve it, as the performance of the teacher is really important to make the student successful. Even though I finally did not end up using their models, I took some of the ideas they already had proved to work and applied them to my own model and used the treatment that they applied to the initial dataset.

1.4. Work planning

For the work organization we used an adaptation of the agile methodology. We divided the whole project into 3 big tasks.

1. Create an initial model.
2. Optimize the model to get good results.
3. Apply the active learning techniques to create new models.

Each of these big tasks was chunked into smaller tasks that were assessed in story points so we could tackle how many tasks we could handle week by week. As we tied with a deadline here sprints were important to know if the timings were on track but there wasn't an implied requirement to use a professional tool (as Jira or Trello) as only one developer was working at the same time in this project.

1.5. Deviations

As already mentioned, finding the initial model was not one of the main purposes of the project, so when we finally decided that we wanted to build it ourselves, it came with some deviations to the project. These deviations affected in a way that further investigations in improving the data quality or the data augmentations were reduced. Instead we focused on an important part of the project in seeing which models worked for the existing data set and which of them gave results good enough to be set as a starting point for the other experiments.

Therefore, we could summarize the initial goal of our project as the following:

- Investigate about how active learning has been achieved and if there are any approaches at Natural Language Processing (NLP). Do experiments about them and try different approaches that could improve our initial model.

And after the initial research we concluded that it was important to add an additional point:

- Obtain a model with a decent accuracy as a starting point and that will be useful when comparing with all the experiments and will give us significant results.

2. State of the art of the technology used or applied in this thesis

2.1. Sentiment classification

One of the most common tasks in the NLP models is the sentiment classification [URL 1; Li 2018]. There are a lot of scenarios where businesses need to know the customer feedback or detect important messages in a sea of opinions. It is important then to be able to monitor these feelings, and NLP models are one of the most common ways [Genç 2019].

There are different types of sentiment classification. We could focus only on the polarity (positive, negative) but in our case we are especially interested in the feelings and emotions of the text. The emotion detection was typically done by lexicon systems [URL 1][Genç 2019] but this leads to errors as some words could express opposite emotions depending on context, that is what machine learning algorithms try to extrapole. More complex systems would involve using a combination in both techniques to get more trustful results and can depend also on the combination of the words in a sequence and not only on the words themselves.

The different machine learning algorithms that have been used to approach this problem are Naive Bayes [Rai 2017], Linear Regression [Singh 2019], SVM's [Reddy 2018] and deep learning (the one that we are using in this thesis).

Some problems we might face when training algorithms to detect emotions are the following [URL 1]:

- Irony or sarcasm, as it could not only don't detect the negative opinion but count it as a positive one.
- Emojis, as depending on the encoding and the people using it might differ a lot and they usually give a lot of information in the sentiment analysis.
- Feeding labeled data. As sentiment analysis is difficult to label even for humans, it is hard to have the same scale for all our labeled data.

2.2 Multi topic classification

Text classification problems usually are binary text classification [Li 2018] (filtering between spam or not, positive or negative sentiment, etc). There are a lot of tutorials about this kind of application and papers that will talk exhaustively about this topic. The objective in multi topic classification is to be able to classify our data in the different labels that we have for that problem. It can even be overlapped, as depending on the input data it is possible that one sample belongs not only to one class between the multiple ones that we have but to various of them at the same time.

There are some problems that we can face when dealing with multi topic classification [Li 2018]:

- Unbalanced classes: even though it is not exclusive of multi topic classification, it is more recurrent in this case as it is less probable having an equitative distribution of the data between more classes. Usually it is really hard to have a dataset that has enough data to classify in a symmetric way each of the topics we are trying to classify. In our case this was even more problematic as none of the classes had a significant representation.
- Biased results: as we are classifying multiple classes at the same time, we need to make sure that when training the model trying to predict one class don't interfere with the results of the other ones.

2.3 Active learning and self training

Deep learning models need a lot of data to be trained. Behind the paradigm of needing more and more, active learning searches to select the ones that could be more useful for the model. The basis of active learning is choosing which data we are willing to use and which one we are going to discard [Gildenblat 2020]. Usually the ones that are more interesting to use are the samples that usually are more difficult to classify. It is obvious that not all the data is useful when training models so getting rid of unnecessary (and not only unnecessary but painful when we are consuming GPU usage) data is a huge advantage.

The problem with deep learning is that neural networks aren't that good when telling they are not confident with the result, they are usually overconfident about the result they got. So in active learning we try to search for small cases where the model is certain that they don't know anything about the sample and try to use that in our favour.

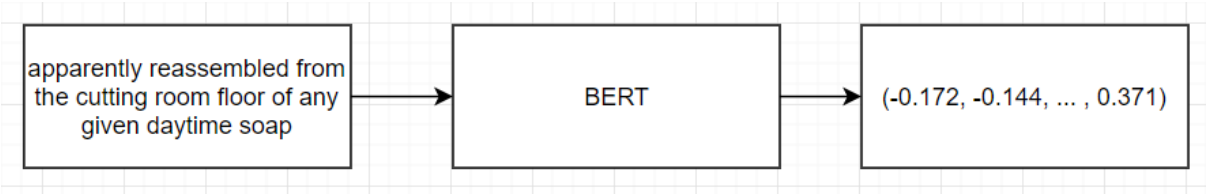
Self training on the other hand is very useful when we need larger amounts of data and data augmentation is not good enough [Lee 2017]. It is a semi supervised learning method where we'll use our trained model to train better models. The first trained model is called teacher and the ones that learn from the output of the first one are called students. In some cases the students can become the teachers of new students and create an iterative process to get better results. Thanks to active learning we need to be able to discard or select which data we are going to use in the following models by putting a threshold on the output of the first model. This is especially dangerous in the multiclass detection as one of the classes can be labeled perfectly while other be feeding missdirected data to the subsequent models.

