


End-to-End Modeling and Characterization of Crowdsourcing Annotators

Saumya Goyal  , Soner Durmaz  , and Alex Marin Felices  

Department of Informatics, Technical University of Munich

 saumya.goyal@tum.de

 soner.durmaz@tum.de

 alex.marin@tum.de

July 30, 2021

Abstract — Crowdsourcing has experienced a big boom with the increasing interest in obtaining labelled data. It is a powerful way of obtaining data in a cheaper and faster way. However, annotator biases and spammers can affect the final quality of the models created with this data. In our paper, we use different end-to-end methods such as Latent Truth Network and Fast Dawid-Skene to try and model annotator-specific biases as bias matrices. These models will help us obtain the ground truth estimation with the help of the singly-labelled Organic dataset. Apart from these, we also propose a new method where we combine both models by creating predictions to convert our dataset into a multi-labelled one. We modelled the biases of each annotator to see if their annotations are reliable or not, and to detect possible spammers. Apart from these, we also clustered the bias matrices to discover groups of annotators that approach the labelling task in the same way. We were able to find these different groups of annotators and by adding noise to one of the annotators, we also were able to cluster it as a spammer. Even though our dataset was quite limited in length, which made the training of bias matrices hard, we were able to show our approach can indeed model biases for annotator clustering and spammer detection.

1 Introduction

As per the [22], the oxford dictionary that we know today, was made with the help of extremely smart people who were called for a re-examination of English language. Because the task of such high magnitude was and is not possible to be carried out by an individual, it is deployed to the crowd. This method of using power of crowd for carrying out a task is called crowdsourcing. The Oxford English Dictionary is therefore the first project in which a task was outsourced to a crowd – defining crowdsourcing in the truest sense of the word that we use today.

There are many applications in crowdsourcing that exist such as fundraising, asking people to review movies and books, or dividing up and parallelizing complex tasks to be completed. The concept deals majorly with breaking up very large problem that may or may not be solved by one individual and too trivial for computers. Amazon Mechanical Turk [23] and Crowdfunder are examples of this global micro-work platforms where the actual job of people is to submit these small tasks.

As per [24], in the recent years, crowdsourcing has gained popularity in the sentiment analysis, where the data produced by humans in the form of tweets, comments, telephonic talks, reviews, status updates, etc., are classified as positive, negative or neutral. The application of this kind of segregation on the human opinionated data has become highly beneficial for the companies to know how their customers think and feel about their products or services. Companies such as Gengo.ai, crowdsource, etc., provide this kind of sentiment analysis to their customers like Facebook, Amazon, Expedia and more.

There can not be just one sentiment analysing algorithm which can produce the accurate results in all applications. Because sentiment analysis depends highly on the context [25], domain and also on the language in which opinion was formulated. For this reason, cognitive abilities of humans are required and hence human annotators are normally used to label this kind of human opinions data. And this task of labelling the sentiments is crowdsourced.

In these days, with the labelled sentiments and deep learning methods, it is very easy to find the bias of a particular human being towards a product or company. But what if the annotators that labelled the data themselves are biased? In that case, our deep learning algorithm would produce an entirely biased results. Also, what if, the annotator spams the labels by marking all the received samples as "negative" sentiment? To tackle this kind of annotator bias we tried com-

paring already existent methods such as David-Skene (DS) [1] and Latent Truth Network (LTNet) [2] and worked on Organic dataset [6] to conclude which annotator bias finding technique works best. We also clustered these annotators based on the bias matrices that we trained for both these models.

1.1 Research Questions

Our entire project focused mainly on finding the techniques suitable for finding bias in the singly-labelled crowdsourced annotated dataset, for this reason we formulated the following research questions:

- 1] How the two approaches - Latent Truth Network and David Skene are different in carrying out the task of finding annotator's biases?
- 2] What is the end-to-end approach in finding the bias?
- 3] Can the model detect the spammers from the Bias matrix?
- 4] Can multi-labelled data be generated by using singly-labelled data through the end-to-end approach?
- 5] Which approach is better in finding the biases in annotators, by finding the ground truth or using end-to-end method?

1.2 Outline

In the further sections, we will discuss about the related work which will include architectures of Latent Truth Network and David-Skene. We will also include details on the methods used and experiments that were carried out. Finally, we will conclude with our results and further projected developments on this topic. We also faced limitations during this project work and have also included that in this report.

2 Related Work

2.1 Crowdsourcing Algorithms

2.1.1 Problem definition

Nowadays, with the increasing need for data because of the continuously growing machine learning research, it is key to find good data collection methods. They should be fast and cheap and maintain the maximum quality possible. This is what crowdsourcing tries to do. It is a way to get lots of data from internet non-expert participants, and in our case, we are interested in the annotations they do to our text-based dataset. One of the main problems with this method of getting

annotations is the unreliability of these workers, which can affect the quality of the data [4]. Generally, there are 2 reasons that may affect its quality: (1) Spammers, which deeply affects the quality of data because they label our provided data in a random way or directly in a bad manner. (2) Annotation biases, which are caused by the different personal backgrounds of the annotators and affect the overall accuracy of the model.

2.1.2 Ground truth estimation

Dawid-Skene [1] is an algorithm to infer the ground truth with the use of a variation of the Expectation-Maximization (EM) algorithm by estimating annotator biases and latent labels in turns. We will use its newer and faster extension named Fast Dawid-Skene to obtain the ground truth estimation from the crowdsourced dataset. This algorithm is used to estimate the true labels by aggregating the crowdsourced labels. It outputs for each data point the most likely correct class, given its labels from many annotators out of a set of classes. The main advantage of this 'hard' version of DS, is that it allows much faster convergence while also maintaining similar accuracy in aggregation.

