Using Annotation Error Detection to estimate the minimum number of annotators required

Mansi 200101064
Mahek Vora 200101062
Nandini Sharma 200101122
Siddharth Bansal 200101093

Abstract

In natural language processing tasks - getting correctly annotated noise free data is a difficult job. An individual annotator is very likely to make mistakes in labeling. The aim of this project is to find out the minimum number of annotators required so that their collective opinion on a dataset can be considered as correct. We also try to develop an AED(Annotation Error Detection) model for toxicity classification of social media comments and analyze the possible reasons for mismatch in the predicted and correct labels.

1 Introduction

Annotated corpora are essential in many scientific disciplines, including NLP(natural language processing). Corpora are used to train and evaluate machine learning models, deduce new knowledge, and reinforce/revise existing theories. Datasets are usually annotated by humans who can and do make mistakes. The usual method of resolving this kind of human error is to ask a group of humans to annotate. The size of this group cannot be too large because while it may give good results, the costs may become economically infeasible, but it also cannot be too small as it may give unsatisfactory results. Thus, finding the correct number of people to annotate a dataset is challenging.
In this article, we propose a method to approximately find the minimum number of people needed to get the correct annotations using an AED model. Our work mainly focuses on demonstrating our method by solving the problem of finding the minimum number of annotators needed to correctly label the toxicity of comments on a particular social networking site using our method.

2 Methodology

In this section, we describe the workflow where we discuss the reconstruction of the Master dataset and splitting it to get the **Analysis dataset** and the **Submaster dataset**. Further, we discuss the model building and training followed by our analysis on the Analysis dataset and experimenting with the hyperparameters to get the estimated minimum number of annotators required.

2.1 Flow

We have a master database that consists of comments and toxicity levels (values from 0 to 1, where 1 represents highly toxic and 0 represents that the message was not toxic). Here, from the dataset analysis, we know that for ambiguous comments, there are more annotators as compared to those which are non-ambiguous. For example, there are more than 1000 annotators for annotating ambiguous comments and around ten annotators for annotating non-ambiguous comments to increase accuracy.
Now, as we need to create a noisy dataset with fewer annotators, we get the frequency of annotations for the comments and plot this frequency distribution. From this distribution, we define the threshold value of the number of annotators **n_thresh**(the comments for which the number of annotators is greater than n_theresh are considered ambiguous, and those having lesser annotations are considered to be unambiguous).
Call the ratio of ambiguous to non-ambiguous comments as β. If β is very close to 0, we reconstruct the dataset by taking an appropriate number of comments from ambiguous and non-ambiguous comments so as to make β=0.1. In our case, 0.015% (considering the n_thresh as 75) of the comments were ambiguous and therefore, to make this ratio 0.1, we took half of the ambiguous comments(30,000) and clubbed them with 135,000 unambiguous comments to get the Submaster Dataset. We took the next half of the ambiguous comments(i.e. 15,000) to form the Analysis Dataset. We did so because, ambiguous comments would decide an upper limit on the minimum number of annotators required. Now, we build our model to predict the toxicity values for the comments considering the comments and other hyper-