Thanks to these techniques we can reduce the manual labeling, which is not only difficult but expensive, and only label the cases that we previously selected.

2.4 Bert

For this project we will take advantage of an already trained model, Bidirectional Encoder Representations from Transformers [Devin 2018]. The objective of this model is to learn how to represent each word within the sentence context, as a vector. It also generates a vector that represents the complete sentence. [Alammar]
In fact, to avoid huge vocabularies for all the possible words, modern NLP relies on subword tokens. Before Bert gets the input of the subwords we need the use of a tokenizer that will create a different token for each subword, and add a CLS token, that is the one that represents the sentence as a whole. This tokenized input tensor will be then passed into Bert, which is responsible for creating a new whole tensor that contains the representation of the meanings of each of the subwords, given the whole sentence, and the CLS vector that will give a contextual meaning of the sentences. For sentiment classification it is really helpful to have a model like this that represents the whole sentence as a vector (the sentence embedding) and that has already been trained for a similar purpose. At the end you only need to do a fine tuning to adapt it to your problem.

In the following image we can see an example of how we could use Bert to create embeddings for a sequence of words.



2.5 Related work

One of the papers that inspired this thesis was *"Self-training with Noisy Student improves ImageNet classification"* [Xie 2020] also clearly ìllustrated in a science blog by Devansh [Devansh 2020]. This research was able to get better performance models with less data. This is a very important fact, as labelling data is painful, and maintaining (or even improving!) the performance results is mandatory. The actual challenge was to apply their procedures to NLP instead of image classification. Obviously both problems are completely different but with the same objective.

Semi supervised learning has been used previously, but in this paper, according to what they explain in the paper, they do different things that hadn't been done before, and that we applied to our project too. Injecting different types of noise to the student, ensuring that the student learns more types of data distribution than the teacher was the first one. The other main difference is that even though the normal process is to iterate from student to teacher multiple times, in the paper they realize that the best result is achieved with one iteration and sometimes with an extra one, which is contra intuitive for the process as we are trying to iterate and get a constant performance improvement.

They apply two different types of noise: model noise and input noise. Model noise refers to noise directly applied to the model, dropout would be a clear example of model noise. Disturbing the process where we train the model by modifying the model itself. By input noise we mean modifying the data so it’s not exactly what it was at the beginning. Adding noise to the training data helps the student to avoid overfitting or just replicating the teacher's behaviour.

In the previous work with this data [Cumalat 2020] they tried to build a model able to predict the different labels by separate using supervised classification. We want to change these methods and use active learning to get results that we would not be able to get with supervised techniques.

3. Project development

In this chapter we are going to explain our proposal for using active learning and self training to try to improve the model results.

First, in section 3.1, we will explain how the database was distributed and why we had to use a different measure, followed by the data preprocessing that we applied. In section 3.3 we will give a detailed view on how we created the teacher model, called teacher. After that we will give a brief insight about why data augmentation was not feasible, and in sections 3.5 we can get the details about the whole process of creating the pseudo labels to use as an input for the student model, that it's explained in 3.6.

In the following figure we can see the steps we need to follow to obtain the final student model.

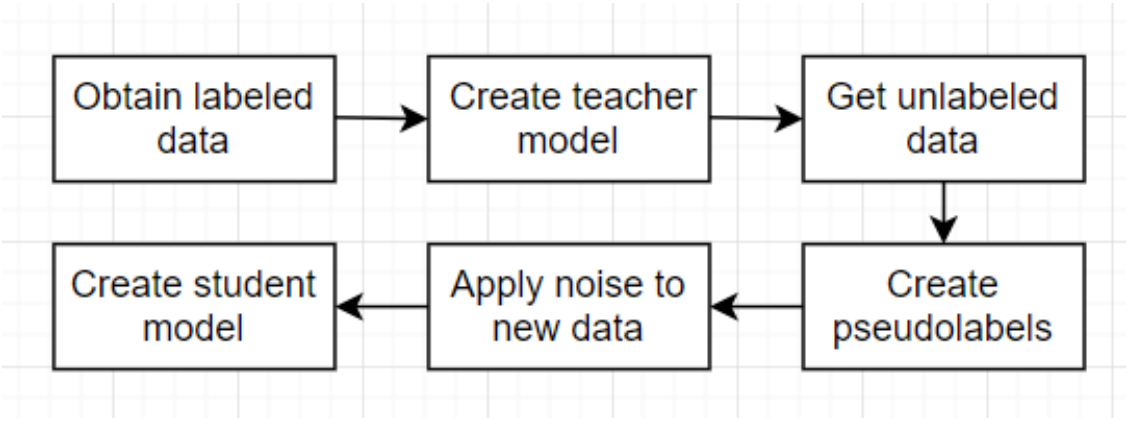


Figure 1: Different steps to get our student model.

3.1 Database

This database was created by the UPC as an investigation project [URL 3]. It consists of small texts extracted from forums, twitter and other similar sources whose objective is to detect conflictive messages in teenagers in different categories.