2.1.3 End-to-end-approaches

Similarly as Major Adverse Cardiac Events (MACE) [5] infers true labels, DS models each annotator with its respective bias matrices. However, we are looking for an end-to-end approach that learns the latent truth while the model is being trained, as well as extracting annotator information and feature distribution [2, 8, 9]. So, we want a model that directly from training is able to obtain all the relevant information, and this is not done by DS or MACE which infer the ground truth only from the labels without considering the input features. Apart from this, there are some disadvantages of the EM algorithm, for example that it can be unstable and more expensive to train [10]. LTNet (see Figure 1) is slightly modified to create a similar approach for crowdsourcing. In this case, it represents the annotator bias by means of confusion matrix estimates, while in the original image-based model, it tries to model imperfect annotations extracted from different image datasets using only one latent truth neural network and the bias matrices from each specific dataset. So, in order to model crowd annotators on our singly-labelled sentiment analysis from the Organic dataset we will use the LTNet.

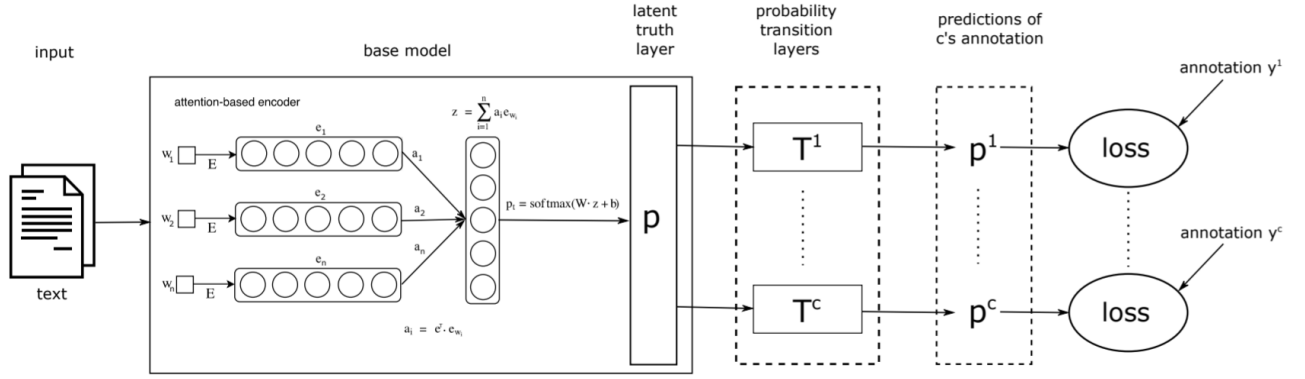


Figure 1 End-to-end trainable LTNet architecture [2]. The base model [7] and the LTNet architecture are inspired by [2] and [19].

2.2 Labelled Crowdsourced Datasets

2.2.1 Multi-Labelled Crowdsourced Datasets

This type of datasets contains for each sample multiple labels and they are a better option than singly-labelled ones when there is a limited amount of data. Although this requires reducing the noise by aggregating the labels using ground truth estimation. It has been proven that with many non-expert annotators it is possible to have better outcomes while also spending less resources than with a few expert annotators [4]. Some examples of this type of datasets are Google GoEmotion [14] and the SEWA database [15]

2.2.2 Singly-Labelled Crowdsourced Datasets

When given unlimited data and given limited resources, singly-labelled datasets are a great option. Even if the annotations are made by non-experts it is possible to model worker quality with single labels [9]. This means that these datasets can be cheaper, while still obtaining similar results as to the ones with multiple annotations. This is the case of the organic dataset we use in this work. Other examples of this type are TripAdvisor Dataset [16], the Amazon Review Dataset [17] and the Large Movie Review Dataset [18].

3 Methodology

3.1 Data Preprocessing

Mistakes, redundancies, missing values, and inconsistencies are the things that compromise the integrity of the model and reduce the efficiency as well. Thus, before using the data for actually getting the insights, it needs to be as organized and clean as possible. For this purpose we tried cleaning the data with few of

the relevant preprocessing techniques listed below (we made use of NLTK Library [12] for doing this):

3.1.1 Stop Word Removal

Stop words are available in big amount in any human language. By removing these words, we remove the low-level information from our text so that we can give more focus to the important information of our dataset. Removal of such words does not show any negative consequences on the model that we train for our task. This task does reduce the dataset size and thus reduces the training time due to the fewer number of tokens involved in the training.

For example, in English there are words like 'a', 'an', 'the', 'and', etc., which exist in nearly every sentence and when these words are removed it is called as the stop words removal preprocessing.

3.1.2 Lower Case Conversion

There is a common approach used during the preprocessing which involves lowercasing everything for the sake of simplicity. It helps to maintain the consistency flow during the Natural Language Processing (NLP) tasks. The lower() function makes the whole process quite straightforward. Also, the model cannot understand the same words if all of the letters are not in the same case, for example the model cannot see the 'paper' and 'Paper' as same. So, lowering all the cases helps the model to understand the context of a sentence.

3.1.3 Lemmatisation

Lemmatisation performs normalization using vocabulary and morphological analysis of words from the

dictionary. It aims to remove inflectional endings and returns the base or dictionary form of a word, which is known as the lemma. Lemmatisation is built on WordNet's [27] built-in morphy function, making it an intelligent operation for text analysis. A WordNet module is a large and public lexical database for the English language. Its aim is to maintain the structured relationship between the words. In English languages, prefixes and suffixes are added all the time which take some extra letters or changes the form of the original word for grammatical reasons such as 'swim' and 'swam' have same meanings but in a different forms. The aim of lemmatisation is to reduce such words by reducing them to the word in their original form. For example, 'cared', 'caring' will be reduced to the word 'care'.

3.1.4 Word to Vector Conversion

As per [26], GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. In order to capture this quantitatively we need some method that can distinguish words 'man' from 'woman', it is necessary for a model to associate more than a single number to the word pair. A natural and simple way to do this for an enlarged set of discriminative numbers is the vector difference between the two word vectors. GloVe was designed in a way such that the vector differences capture as much as possible the meaning specified by the juxtaposition of two words. Hence we made use of GloVe50 for our task separating each word into 50 different contexts to give meaning that the computer understands.

