A Large-Scale Semi-Supervised Dataset for Offensive Language Identification

A Large-Scale Semi-Supervised Dataset for Offensive Language Identification

Sara Rosenthal[1], Pepa Atanasova[2], Georgi Karadzhov[3], Marcos Zampieri[4], Preslav Nakov[5]
IBM Research[1], University of Copenhagen[2], University of Cambridge[3]
Rochester Institute of Technology[4], Qatar Computing Research Institute, HBKU[5]
sjrosenthal@us.ibm.com, pepa@di.ku.dk
georgi.karadzhov@cl.cam.ac.uk, marcos.zampieri@rit.edu
pnakov@hbku.edu.qa

Abstract

The use of offensive language is a major problem in social media which has led to an abundance of research in detecting content such as hate speech, cyberbulling, and cyber-aggression. There have been several attempts to consolidate and categorize these efforts. Recently, the *OLID* dataset used at SemEval-2019 proposed a hierarchical three-level annotation taxonomy which addresses different types of offensive language as well as important information such as the target of such content. The categorization provides meaningful and important information for understanding offensive language. However, the *OLID* dataset is limited in size, especially for some of the low-level categories, which included only a few hundred instances, thus making it challenging to train robust deep learning models. Here, we address this limitation by creating the largest available dataset for this task, *SOLID* . *SOLID* contains over nine million English tweets labeled in a semi-supervised manner. We further demonstrate experimentally that using *SOLID* along with *OLID* yields improved performance on the *OLID* test set for two different models, especially for the lower levels of the taxonomy. Finally, we perform analysis of the models' performance on easy and hard examples of offensive language using data annotated in a semi-supervised way.

1 Introduction

Offensive language in social media has become a concern for government entities, online communities, and social media platforms. One of the most common strategies to tackle the problem is to train systems capable of recognizing messages containing offensive language, which can then either be deleted or set aside for human moderation.

WARNING: This paper contains tweet examples and words that are offensive in nature.

There have been a number of studies on the application of computational methods to dealing with offensive language, particularly for English (Davidson et al., 2017; Basile et al., 2019). As pointed in a recent survey (Fortuna and Nunes, 2018), various terms have been used in the literature to describe phenomena with overlapping characteristics such as toxicity, hate speech, cyberbullying, and cyber-aggression, to name a few.

Recent studies have investigated the overlap between these abusive language detection tasks (Waseem et al., 2017). The Offensive Language Identification Dataset, or *OLID* ,[1] (Zampieri et al., 2019a), used in SemEval-2019 Task 6 (OffensEval) (Zampieri et al., 2019b), is one such example. *OLID* contains 14,100 English tweets, which were manually annotated using the following three-level taxonomy:

A: Offensive Language Detection;

B: Categorization of Offensive Language;

C: Offensive Language Target Identification.

The taxonomy proposed in *OLID* makes it possible to represent different kinds of offensive content such as hate speech and cyberbulling as a function of the *type* and of the *target* of a post. For example, many offensive messages targeting a group are likely to be hate speech whereas many offensive messages targeting an individual are likely to be cyberbulling. *OLID* 's taxonomy became popular due to its simplicity and flexibility. It has been used to annotate datasets in other languages such as Arabic (Mubarak et al., 2020) and Greek (Pitenis et al., 2020), thus allowing for multilingual learning and analysis.

[1] https://sites.google.com/site/offensevalsharedtask/olid

An inherent feature of the hierarchical annotation approach is that the lower levels of the annotation scheme contain a subset of the instances in the previous level of annotation. This represents an important shortcoming as more levels of annotation are included in the taxonomy, the less instances there are be in each lower category, thus making it very difficult to train robust deep learning models on such datasets. In addition, due to the natural infrequency of offensive language (e.g., less than 3% of the tweets are offensive when selected at random), obtaining offensive content is a costly and time-consuming effort. In this paper, we address these limitation by proposing a new dataset: **S**emi-Supervised **O**ffensive **L**anguage **I**dentification **D**atatset (*SOLID*).[2]

Our contributions can be summarized as follows:

1. We are the first to apply a semi-supervised method for collecting new offensive data using *OLID* as a seed dataset, thus avoiding the need for time-consuming annotation.

2. We create and publicly release *SOLID* , a dataset containing 9 million English tweets for offensive language identification, the largest dataset for this task.

3. We demonstrate sizeable improvements compared to prior work on the mid and lower levels of the taxonomy where gold training data is scarce when training on *SOLID* and testing on *OLID* .

4. We provide a comprehensive analysis of *EASY* (i.e., simple explicit Tweets such as using curse words) and *HARD* (i.e., more implicit tweets such as the use of underhanded comments or racial slurs) examples of offensive tweets.