This database was already preprocessed by the thesis "*Sentiment analysis on short Spanish and Catalan texts using contextual word embeddings"* [Cumalat 2020]. The initial database that was manually labeled was tagged across 7 categories from 1 to 5. To optimize the results of the classification models it ended up being a binary classification (only 1 or 0) across 4 different categories (that grouped the 7 into subgroups). So from 7 categories labeled from a range of 1 to 5, in the end we had only 4 categories with 0 or 1 values.

The languages of these texts are Catalan and Spanish, but it is important to mention that they are not formal registers. As the texts were extracted from blogs of young people on the

internet, most of the texts had an invalid format that most of the language detectors wouldn't be able to recognize. The following table shows the distribution of the texts:

Catalan	Spanish	Violence	Substance	Sex	Disorders	No abuse
109141	111719	42227	6209	29621	43615	121747

Table 1: Summary of the data set classes and languages.

As we can see in the table, a post can have multiple categories (that's why the sum of all the posts doesn't match the sum of catalan and spanish cells). We grouped all the posts that didn't have any class present in "no abuse".

We can see that the imbalance of the class is huge. If we are trying to identify substance for example, we only have around 2.5% of the data as positive, which could lead our model to overfitting the negative answer. Classifying all the input as "non substance" would give our model a 97,5% of accuracy, that while true, is completely useless.

In order to get a better understanding of the results of the model, we will use the F-score that combines precision and recall in order to make our model more measurable and comparable. We will calculate a different F-score for each of the classes we are trying to predict as in the end our model behaves as if we had 4 different binary classifiers.

The formula to calculate this value is the following, with a beta value of 2 [URL 2]:

$$F_\beta = \frac{(1+\beta^2)\cdot \text{true positive}}{(1+\beta^2)\cdot \text{true positive} + \beta^2 \cdot \text{false negative} + \text{false positive}}.$$

The higher the beta value, the more importance we give at avoiding false negatives, this is why we need a beta of at least 2 to measure our problem.

We divided our data set in 3 groups, train, test and development, as usual. The distribution of these groups was 70%, 20% and 10%, respectively, randomly distributed. We used the same random seed in all the data distribution (39) in order to be able to replicate the experiments in the future without getting different results.

3.2 Data preprocessing

The first step was cleaning the data, we applied several data cleans in order to get a more consistent input for the model:

- Deleting punctuation marks.
- Lower case.
- Normalization: It eliminates characters that are repeated more than two times in the same word consecutively, as it is common in social media posts.

- Deleting stopwords.

With these steps we now have a more normalized input that we can use to create embeddings for computing the Bert embeddings.

3.3 Initial model

In this goal we describe a strong baseline that will be used later to compare the use of active learning and will be used at the same time as the teacher.

We started our model by using a fine tuning of the Bert model [Joshi, 2020]. The Bert model we used was the multilingual-uncased [LIB 1], as we needed to treat both spanish and catalan. The Bert model outputs a vector for each of the subwords and then a CLS vector that represents the whole sentence. It has been researched in different papers that for the majority of the problems using CLS leads to better results than using for example the mean of all the subwords vectors [Choi, 2021].

After doing all the data preprocessing we explained previously we will have a more normalized input that we can use to create embeddings to use as input to the Bert model. We apply the tokenizer associated with the Bert model that will divide the sentence into a sequence of tokens that represent subwords by different numbers.

With the Bert output we first considered using it in Support Vector Machines (SVM's), because we saw that these were the best results in the previous thesis. However, after trying it with different amounts of data, and considering all the training data we had, we realized that the training took exponential time, so it was not a feasible option for our purpose [Q&A 1]. SVM's don't allow GPU usage to train them and the amount of data augments the complexity of the problem too much, so using CPU for such a hard problem was not an option.

So finally we decided to go for another path with the Bert embeddings.

After applying a dropout regularization we use a linear layer (fully connected) that goes from the embedded dimension that Bert output of the sentence to the number of classes we are trying to predict (from 768 to 4). This will project the Bert output into our different 4 classes. We initialized the linear layer with the common xavier uniform initialization and the bias as zeros.

Before making any adjustments to the model, we went for a 0.1 dropout value and a learning rate of 2e-5. These are standard values that are used normally for NLP models and give a first good result before tuning these hyperparameters. For the batch size we went for the standard 32 sentences per batch.

For the loss function we used BCEWithLogitsLoss. This loss function combines a sigmoid layer followed by a BCELoss.

The baseline described in this section will be used also as the teacher model in our proposal of using active learning for multi topic classification.

3.4 Data augmentation

One of the initial ideas was to use data augmentation to check if the results of the first model improved only with this technique. In text processing it is harder to create data augmentation than in images, for example, so we proposed using a translator to duplicate the size of the data input. As we are using both catalan and spanish texts we could translate each text to the other language and use this total as the train input. This was feasible because the model we were using was multilingual so we could use both languages to train our model and it will still be targeting the data to be evaluated (as it is as well in both languages).

This experiment failed due to the format of the input text. We were not able to find a decent translator that could handle the format of the texts and provide significant translations. A huge percentage of the words have orthographic mistakes and repeated letters, or even words that don't even exist, as this data was extracted from teenagers' blogs. Most of the translations at the end were just keeping most of the words unchanged and changing the order of the sentences. This couldn't make any improvement to the model as we were just duplicating most of the sentences instead of creating new data.