3.2 Modelling Architecture

Our architecture has three different components. They are; a base model in which we take sample text and get latent truths, a bias model which takes latent truth and learns the bias matrix to give model predictions for each annotators, and the last step contains the FDS algorithm which finds the ground truth of each sentence based on annotator's prediction. In our experiments, we used these three components with different settings. Further we explain these three components.

3.2.1 Base Model

This model takes preprocessed sentences which are converted to GloVe50 vectors. Next the attentions are trained as a parameter. The output of the attention vector is then forwarded to the linear layer which generates the logits. Next, these logits are sent to the softmax function the output of which is called as the latent truth. Until this step everything is common for all the annotators, so the latent truth layer is the common base which gives probabilities of each class in which the sentences can be classified as for all the annotations.

Attention Layer Attention is a vector, that outputs of dense layer using softmax function. Before attention mechanism, translation relied on reading a complete sentence and compressing all the information into a fixed-length vector, for example, a sentence with hundreds of words represented by several words will surely lead to information loss, inadequate translation, etc.. Whereas the attention mechanism fixes this problem, it allows machine translator to look over all the information the original sentence has and then generates the correct word according to the current word it works on and relying on the context as well. It can even allow translator to zoom in or out (focusing on local or global features).

Aim of the use of attention layer is the trying to find important words which can be beneficial in finding the context of the sentence. This layer takes word vectors and multiplies them with the attention vector, and takes the weighted averages of the word attentions, passing the results to the further layer.

Softmax Layer As per [20], the softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1. The input values can be positive, negative, zero, or greater than one, but the softmax transforms them into values between 0 and 1, so that they can be interpreted as probabilities. If one of the inputs is small or negative, the softmax turns it into a small probability, and if an input is large, then it turns it into a large probability, but it will always remain between 0 and 1. The softmax function is sometimes called the softargmax function, or multi-class logistic regression. This is because the softmax is a generalization of logistic regression that can be used for multi-class classification, and its formula is very similar to the sigmoid function which is used for logistic regression. Softmax is very useful because it

converts the scores to a normalized probability distribution, which can be displayed to a user or used as input to other systems. For this reason it is usual to append a softmax function as the final layer of the neural network.

The aim of use of the softmax function is classifying the vectors into classes. It gives us the output which are the probabilities telling a specific class the sentence belongs to.

3.2.2 Bias Model

It starts with a layer that contains bias matrices (they are only matrices in the dimensions of (number of classes)*(number of classes) which are used as training parameters. All the annotator specific bias matrices are multiplied with latent truth and the result is used as the annotator specific prediction of that particular sentence. With this Bias model we are able to get 10 different annotations for every sentence. This particularly becomes useful when we pass the output of this model as the input of FDS.

Bias Matrices In our case, trained bias matrices can be considered as matrices that tell us about the personality of the annotators. For example, if a person is optimistic then we expect that the model classifies the samples for that person mostly positive so for example the column of the bias matrix that corresponds to positive sentiment will contains higher probability than other cells of the matrix. Formally, every cell of the bias matrix tells, the probability of a sentence being annotated as positive when we know originally that the sentence was annotated as positive. The deviation in this captures the annotator specific bias.

Similarly if the person is pessimistic, the bias matrix will have more values in the column and row corresponding to the negative sentiment label. Finding an accurate bias matrix is the ultimate aim of the project, as it helps in determining what a particular annotator will annotate a newly found sentence as.

Loss Function As per Jason Brownlee [21], Deep learning neural networks are trained using the stochastic gradient descent optimization algorithm. As part of the optimization algorithm, the error for the current state of the model must be estimated repeatedly. This requires the choice of an error function, conventionally called a loss function, that can be used to estimate the loss of the model so that the weights can be updated to reduce the loss on the next evaluation. Neural network

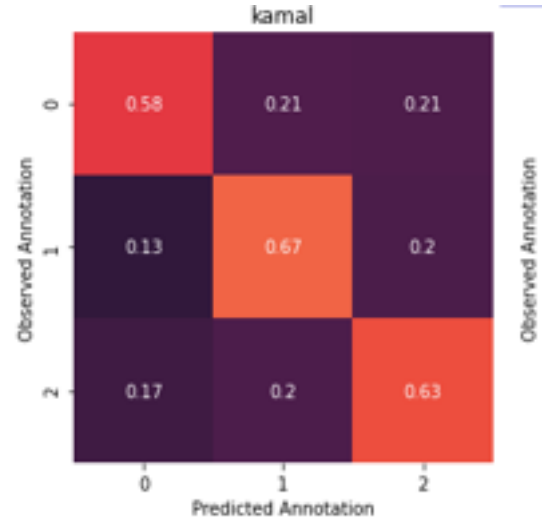


Figure 2 Bias Matrix - Example of the bias matrix for annotator Kamal. (Legend: 0 = Negative, 1 = Neutral, 2 = Positive sentiment)

models learn a mapping from inputs to outputs from examples and the choice of loss function must match the framing of the specific predictive modeling problem, such as classification or regression. Further, the configuration of the output layer must also be appropriate for the chosen loss function.

In our case, the aim of the loss function is measuring the error between prediction and actual value, for that reason we made use of the Negative Log Likelihood function with the Adam optimizer.

3.2.3 Ground Truth Estimation Model

We have created a ground truth estimation model with our singly-labelled dataset. This is optional ground truth estimation model during the bias modeling. In this stage, we used FDS algorithm to get ground truths based on annotator specific predictions. We used the FDS algorithm for this purpose. Internally the EM algorithm is used for the ground truth estimation purpose. As the input to this algorithm we passed on the predictions from the LTNet model which gave us the multi-labelled dataset. As an output we received ground truth estimation for every sentence, which helped in creating the confusion matrix.