The remainder of this paper is organized as follows: Section 2 presents related studies in aggression identification, bullying detection, and other related tasks. Section 3 describes the *OLID* dataset and annotation taxonomy. Section 4 introduces the computational models used in this study. Section 5 presents the *SOLID* dataset. Section 6 discusses the experimental results and section 7 offers additional discussion and analysis.

[2] Available at: https://sites.google.com/site/offensevalsharedtask/solid

2 Related Work

In this section, we discuss offensive language detection and related tasks.

Aggression Identification: The TRAC shared task on Aggression Identification (Kumar et al., 2018) provided participants with a dataset containing 15,000 annotated Facebook posts and comments in English and Hindi for training and validation. A Facebook and Twitter dataset were used for testing. The goal was to discriminate between three classes: non-aggressive, covertly aggressive, and overtly aggressive.

Bullying detection: There have been several studies on cyber-bullying detection. For example, Xu et al. (2012) used sentiment analysis and topic models to identify relevant topics, and Dadvar et al. (2013) used user-related features such as the frequency of profanity in previous messages.

Hate speech identification: This is by far the most studied abusive language detection task (Kwok and Wang, 2013; Burnap and Williams, 2015; Djuric et al., 2015; Ousidhoum et al., 2019; Chung et al., 2019). One of the most widely used datasets is the one by Davidson et al. (2017), which contains over 24,000 English tweets labeled as non-offensive, hate speech, and profanity. A recent shared task on the topic is HatEval (Basile et al., 2019).

Toxicity detection: The Toxic Comment Classification Challenge[3] was an open competition at Kaggle, which provided participants with comments from Wikipedia organized in six classes: toxic, severe toxic, obscene, threat, insult, and identity hate. The dataset was also used outside of the competition (Georgakopoulos et al., 2018), including as additional training material for the aforementioned TRAC shared task (Fortuna et al., 2018).

Offensive language detection: This is the most relevant line of research, with several shared tasks on this topic. For example, the GermEval 2018[4] (Wiegand et al., 2018) shared task focused on offensive language identification in German tweets. A dataset of over 8,500 annotated tweets was provided for a course-grained binary classification task in which systems were trained to discriminate between offensive and non-offensive tweets. Another

[3] http://kaggle.com/c/jigsaw-toxic-comment-classification-challenge

[4] http://projects.fzai.h-da.de/iggsa/

Tweet	Level A	Level B	Level C
@USER Does anyone care what that dirtbag says???	OFF	TIN	IND
Poor sad liberals. No hope for them.	OFF	TIN	GRP
LMAO....YOU SUCK NFL	OFF	TIN	OTH
@USER What insanely ridiculous bullshit.	OFF	UNT	NULL
@USER you are also the king of taste	NOT	NULL	NULL

Table 1: Examples from the *OLID* dataset

relevant shared task is the one at HASOC 2019[5] (Mandl et al., 2019). In this paper, we build on the work of Zampieri et al. (2019a) and the aforementioned *OLID* dataset. *OLID* is annotated following a hierarchical three-level annotation schema. It differs from other datasets that have used hierarchical annotation schemes (Basile et al., 2019; Mandl et al., 2019) because it takes both the target and the type of offensive content into account. This allows multiple types of offensive content (e.g., hate speech and cyberbulling) to be represented in *OLID* 's taxonomy. Here, we extend this work by creating a large-scale semi-supervised dataset, following the same annotation taxonomy as in *OLID* .

3 The OLID Dataset

The *OLID* (Zampieri et al., 2019a) dataset tackles the challenge of detecting offensive language using a hierarchical labeling schema.

3.1 Labeling Schema

The schema for the *OLID* proposes a hierarchical modeling of offensive language, which classifies each example using the following three-level hierarchy:

3.1.1 Level A: *Offensive Language Detection*

Level A asks whether the text is offensive (OFF) or not (NOT):

NOT: content that is neither offensive, nor profane;

OFF: content containing inappropriate language, insults, or threats.

3.1.2 Level B: *Categorization of Offensive Language*

Level B asks whether the offensive text is targeted (TIN) or not (UNT):

TIN: targeted insult or threat towards a group or an individual;

UNT: text containing untargeted profanity or swearing.

3.1.3 Level C: *Offensive Language Target Identification*

Level C categorizes the target of the offensive content:

IND: the target is an individual explicitly or implicitly mentioned in the conversation;

GRP: hate speech, targeting group of people based on ethnicity, gender, sexual orientation, religious belief, or other common characteristic;

OTH: targets that do not fall into any of the previous categories, e.g., organizations, events, and issues.

3.2 Annotated Data

The training part of the *OLID* dataset contains 13,241 examples, while the testing part contains 860 tweets. Detailed statistics about the distribution of the labels in all 3 levels are shown in Table 3. We can see that there is a substantial class imbalance on each level of annotation, most significant at Level B. Furthermore, there is a sizable difference in the total number of annotations between different levels: level A contains 13,241 tweets, but level C only has 4,089 tweets. While this is expected, based on the nature of the annotation schema itself, it indicates a need for creating a large dataset. Table 1 shows examples from the *OLID* dataset, demonstrating all possible label combinations.