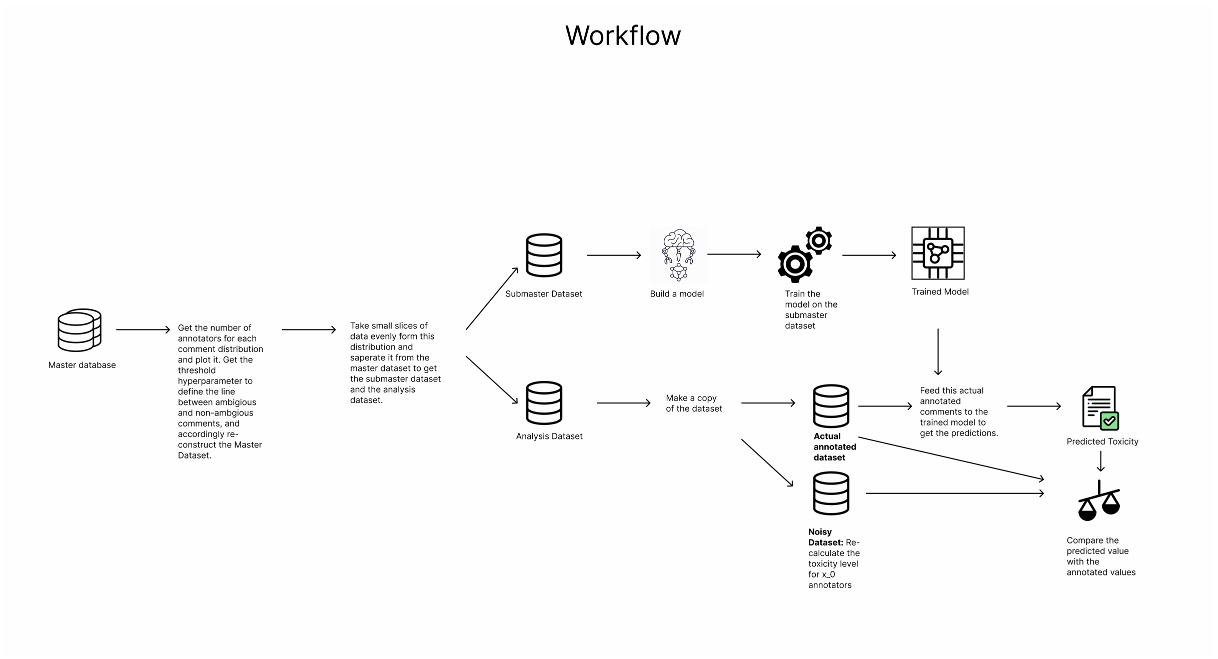

Workflow
Master database
Get the number of annotators for each comment distribution and plot it. Get the threshold hyperparameter to define the line between ambigious and non-ambgious comments, and accordingly re-construct the Master Dataset.
Take small slices of data evenly form this distribution and saperate it from the master dataset to get the submaster dataset and the analysis dataset.
Submaster Dataset
Build a model
Train the model on the submaster dataset
Trained Model
Analysis Dataset
Make a copy of the dataset
Actual annotated dataset
Feed this actual annotated comments to the trained model to get the predictions.
Predicted Toxicity
Noisy Dataset: Re-calculate the toxicity level for x_0 annotators
Compare the predicted value with the annotated values

Figure 1: This figure represents the workflow of the project.

parameters(will be decided based on the accuracy of results).
Next, we train our model on the Submaster Dataset. On the other side, we create a copy of our Analysis dataset and to make it noisy, we consider the annotations of only m (less than the minimum number of annotators of the set of comments in the analysis dataset) annotators and calculate the new toxicity values. Now, we feed the set of comments from the Analysis dataset to the trained model and get the predicted toxicity values.
We then compare the predicted toxicity values with the annotated toxicity values of the noisy dataset and those of the actual dataset. Then, we analyse the comments for which we observe a significant deviation from the predicted values.

2.2 Experimenting

To move the toxicity annotations close to the predicted values, we increase the value of **m** and observe the deviation.Thus reaching the value **m_0** for m for which we do not observe any great improvement in the annotated toxicity values. We call this value (m_0) as the sufficient number of annotators for annotating this dataset.

3 Dataset Description

3.1 Overall dataset explanation

The dataset is borrowed from Kaggle - Jigsaw Unintended Bias in Toxicity Classification. It has 1999516 rows all of which are different comments on the platform civil comments and contains several attributes of the comments including toxicity. Here toxicity in online conversations is defined as anything rude, disrespectful or otherwise likely to make someone leave a discussion.We have 46 columns for each sample in the data. The attributes include toxicity sub attributes (severe_toxicity, obscene, threat, insult, identity_attack, sexual_explicit), 25 identity based attributes (includes gender, religions), annotator counts for toxicity, identity and metadata from civil comments. Each comment was shown to up to 10 annotators, some comments were seen by many more than 10 annotators (up to thousands), due to sampling and strategies used to enforce rater accuracy. We have decided a threshold based on the number of annotators on comments to decide the ambiguity of a comment.Keeping the value of the threshold 70, roughly 0.015% of the dataset(about 30,000 comments) is ambiguous. So to avoid working with overly skewed data we are taking only a portion of the unambiguous data. This reduced the size of our dataset to about 300,000 entries.

3.2 Insights From the graph

From the graph (fig.2) we can infer that majority of the comments are not toxic (having toxicity between 0 to 0.3). Therefore, in order to get meaningful conclusions from our analysis dataset, we

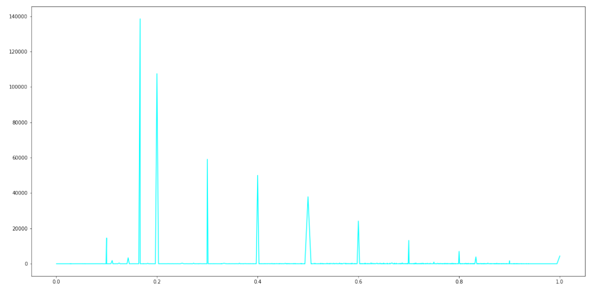


Figure 2: This graph represents the number of comments(*y-axis*) v/s the toxicity value(*x-axis*)

need to slightly normalize this distribution. This is explained in the methodology.