3.3 Combining the Models

To get a result from annotator opinions, we combined the predictions from the Bias model and Ground Truth Estimation model. We did this because our dataset is singly-labelled and we cannot benefit from different

opinions in our opinion mining task. So, to convert our singly-labelled data to a multi-labelled data we used predictions. Then, we fed the Ground Truth Estimation model with that multi-labelled dataset.

3.4 Annotator Clustering

To cluster the annotators, we used the K-Means and Hierarchical clustering algorithms. It is an algorithm that classifies based on the list of numerical values. So, we decided to create such list via flattening the bias and confusion matrices. We also used Principal component analysis (PCA) for reducing the dimension of our model as a preprocessing for the clustering task.

K-Means Algorithm Basically, the aim of the K-Means algorithm is clustering the samples into K classes. K-Means tries to optimize the clusters such that the samples in a class are maximally similar and the samples in different classes are maximally different.

4 Experiments

4.1 Our Dataset

We performed all our experiments on the Organic Dataset which contains social media texts discussing organic food related topics [7].

Source: The dataset was crawled in 2017 from Quora, a social question-and-answer website. Search terms used were - "organic", "organic food", "organic agriculture", and "organic farming". The comment was marked relevant by a domain expert if they had relevance with organic food or agriculture and discussed characteristics, advantages, and disadvantages of organic food production and consumption. Among all the crawled data, around 4,161 comments were chosen from the 10,439 comments that were crawled.

Annotation Scheme: Each sentence was labelled with sentiment (positive, negative, neutral) and entity, the sentiment target, annotated. For our sentiment analysis task we only considered sentiments as label for each sentence.

Annotation Procedure: The data was annotated by 10 annotators and it was divided into 10 batches of 1000 sentences for each annotator and none of these batches shared any sentences between each other.

Data Split : 4616 sentences were retrieved which contained organic entities with 39 percent neutral, 32 percent positive, and 29 percent negative sentiments. For the experiment we had split the annotated data

as 80 percent training, 10 percent validation, and 10 percent test set.

We Experimented with four different methods to find Bias among the annotators and used different architecture in each of these methods. Next section discusses about them in detail.

4.2 Experiment 1: End-to-End approach

We used the complete training dataset containing 4,161 sentences and trained the entire model in end-to-end fashion. Unlike LTNet and DS, in this approach, we trained the attention vectors, Latent-Truth layer and bias matrix simultaneously. This made sure that the prediction of sentiment from the model was dependent on the input sentences and not just on the sentiment labels received from the annotators.

We had to use masking technique to mask annotators which are not being considered in a one loop. This was particularly important because we trained the entire training model in one go and not 10 different times for 10 annotators.

We trained our model by dividing it into x batches of y size. The loss function that we used was negative log-likelihood and we used optimizer as Adam. The learning rate was found using the random search where we found 1e-5 and finally we fine-tuned it with grid search for values [1e-3:1e-7] and concluded that learning rate of 1e-5 gave us the best results in terms of accuracy and F1-score during the training.

Below are few of the mathematical equations that we were used to find the latent truth and bias matrix, as per [2]

$$a_j = e \cdot e_{w_j} \quad (1)$$

$$z_n = \frac{1}{L} \sum_{i=1}^L a_i \cdot e_{w_i} \quad (2)$$

$$p_n = \text{softmax}(W \cdot z_n + b) \quad (3)$$

$$P(y_n^i | x_n) \neq P(y_n^j | x_n), \forall x_n \in \mathcal{X}, i \neq j \quad (4)$$

$$\tau_{ij}^c = P(y_n^c = j | y_n = i) \quad (5)$$

$$P(y_n^c = j | x_n; \theta) = \sum_{i=1}^L P(y_n^c = j | y_n = i) P(y_n = i | x_n; \theta) \quad (6)$$

We initially trained our model for 100 epochs but as the training data was not big enough, the accuracy of

our model in 100 epochs was not good and hence we increased the number of epochs to 500. This increased our confusion matrix diagonal values. For fine tuning our predictions we also initialized our weight vectors of Linear model with Xavier Initialization.

We stopped our training when the confusion matrix had high values in the diagonal and very low values in non-diagonal positions. This made sure the learnable parameters had learned about each annotator.

During the training our model learned attention vectors, linear layer parameters and 10 different bias matrices for every annotator.

4.2.1 Results

As a result, we found a model which has overall accuracy of 46%, and overall F-1 score of 38%. Furthermore, we also have bias matrices (see appendix 1) which are the annotator specific network parameters and also annotator specific confusion matrices (see appendix 2) based on the output of model and observed labels. Then we saw that the confusion matrices gave us better results on neutral sentiment, and worst results on negative sentiment. And the two matrices, which are bias and confusion, were not similar especially for negative sentiments. In our perspective the reason behind this is our low overall accuracy and F1 score.

4.3 Experiment 2: Ground Truth finding

While finding the bias the most intuitive way that comes to our minds is finding the ground truth, for example, whether the sentiment of the sentence was actually positive or not and did the annotator labelled it as positive or incorrectly as negative or neutral?

This variation from the ground-truth in the annotators labelling is the bias in the annotators thinking and this is what we need to find. As per David Skene paper they used the non-supervised EM algorithm to find the ground truth estimates for multi-labelled dataset by using only the annotators review and not on actual text sentence [3].

However, in our case, every sentence was labelled only once and only by one annotator. In that case we needed 10 different annotator reviews for each sentence to generate the ground truth estimates. Hence, in this experiment we first tried getting the ground truth estimates by generating 10 predictions for every sentence.

Each annotators prediction for each sample (multi-label annotation) was generated using the experiment

1 where we simply took each sentence and passed through our architecture of experiment 1, just as you would do in the testing phase. This step was essential to get the ground truth of sentiment for every sentence.