3.5 Improving the model with unsupervised active learning

3.5.1 General scheme

We have followed the methodology proposed by Xie et al [Xie 2020], but applied to NLP instead of images. To achieve it we first needed to obtain an unlabeled database with a similar domain. With this dataset we will create new data for our next model by the creation of predicted labels (pseudolabels) by our teacher model. Once we have these new pseudolabels we will apply noise to them and use this new data as the training dataset all together with the previous data we used for the teacher model.

3.5.2 Creating pseudolabels

The additional database that we chose was "*HaterNet a system for detecting and analyzing hate speech in Twitter*". It has a similar domain, as they are spanish extracted tweets that were somehow related with agressions. The kind of jargon was not exactly the same, but for the purpose it was enough to train a student model that could give significant results.

The database consists of two million tweets, so this is about 10 times the size of our initial database. The main problem for this database was as well that in some of the topics that we were trying to identify there was not enough representation.

The process for creating pseudo labels was the following:

- Apply the bert tokenizer to map the sentences to the bert expected input.
- Pass the tokenized sentences to the model and retrieve the labels.
- Analyze the labels.
- Create a rule to decide which ones to keep.
- Map the label to a 0 or 1 value to match the initial database format.
- Create a new dataset with the selected ones and their respectively created label.

As the final activation function of the model is a sigmoid we used that value to discard the ones with low confidence so the model only learnt with sentences that were closely related to the topics we were analyzing. Given that the initial dataset was highly unbalanced and that we had a lot of data we decided to use only the ones that gave a positive value for any of the classes so the model could learn better for the topics we are interested in.

Finally we mapped the result of the teacher to a [0-1] value and stored the text with this label to have a clean self-labeled database, which was one of the initial problems to solve.

3.5.3 Creating student model

The final step was to create a student model that could overperform the teacher. In order to achieve that we needed to apply the two different types of noise so the student had to overcome the difficulties and still be able to learn how to make good predictions.

For the model noise we used dropout, and for the input noise we used word and character modifications as we will explain in the experiments section.

After applying the noise to the pseudo labeled database, we merged all the new data with the training data of the first dataset, as we don’t need any additional test or development data.

The model we used for creating the student was identical to the teacher one, except for the dropout value, which is a bit higher, because we want to focus on this new data set to achieve our objective.

4. Experiments

4.1 Working with unbalanced data

The initial model was not giving good results due to the unbalanced data, as we will see in the next figures. Most of the predictions were just asserting as negative all the classes, so while getting a good accuray the F-score modified was under 60%. To solve this problem we set up an experiment that could use this imbalance in our favour.

1. Calculate the number of instances of each of the classes: 42227 (violence), 6209 (substance), 29621 (sex), 43615 (disorders).
2. Introduce weights in the loss function: Using the *BCEWithLogitsLoss* function, this can be done with the pos_weight parameter, that is the % of instances that our dataset has in order to use this value to weigh more the true positives the less positives there are in the data set. It's possible to trade off recall and precision by adding weights to positive examples.

   For example, if a dataset contains 100 positive and 300 negative examples of a single class, then pos_weight for the class should be equal to 300/100=3

In our case the values used for each class was [4.23, 34.571, 6.4562, 4.0639].

This will force the model to adjust its parameters to give more importance to the positive outputs, the results were definitely helpful and gave us an initial version with a good F-score. We can see below the difference of the results with and without using the weight pos parameter for the baseline model that we will use as a teacher for future experiments.

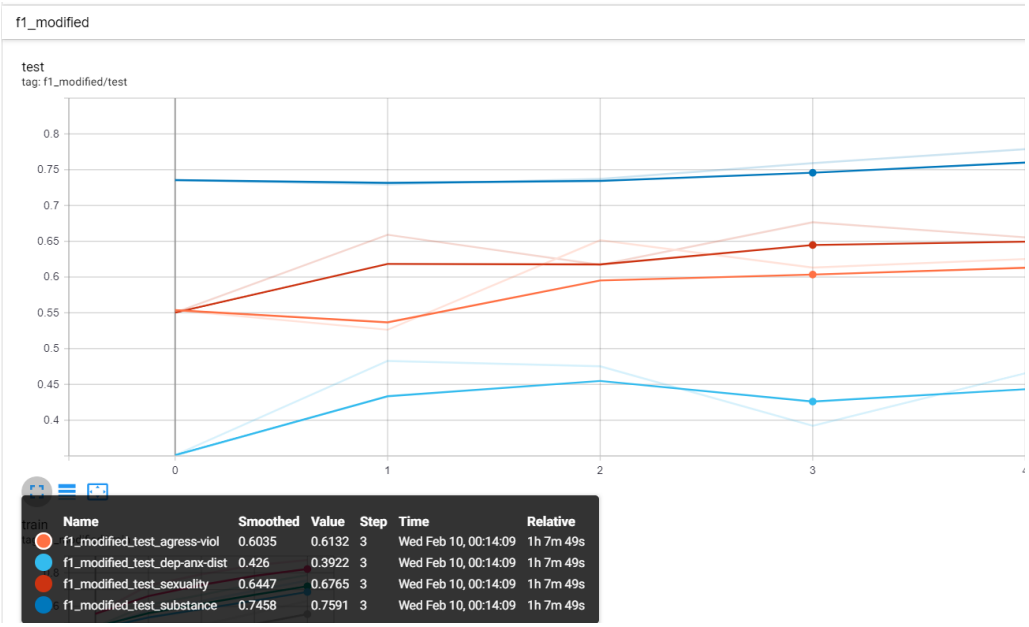


Figure 2: F-score modified value of the 4 classes without the pos_weight parameter.