3.3 Prepossessing

As there are 46 columns in the Master dataset, we have to drop a few in order to get only the relevant ones. For a baseline model, we are only considering the toxicity and comments column.

4 Model

4.1 Model description

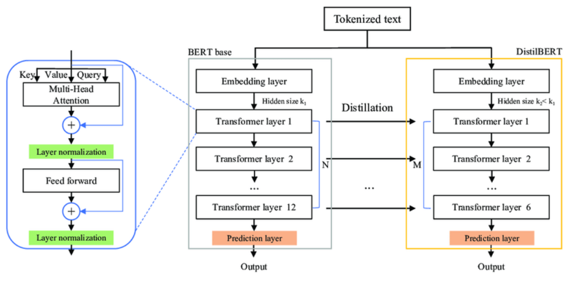


Figure 3: the DistilBERT model architecture and components (Dahou et al.)

We have fine-tuned an instance of a pre-trained DistilBERT model (from hugging face). This model is a distilled version of the BERT base model DistilBERT is a transformers model that uses the BERT base model as a teacher, which was pre-trained in a self-supervised fashion on the raw texts, with no humans labelling them in any way with an automatic process to generate inputs and labels from those texts using the BERT base model. The 3 main objectives for pertaining are as follows:

- **Distillation loss:** The model is trained to return the same probabilities as the BERT base model.

- **Masked language modelling (MLM):** this is part of the original training loss of the BERT base model. When taking a sentence, the model randomly masks 15% of the words in the input then run the entire masked sentence through the model and has to predict the masked words. It allows the model to learn a bidirectional representation of the sentence which makes it different from traditional recurrent neural networks (RNNs) that usually see the words one after the other, or from autoregressive models like GPT which internally mask the future tokens.

- **Cosine embedding loss:** the model was also trained to generate hidden states as close as possible to the BERT base model.

So, the model learns the same inner representation of the English language as its teacher model, while being faster for inference or downstream tasks.

4.2 Hyperparameters

One of the hyperparameters is the threshold that determines if a comment is ambiguous. If the number of annotators for a comment exceeds 70, the comment will be considered ambiguous. 29427 comments from the entire dataset(1999516 entries) are thus considered ambiguous. Now, we will construct our dataset using a fraction of ambiguous and unambiguous comments, and this fraction will be considered as another hyperparameter (roughly 10% of the Submaster Dataset is the ambiguous comment and the rest are unambiguous in our case). For training the distilBERT model, we also need multiple hyperparameters such as number of epochs, batch size and number of batches, etc.

4.3 Metrics

For evaluation, we used **RMS** value of the differences in the values predicted by our model and the values obtained by using a limited number of annotators. Our goal is to minimise the gain obtained by increasing the number of annotators. This is the point of saturation for the annotator count and the optimum number of annotators required, which is determined by the value at which the slope tends to zero in fig.6a.

5 Results

5.1 Finding optimum number of annotators

The toxicity count observed for the 100 different comments (from the analysis dataset) for different number of annotators and the actual values are

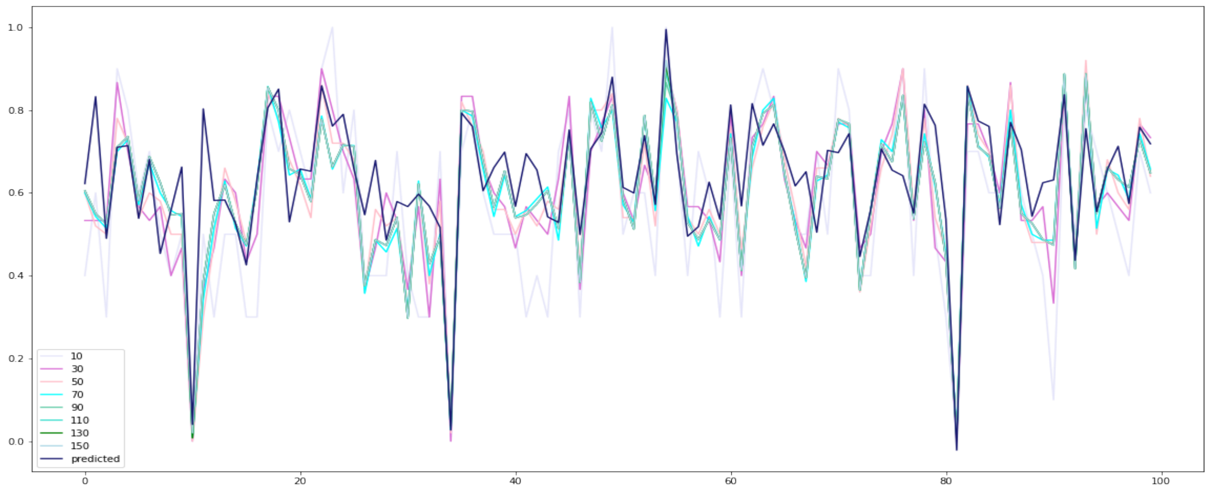


Figure 4: Toxicity count for various number of annotators and the predicted values.