Next we passed our newly generated dataset to the FDS algorithm which gave the ground truth estimation. Following this, we created the confusion matrix considering the original dataset labels of each annotator and comparing them with the ground truth that we generated.

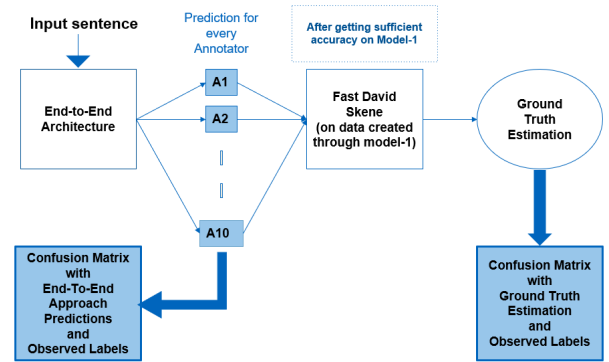


Figure 3 Hybrid Model - Architecture of the end-to-end trainable LTNet model chained to the FDS[3]. The base model is a simple attention model with a single trainable attention vector and linear layer with parameters W and b to get the latent truth. Each row of the transition matrix T is constrained to be summed to 1.

4.3.1 Results

As a result, we found a model which has overall accuracy of 36%. This is lower than the result of experiment 1. But these numbers should not be compared because their method is very different. In the end to end approach we are looking for annotator specific prediction, whereas here we use ground truth which considered the predictions collectively. Furthermore, we also have annotator specific confusion matrices (see appendix 3) based on the output of model and observed labels. Then we saw that, the confusion matrices gave us better results on neutral sentiment, and worst results on negative sentiment as like previous experiment. And the two matrices, which are bias and confusion, were not similar especially for negative sentiments. In our perspective the reason behind this is our low overall accuracy.

4.4 Experiment 3: Clustering the annotators

Our main aim of the task was to find the biases in the annotator for the sentiment analysis and also to cluster them.

For clustering the annotators we used the bias matrix that was a learned parameter from our experiment 1. As the bias matrix gave us probability of annotator giving label as predicted label (pl) when the actual value is actual label (al), where al and pl belong to the set positive, negative, neutral, this was the best parameter to cluster the annotator.

We first used the K-Means algorithm for the clustering task. We used the elbow method to reach to the conclusion of using the $K = 3$ for the K-means algorithm. But for this task we required a list of numbers. As we had bias matrix, we had to flatten the matrices into lists. This gave us a list of 9 values for each annotator. Using these list of 9 values we clustered the annotators, for this we made use of K-Means function of the Sklearn library [28].

We did not wanted to rely on just one approach of clustering, hence we used the Hierarchical clustering as well. We used the same 9 values (3x3) from the model 1 bias matrix for each annotator. This time we got the dendrogram diagrams and it helped us cluster the annotators.

For confirming our clustering approach, we tried reducing the dimension of our bias matrices, using PCA, and we chose the first two principal components. Then again using the same approach of hierarchical clustering we clustered the annotators.

4.4.1 Results

As a result, we found three clusters. The first cluster only contains the annotator 'Fahad', the second cluster contains the annotators 'Hannah', 'Omar' and 'Felix', and the third cluster contains rest. It can be seen in our dendrogram from Hierarchical Clustering in appendix 4, which is also the same for PCA analysis.

4.5 Experiment 4: Spammer Detection

We wanted our model to detect the spammers in the crowdsourced dataset and hence in this experiment we tried finding the spammer keeping our architecture the same but by adding a spammer in our dataset.

As we had a very limited number of annotator specific annotations, we added noise to one of our annotators, 'Sumit', by converting his neutral annotations into the positive annotations. This made 'Sumit' a

spammer who in any case marks the sentiment of the sentence as positive.

Next we tried training our model for up to 500 epochs again but this time with our self created spammer's annotation included. One we had bias matrices and the ground truth estimations we again clustered our annotators. Hoping that this time our clustering algorithm will spot our spam annotator and cluster it separately. Our goal in this experiment was only to detect the 'Spammer' and 'Non-spammer' groups and not to do more clustering within these sub clusters. So we stopped our clustering when we got two different clusters in the hierarchical clustering.

We also reduced the dimension of our bias matrices using PCA for this clustering as well. We used the first two principal components and passed on to the function for clustering.

4.5.1 Results

We manipulated the data for annotator 'Sumit'. And, our clustering model detect the 'Sumit' as a different cluster from others, we had two clusters, one is for spammers, and the other one is for non-spammers. You can see the resulting bias matrices based on the trained model which includes manipulated data of 'Sumit' in appendix 5.

5 Limitations

With the singly-labelled Organic dataset, we had 4,161 sentences that were relevant and amongst that they had to be distributed to 10 different annotators. So every annotator got around 400 (approx) dataset for annotation. And because the data was evenly distributed for the positive, negative and neutral sentiment - for each class for each annotator there were around 130 annotations. This made the training of bias matrices very hard.

Because the dataset was small we had to run many epochs during the training process. This consumed lots of computational resources and time. Especially for finding the hyper-parameters we had to run many epochs too.

6 Conclusion

We showed with the help of Latent Truth Network (LTNet)[2], architecture and bias modeling on the singly-labelled crowdsourced data, how we can cre-

ate an end-to-end model for finding the bias in the annotators.

We figured out how the two approaches of LTNet and Fast David-Skene (FDS) [3] are different from each other for the bias modelling. The LTNet trained in the end-to-end fashion considering the actual text for finding the latent truth and learning the attention vectors during the training and this latent truth then becomes the common ground for all the annotators bias matrices. Whereas in the latter approach of FDS, the ground truth sentiment for each sentence is found using the multi-labelled dataset.

We found that the multi-labelled predictions that are produced from our experiment 1 could be chained to the FDS input. This particularly works because the output of our experiment 1 and input of FDS is in line. This approach can thus help in finding the ground truth in the singly-labelled dataset, which is considered as a very difficult task.