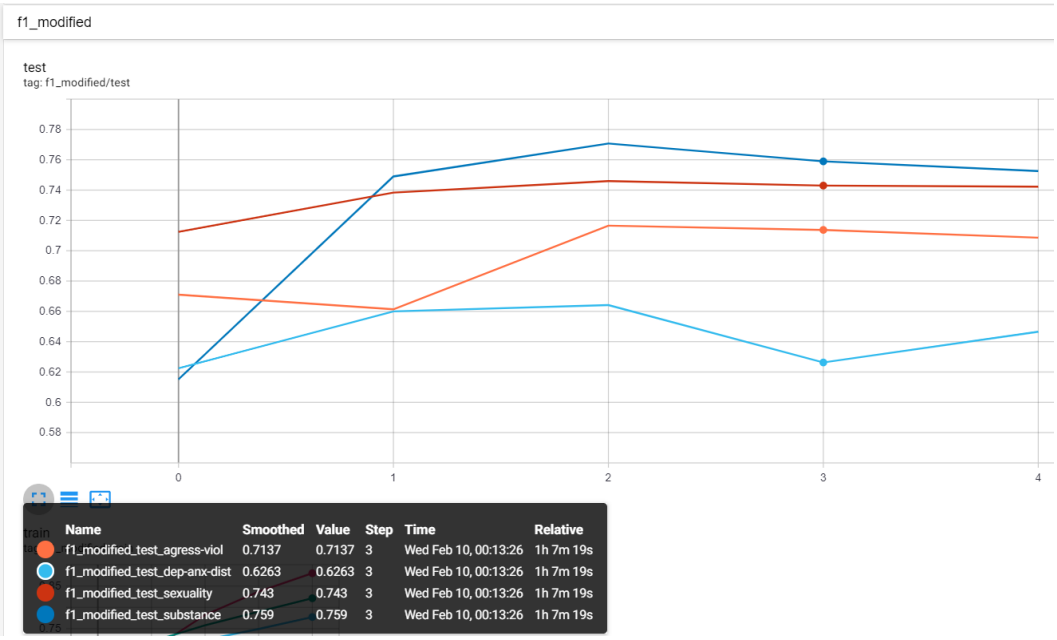


Figure 3: F-score modified value of the 4 classes using the pos_weight parameter.

We can observe a slight improvement in substance (the most unbalanced class, therefore the most difficult to learn from) and a huge improvement in all the other classes. In dep-anx-dist (disorders) we were able to improve by 0.2 the previous result (from 0.426 to 0.6263).

4.2 Choosing the best model

Before setting the teacher model, we wanted to test some parameters and choose which one would be our teacher. There were various things that were to be tested, and we used the *DEV* dataset to compare the results and decide which one was performing better. The experiments were done independently to make sure that the performance difference were made by that decisions and always based on the F-score results:

- Hyperparameters: Hyperparameter set up is a common thing in all the model's decisions. There are a few values that are usually used when we talk about learning rate and dropout. We tested the most common to choose which one was getting us better results.
- Model structure: The first approach was to use the bert model, a dropout layer and then one linear to map the embedded texts dimensions into the 4 different classes. To give the model more flexibility we added 4 hidden layers before the last step and got a bit better results than only with 1 linear layer. The [paper] used model noise by dropping some of the layers, as for image classification we need a huge quantity of layers. So adding additional hidden layers could help as well in the future to apply this kind of noise.
- Number of epochs: We used early stopping rules in order to check how many epochs were optimal to avoid overfitting of the training data. This experiment was carried without the pos_weight value. The final result was to use 4 epochs. As we can see in the following graph, for most of our classes the F-score modified value gets its better

value at the 4th epoch, so in general terms it does not make sense to keep training after the 4th epoch.

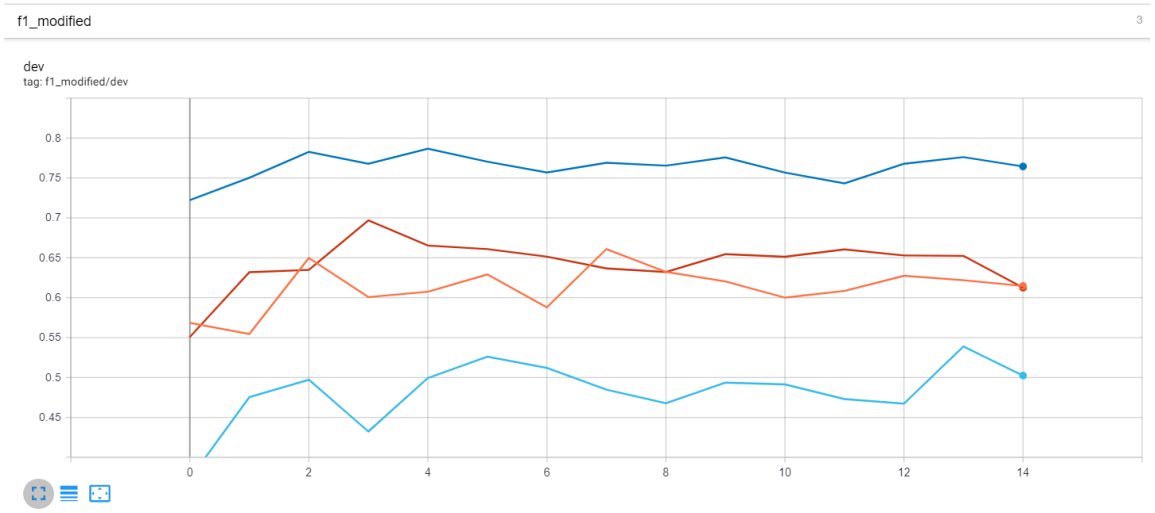


Figure 4: F-score modified for development of the 4 different classes respect the number of epochs.