shown in fig. 4. We can see that on increasing the number of annotators, the graph for toxicity count gets closer to the actual values.(the lines become a darker shade for higher annotator count to better show the convergence of toxicity values) As explained before, The optimal number of annotators is the point where on increasing the number of annotators, the RMS value doesn't change much or the slope of the RMS vs annotator count becomes very small. From fig.6a, we can see that the optimal point, or the optimum annotator count came out to be around 90 annotators.

5.2 Prediction vs actual values

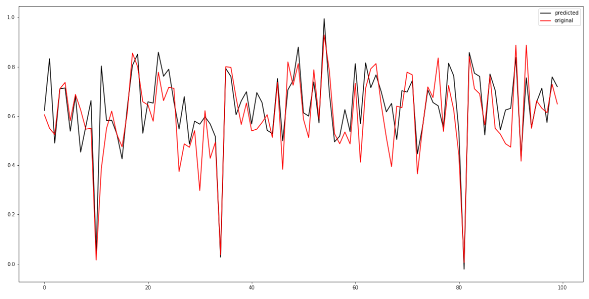
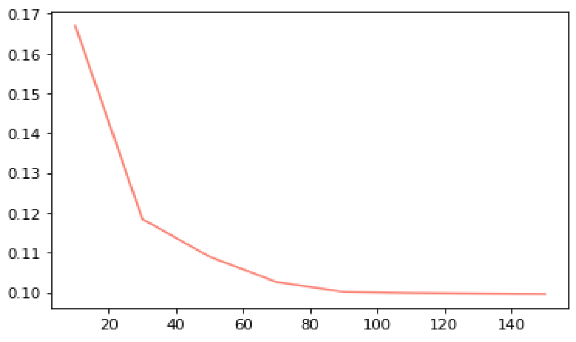
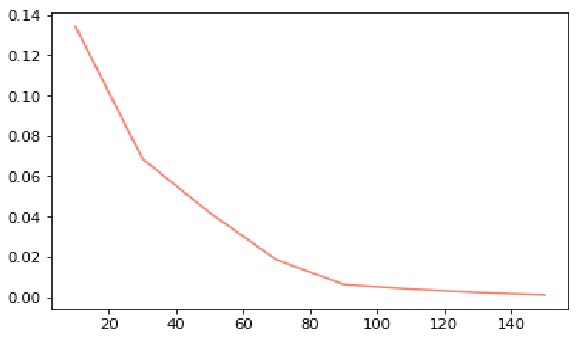


Figure 5: actual values vs predicted values

From the figure fig.5 we can see that the values predicted by our model roughly match the actual values of toxicity. Looking at the figure fig.6b, which is the RMS graph when using the actual values of the toxicity, we see that the slope of the graph becomes small near 90 annotators(where the x-axis represents the number of annotators) in the original dataset as well showing that our model can find the saturation point accurately.



(a) RMS with predicted values



(b) RMS with Original Values

Figure 6: RMS values vs Annotator count

6 Real life Application

In real life scenario, consider a problem where we need to find the expected number of annotators for a given set of comments. The approach with our pre-trained model would go as follows:

1. Let the size of the dataset be x comments. Shuffle the dataset and take a small portion(1%) and feed it to the model to get the predicted values.

2. Now, assign a few annotators say 10 and make them annotate the 1% of the dataset. And keep on increasing the number till saturation is reached. use the slope of the RMS vs annotator count curve for finding out saturation.

3. This should be the optimal number of annotators required to be hired for the job.

7 Conclusion and Future Work

We have applied our algorithm on the given problem and arrived at results which were very similar to what we obtained by using the actual anotated data. For further improving the model, some hyperparameters could be tuned. We could also analyse the nature of comments for which we observe a significant deviation from the annotated values and classify them to figure out the issues in our model ad improve them. **Human In Loop** principle could also be used to improve our model by feeding in the corrected annotations for the Analysis dataset.

8 Links for reference:

1. **Data Analysis:** Notebook Link
2. **Model Training:** Notebook Link
3. **Result Analysis:** Notebook Link
4. **Model:** Drive Link
5. **Dataset:** Drive Link

References

Abdelghani Dahou et al. 2022. Improving Crisis Events Detection Using DistilBERT with Hunger Games Search Algorithm.
the diagram for the DistilBERT model architecture and components.