The bias and the confusion matrices produced by our improved architecture was able to precisely detect the spammer amongst the annotators. Moreover, the clustering of the annotators based on the bias matrix also seemed to work with our architecture. Also, the produced bias shows high robustness under very noisy conditions making the approach potentially usable outside of lab conditions.

As we worked on just singly-labelled dataset, we tell that if finding ground truth is not absolutely necessary we can ignore the chaining part and find the bias in the annotators with the end-to-end latent truth model approach, whereas if there is the necessity to find the ground truth labels chaining of the two architecture could result in potential outcome.

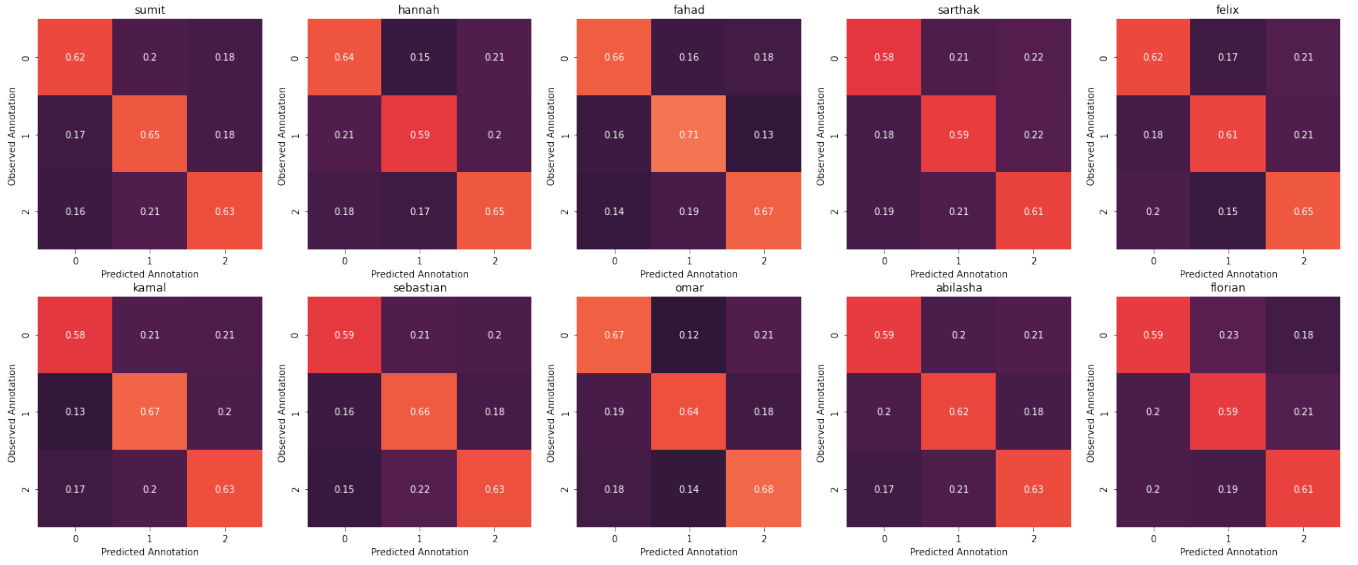
We believe it is necessary to conduct more experiments on more datasets from different sources to solidify our conclusions regarding our hybrid approach of finding ground truths, as the singly-labelled crowdsourcing use case was performed on a very small dataset. Furthermore, we believe that there might be many different use cases other than the sentiment analysis which can be explored.

References

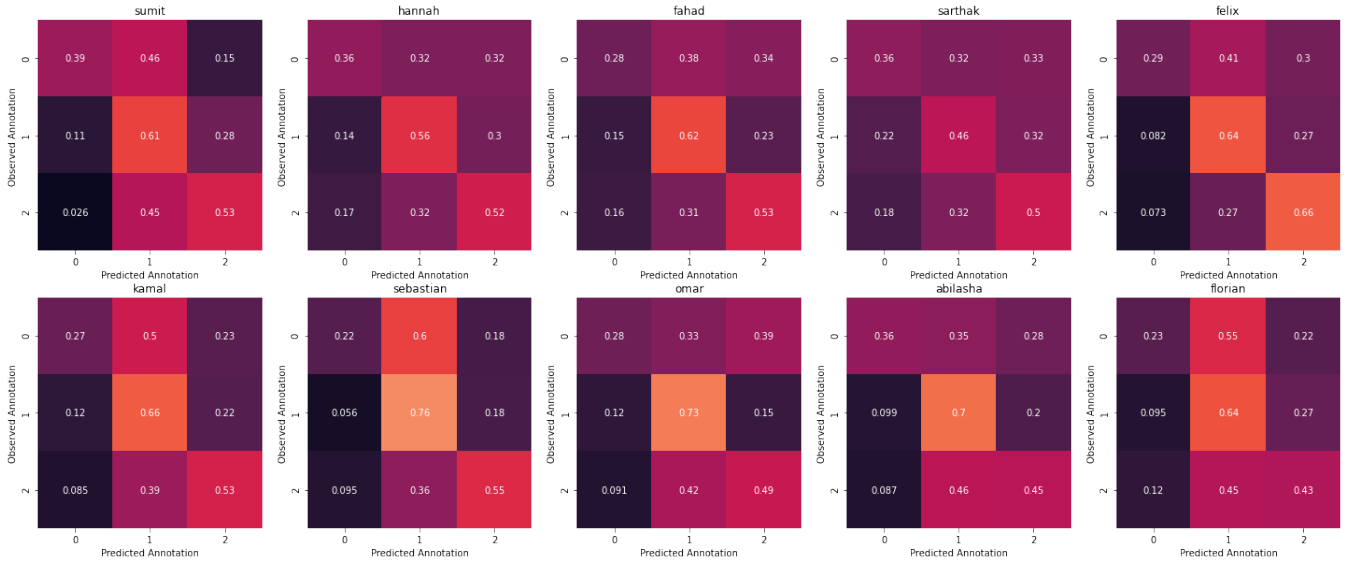
- [1] Dawid, A. P., & Skene, A. M. (1979). Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1), 20-28.
- [2] Zeng, J., Shan, S., & Chen, X. (2018). Facial expression recognition with inconsistently annotated datasets. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 222-237).
- [3] Sinha, V. B., Rao, S., & Balasubramanian, V. N. (2018). Fast dawid-skene: A fast vote aggregation scheme for sentiment classification. *arXiv preprint arXiv:1803.02781*.
- [4] Snow, R., O'connor, B., Jurafsky, D., & Ng, A. Y. (2008, October). Cheap and fast—but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 conference on empirical methods in natural language processing* (pp. 254-263).
- [5] Hovy, D., Berg-Kirkpatrick, T., Vaswani, A., & Hovy, E. (2013, June). Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 1120-1130).
- [6] Organic dataset. (2021). [Dataset]. <https://github.com/ghagerer/organic-dataset>
- [7] Hagerer, G., Szabo, D., Koch, A., Dominguez, M. L. R., Widmer, C., Wich, M., Danner, H., & Groh, G. (2021). End-to-End Annotator Bias Approximation on Crowdsourced Single-Label Sentiment Analysis.
- [8] Rodrigues, F., & Pereira, F. (2018, April). Deep learning from crowds. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 32, No. 1).
- [9] Khetan, A., Lipton, Z. C., & Anandkumar, A. (2017). Learning from noisy singly-labeled data. *arXiv preprint arXiv:1712.04577*.
- [10] Chu, Z., Ma, J., & Wang, H. (2020). Learning from crowds by modeling common confusions. *arXiv preprint arXiv:2012.13052*.
- [11] Pennington, J., Socher, R., & Manning, C. (2014, October). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL <https://www.aclweb.org/anthology/D14-1162>.