In figure 3 we can see the F-score modified of the 14 epochs that we used to choose which was the earliest epoch where we could stop training.

Once we had all these parameters set, we used this version of the model as the teacher. Furthermore the teacher model is also the baseline model, that is the one that we will use to compare the results that we get with the student model, applying active learning.

4.3 Creating the student model

Once we had the teacher model, we used it to create pseudo labels of the external non-labeled database. We kept the initial texts with the new labels we created and decided a threshold to discard the low confidence ones and all that did not have presence of any of the classes, so we could get more instances of the unbalanced classes.

In the original paper they use two different types of noise: the model noise and the data noise. In our case we decided to go mainly for the data noise, as our model is not that big to apply stochastic depth, but we increased the dropout of the model from 0.1 to 0.3 to avoid overfit and create the model noise at the same time.

For the data noise, as we are not using images as in the paper, we had to create our own type of data noise. We did 3 different experiments, each of them using a pseudo-random function to apply noise to all the new data we just incorporated.

The following noises were applied:

- Random Augmentation: Swap some characters from the sentence for random similar ones.
- Delete and swap: Delete some random words and swap the order of some of the words in any place of the sentence.

- RandomAug, delete and swap: a combination of both previous noises.

After this we tokenized the sentences again to get the embeddings for Bert and trained a model from scratch with both datasets, the old one and the one we just created.

4.4 Results

For each of the different noises we trained a different model. Even though previous results showed that 4 epochs should be enough to get the maximum F-score, we trained a total of 14 epochs each one to see the behaviour of each of the models.

It is difficult to choose which epoch is the best one to stop your model, as not all the different classes reach their maximum at the same point. For each of the noises we can present the following results, even though in some classes we could get better results in a previous or later epoch, we kept the model version that had a better performance overall.

In figure 5 we can see the model that had the Random Augmentation applied in its data. We kept the version of the 8th epoch as the substance class reaches its maximum in this epoch. The maximum value achieved through all models for this class was with this noise applied, with a value of 0.7946.

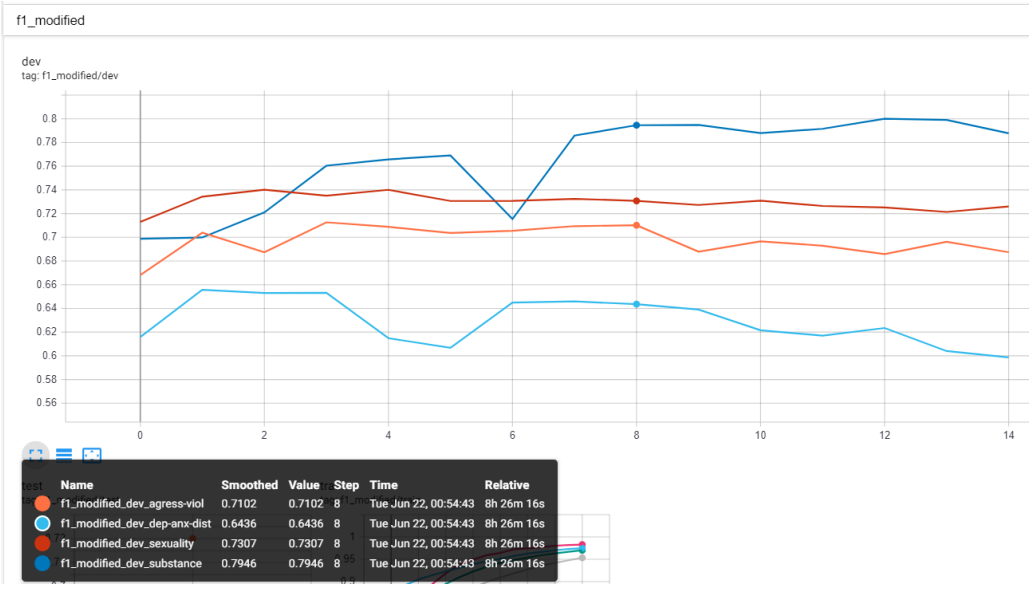


Figure 5: F-score modified for the student that had noise 1 applied.

Figure 6, for instance, the one with word deletion and word swap was the one that performed the best in the dep-anx-dist category. This category was the one that was more present in the second dataset we incorporated, so improving results in this category is significant in order to determine which noise is performing better.