- [12] Bird, S., Loper, E. & Klein, E. (2009), *Natural Language Processing with Python*. O'Reilly Media Inc.
- [13] <https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/>
- [14] Demszky, D., Movshovitz-Attias, D., Ko, J., Cowen, A., Nemade, G., Ravi, S.: *Goemotions: A dataset of fine-grained emotions* (2020)
- [15] Kossai, J., Walecki, R., Panagakis, Y., Shen, J., Schmitt, M., Ringeval, F., Han, J., Pandit, V., Toisoul, A., Schuller, B., Star, K., Hajiye, E., Pantic, M.: *Sewa db: A rich database for audio-visual emotion and sentiment research in the wild*. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43(3), 1022–1040 (2021). <https://doi.org/10.1109/TPAMI.2019.2944808>
- [16] Thelwall, M.: *Gender bias in sentiment analysis*. *Online Information Review* 42(3), 343–354 (2018)
- [17] Ni, J., Li, J., McAuley, J.: *Justifying recommendations using distantly-labeled reviews and fine-grained aspects*. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. pp. 188–197. Association for Computational Linguistics, Hong Kong, China (Nov 2019). <https://doi.org/10.18653/v1/D19-1018>
- [18] Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: *Learning word vectors for sentiment analysis*. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA (2011)
- [19] Dhall, A., Goecke, R., Lucey, S., Gedeon, T.: *Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark*. In: *ICCV Workshops*. pp. 2106–2112 (2011)
- [20] Wood, T.: <https://deeptai.org/machine-learning-glossary-and-terms/softmax-layer>
- [21] Brownlee, J.: <https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/>
- [22] *Evolution of crowdsourcing - from its beginnings to the present*. (2018, April). Clickworker. <https://www.clickworker.com/2018/04/04/evolution-of-crowdsourcing/>
- [23] Sorokin, A., & Forsyth, D (2008). *Utility data annotation with amazon mechanical turk*. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, Anchorage, AK, USA, 2008. IEEE Computer Society. doi: 10.1109/CVPRW.2008.4562953.
- [24] Faggella, D. (2018, December 13). *Crowdsourced Sentiment Analysis – Applications in Social Media and Customer Service*. Emerj. <https://emerj.com/partner-content/crowdsourced-sentiment-analysis-applications-social-media-customer-service/>
- [25] Kara, Y. E., Genc, G., Aran, O., & Akarun, L. (2015). *Modeling annotator behaviors for crowd labeling*. *Neurocomputing*, 160, 141-156.
- [26] Pennington, J. (2015, August). *GloVe: Global Vectors for Word Representation*. Nlp.Stanford.Edu. <https://nlp.stanford.edu/projects/glove/>
- [27] Miller, G. A. (1995). *WordNet: a lexical database for English*. *Communications of the ACM*, 38(11), 39-41.
- [28] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). *Scikit-learn: Machine learning in Python*. *the Journal of machine Learning research*, 12, 2825-2830.