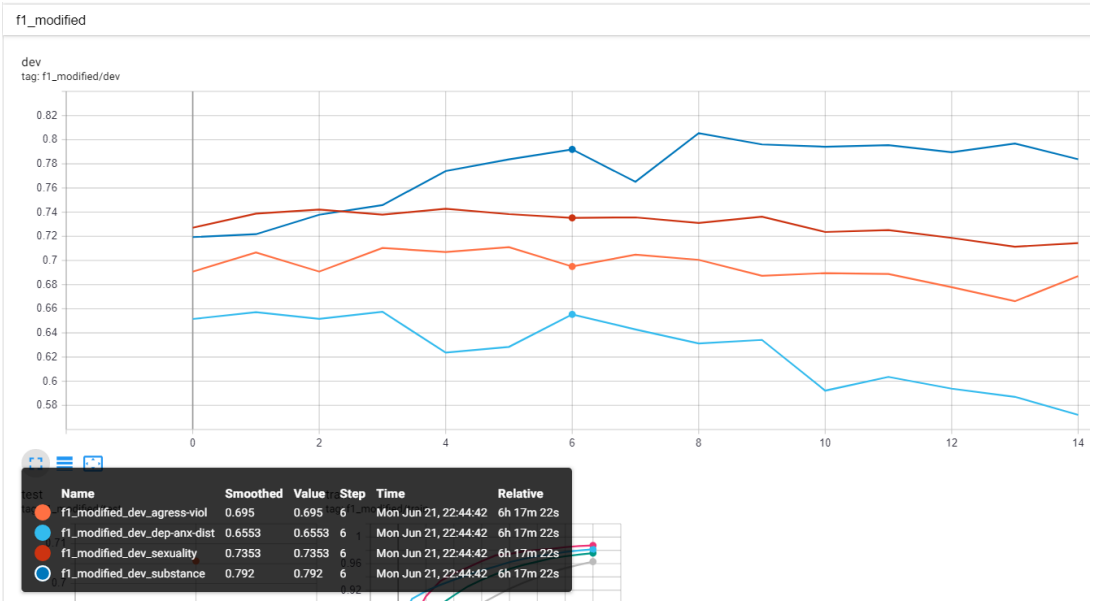


Figure 6: F-score modified for the student that had noise 2 applied.

Figure 7 was the one that had both noises applied. It is interesting to highlight that even though it didn't get the best results in any of the classes, all the classes had an F-score very similar to the best performant one and that it is the one that needed less epochs to get good results.

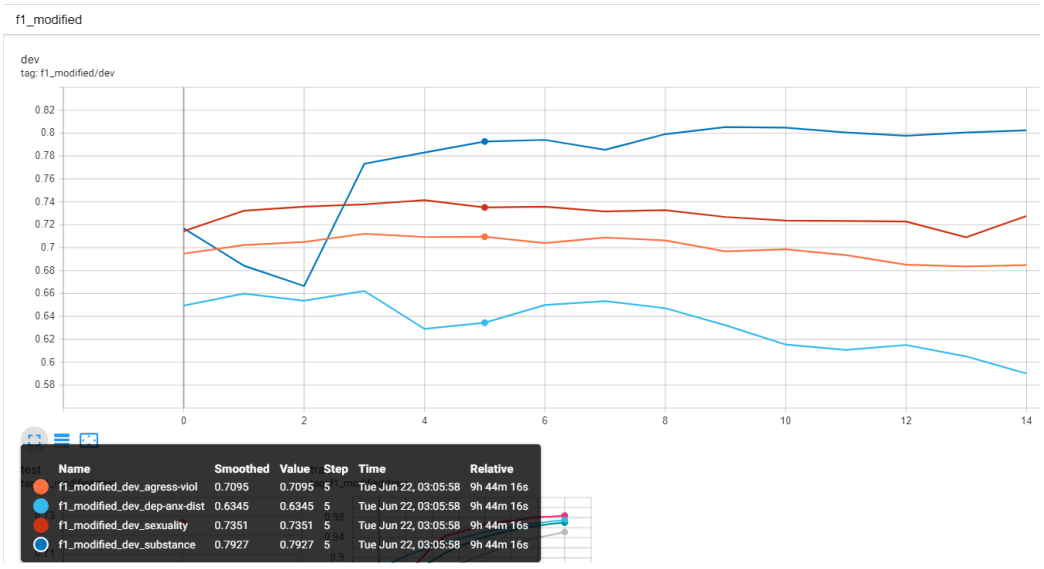


Figure 7: F-score modified for the student that had noise 3 applied.

	Agress-viol	Dep-anx-dist	Sexuality	Substance
Teacher (epoch 3)	**0.7137**	0.6263	**0.743**	0.759
Noise 1 (epoch 8)	0.7102	0.6436	0.7307	**0.7946**
Noise 2 (epoch 6)	0.695	**0.6553**	0.7353	0.792
Noise 3 (epoch 5)	0.7095	0.6345	0.7351	0.7927

Table 2: Summary of the 3 student models compared with the baseline model.

As we can observe at the table the only category that improved significantly was substance, and we can see that all the students behaved in a really similar way, which could mean that all the noises were actually very similar between them.

5. Budget

There are different costs that we have to take into account when developing all this project:

- Labeling dataset: As we need an initial labeled dataset, we need people to label it. We assume that they will do 30 labels per hour, and we needed 200.000 samples. So this would make a total of 50.000€. However this dataset was already available when we started the project.
- Data scientist: a 3 months project for a junior data scientist part time would be 3.000€.
- Software developer: we need one senior developer to carry out all the project development and coordinate the results, 15.000€.
- Servers and computers: we need 2 computers for the developers and a pay per use server to carry out the experiments, 7.000€, as the computers can be amortized in the future by other developers.

So the initial investment to do the proof of concept would be about 75.000€ (25.000€ if we don't count the labeling). If the results are good we could then request for extra budget to continue with the project and integrate it into a real app to make the investment worthwhile.

6. Conclusions and future development

The initial objective was to create a good teacher model and improve their results in order to use unlabeled data to keep improving the model. It is a hard topic to detect sentences of teenager blogs as it is a really concrete domain and jergon. Thanks to active learning and supervised learning we were able to surpass some of the F-score modified values that we initially had and create models that were performing better in some of the classes that we were trying to detect.