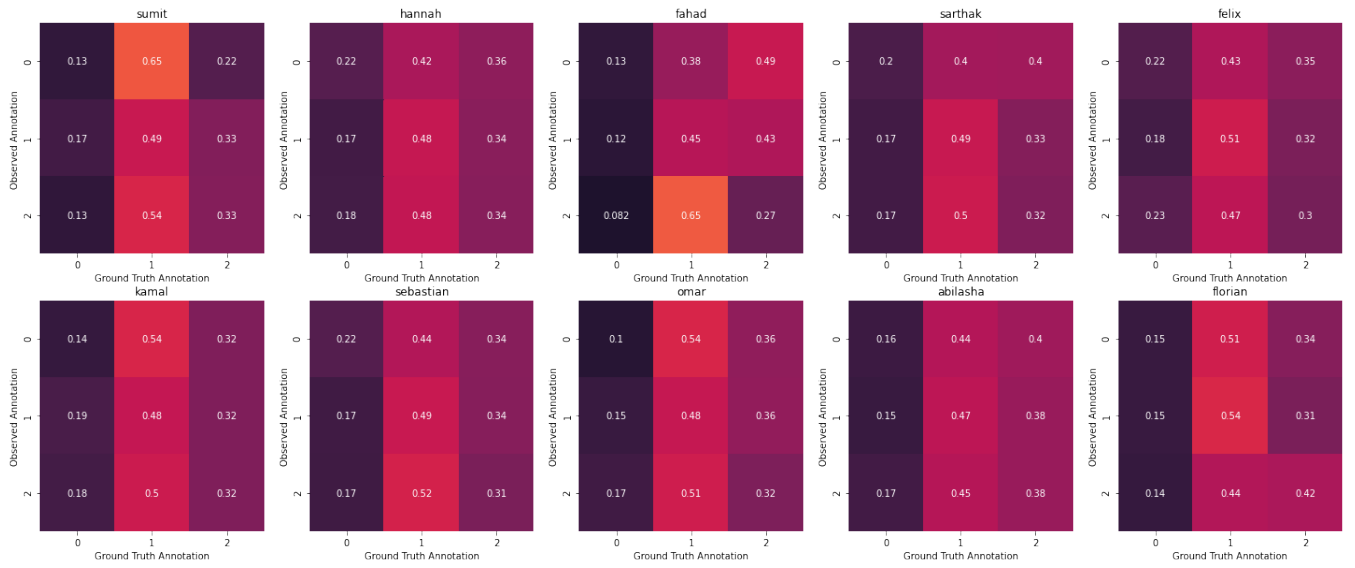
Appendices



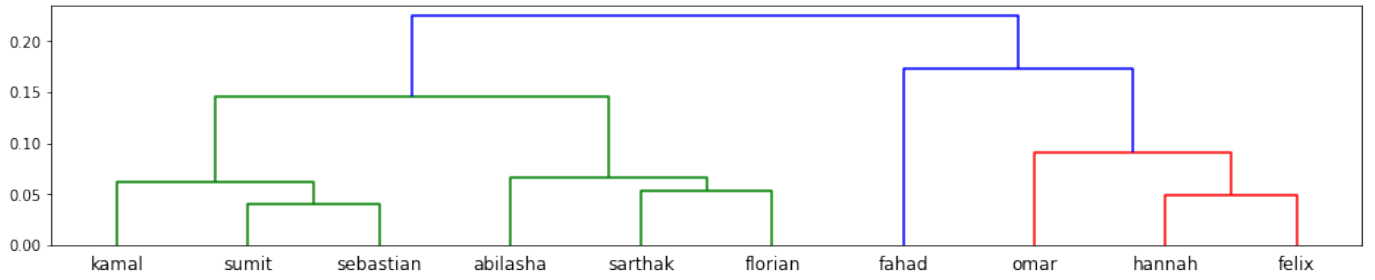
Appendix 1: Bias Matrices for Experiment 1 - Without Ground Truth estimation



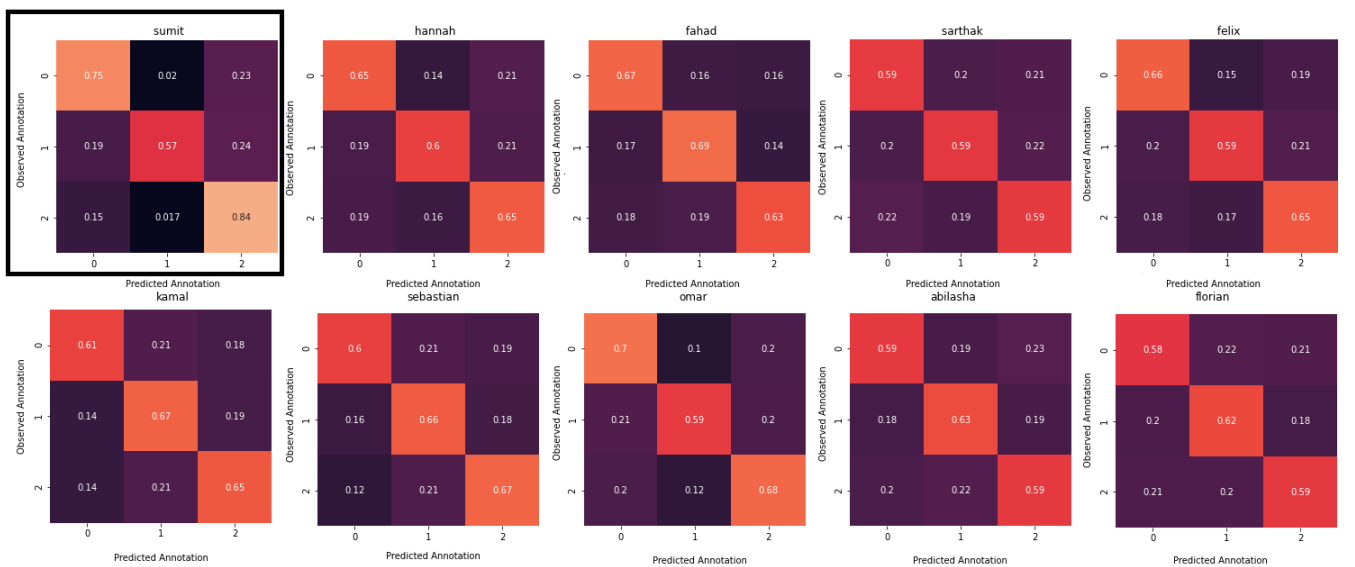
Appendix 2: Confusion Matrices for Experiment 1 - Without Ground Truth estimation



Appendix 3: Confusion Matrices for Experiment 2 - Using Ground Truth estimation



Appendix 4: Dendrogram for Experiment 3 - Annotator Clustering



Appendix 5: Bias Matrices for Experiment 4 - Spammer detection