The initial results with the teacher model were promising, as we managed to surpass by more than 15% previous results with the same data. On the other hand I am certain there is a lot of room for improvement and either a deep theoretical analysis or many experiments should be done in order to improve the teacher model and take advantage of the data we have. Each hyperparameter or different techniques for the Bert fine tuning, from freezing Bert and training after that only the linear layers, to trying to improve our data quality with different preprocessing techniques.

There are a lot of different noises we can apply to sentences, and not only different types but the impact that you want to make in each of the sentences with each of the noises. For example, there exist more complex techniques that use synonyms or words of the same domain to apply word substitution. Each different way we apply them could lead to better results really fast, as we would be creating a greater distribution of our data and a more complex one.

I believe that we should invest time in creating complex noisy data that offer similar sentences without taking the complete context of them, and try all of this noise to see if it gives better results. After that I would suggest going for a more complex model and trying to apply stochastic depth and different ways of dropout to see if model noise infers positively in the results.

As the whole pipeline (from unlabeled data to student) is already done, trying new techniques and different noises is not that much time-consuming as the first time.We should be able to try a lot of experiments only by focusing in creating complex noise functions and follow the path that gives better results, so little by little we could improve the quality of our final student.

Once the student starts getting better results in all the different classes than the teacher, the future steps would be trying to do an iterative process using the former student as the teacher, and try to optimize the number of steps of this iteration we should do. It is possible that this technique fails, as explained by Xie et al, but it is worth the try if we have enough unlabeled data.

Another important thing to note, is that if the results are not improving enough with all mentioned previously, we might want to try a new database. It is important to say that the domain of the unlabeled dataset isn’t too close to the original dataset, so trying to learn from

two different domains (and more with this kind of training) is very complex and might lead to bad results. Even though both of the datasets focus on short sentences, the kind of jargon and the presence in some of the classes might not be adequate.

This is not too important in our case as it is only a proof of concept, but when implementing an application for business it is important than both the labeled data and the unlabeled data match the same domain that will be used for the application, so the finetuning of the Bert in the teacher and creating the student has to be trained and optimized for the results we want to obtain. For instance if we want to prevent cyberbullying in messages that usually arrive to the teenager phones, we could try to extract a lot more in that domain and only label a small portion of them, which would be a lot cheaper and could end up leading to the same or even better results than if we tried to label all of them.

7. Bibliography

[Alammar] Jay Alammar. "A Visual Guide to Using BERT for the First Time".

[Choi, 2021] Choi, Hyunjin, et al. "Evaluation of BERT and ALBERT Sentence Embedding Performance on Downstream NLP Tasks." 2020 25th International Conference on Pattern Recognition (ICPR). 2021.

[Cumalat 2020] Cumalat Puig, Eudald. "Sentiment analysis on short Spanish and Catalan texts using contextual word embeddings". MS thesis. Universitat Politècnica de Catalunya, 2020.

[Devansh 2020] Devansh. "Why Self-training with Noisy Students beats SOTA Image classification while using fewer resources. Pt 1: Setup and Noise". 2020.

[Devin 2018] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

[Donicke 2019] Damaschk, Matthias, Tillmann Dönicke, and Florian Lux. "Multiclass text classification on unbalanced, sparse and noisy data." Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing. 2019.

[Genç 2019] Özgür Genç. "The basics of NLP and real time sentiment analysis with open source tools". Towards data science blog. April 2019.

[Gildenblat 2020] Jacob Gildenblat. "Overview of Active Learning for Deep Learning". 2020.

[Joshi, 2020] Prateek Joshi. Transfer Learning for NLP: Fine-Tuning BERT for Text Classification. 2020.

[Lee 2017] Lee, Hye-Woo, Noo-ri Kim, and Jee-Hyong Lee. "Deep neural network self-training based on unsupervised learning and dropout." International Journal of Fuzzy Logic and Intelligent Systems 2017.

[Li 2018] Susan Li. "Multi-Class Text Classification with Scikit-Learn". February 2018.

[LIB 1] BERT multilingual base model (uncased). Hugging Face.

[LIB 2] BCEWithLogitsLoss, pytorch.

[Q&A 1] "Can support vector machine be used in large data?". Cross Validated, question and answer site for people interested in statistics, machine learning, data analysis, data mining, and data visualization.

[Rai 2017] Abhinav Rai. "Text Classification in NLP — Naive Bayes". 2017.

[Reddy 2018] Vasista Reddy. "Sentiment Analysis using SVM". 2018.

[Singh 2019] Singh P. Linear Regression. In: Machine Learning with PySpark. Apress, Berkeley, CA. 2019.

[URL 1] Sentiment Analysis: A Definitive Guide , Monkey Learn.

[URL 2] F-score, Wikipedia.

[URL 3] “Cómo combatir el ciberbullying a través de las tecnologías móviles”, Mobile World Capital Barcelona. 2017.

[Wei 2020] Colin Wei, Kendrick Shen, Yining Chen, Tengyu Ma. “Theoretical Analysis of Self-Training with Deep Networks on Unlabeled Data”. 2020.

[Xie 2020] Xie, Qizhe, et al. "Self-training with noisy student improves imagenet classification." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

[Meijomil 2019] Susana Meijomil. "Guía avanzada de Google BERT: qué es, cómo funciona y en qué te afecta".. 2019.