4 Models

Below, we describe the models we use in the semi-supervised annotation framework and subsequently

[5] https://hasocfire.github.io/hasoc/2019/index.html

for evaluating the contribution of *SOLID* for offensive language identification. In order to maximize the diversity of the models' rationales, we construct a suite of heterogeneous machine learning models consisting of PMI (Turney and Littman, 2003), FastText (Joulin et al., 2016), LSTM (Hochreiter and Schmidhuber, 1997), and BERT (Devlin et al., 2019). These models have diverse inductive biases, which is an essential prerequisite for our semi-supervised setup (see Section 4.5).

4.1 PMI

We use a PMI-based model that computes the n-gram-based similarity of a tweet with respect to the tweets of a particular class c in the training dataset. In its nature, the model is naïve as it accounts only for the n-gram frequencies in the discrete token space and only for the context of the n neighboring tokens. Following (Turney and Littman, 2003), we compute the PMI score for each n-gram in the training set with respect to each class as follows:

$$PMI(w_i, c_j) = log_2 \left(\frac{p(w_i, c_j)}{p(w_i) * p(c_j)} \right) \quad (1)$$

where $p(w_i, c_j)$ is the frequency of n-gram w_i in instances of class c_j, $p(w_i)$ is the frequency of n-gram w_i in instances from the entire training dataset, and $p(c_j)$ is the class frequency.
Additionally, we find that semantically oriented PMI scores contribute to an improved performance of this naïve method, and we compute them as follows:

$$\begin{aligned} PMI - SO(w_i, c_j) = \\ = log_2 (\frac{p(w_i, c_j) * p(C \setminus \{c_j\})}{p(w_i, C \setminus \{c_j\}) * p(c_j)}) \end{aligned} \quad (2)$$

where $C \setminus \{c_j\}$ is the set of all classes except c_j.

At test time, for each instance, we add the PMI and the PMI-SO scores for each unigram and bigram with respect to each class. Then, we select the class with the highest score. We additionally remove words that appear less than five times in the training set and we add a smoothing constant of 0.01 for all frequencies. If the instance contains no words with associated scores, we choose NOT for Level A and UNT for level B, which are the most likely classes to contain a word with a neutral class orientation. For Level C, we choose class IND, which is the majority class.

4.2 FastText

Considering the naturally noisy structure of tweets, a reasonable extension over the word-based model is to use subword representations. A strong subword model is FastText (FT) (Joulin et al., 2016), which has shown strong performance on various tasks without the need for extensive hyper-parameter tuning. The FastText model for text classification uses a shallow neural model, similar to the continuous bag-of-words model (Mikolov et al., 2013), but instead of predicting the word, based on its neighbors, it predicts the target label based on the words in the sample. The input of this neural model is a one-hot representation of the word and also of its n-grams (Joulin et al., 2016).

We use FastText for two main reasons. First, FastText is inherently different from both the simple PMI model and from the heavy-lifting LSTM and Transformer models, thus giving a valuable modeling representation for the overall ensemble. Second, while deep neural models have greater representational and modeling power, their performance may vary greatly depending on the task, the number of training examples, and the amount of hyper-parameter tuning performed. Thus, we argue that the robustness and the simplicity of FastText is a good choice for semi-supervised training.
We train the FastText model with bigrams and a learning rate of 0.01 for Levels A and B, and with trigrams and a learning rate of 0.09 for Level C. For all tasks, we use a window of size 5 and a hierarchical softmax loss.

4.3 LSTM

The LSTM (Hochreiter and Schmidhuber, 1997) model builds a continuous representation of the input tweet in a sequential manner, where at each step, it decides which information to update, reset, and output to the next step. The LSTM model can account for long-distance relations between words, but its loss of information along the steps can become severe. One partial solution of the information bottleneck problem, which we also use, is to use an attention mechanism (Vaswani et al., 2017), thus allowing for lookups of previous step outputs.
The first layer of the LSTM model is an embedding layer, which we initialize with a concatenation of the GloVe 300-dimensional (Pennington et al., 2014) and FastText's Common Crawl 300-dimensional embeddings (Grave et al., 2018). The

embedding layer is then followed by a dropout and a bi-directional LSTM layer with an attention mechanism on top of it. We concatenate the results of the attention mechanism with both averaged and maximum global poolings over the outputs of the LSTM model. The final prediction is produced by a sigmoid layer for Levels A and B, where we have binary classification, and softmax for Level C, where we have three classes.

We train the LSTM model using early stopping with a patience of five epochs over the validation loss. For Level A, we use an LSTM model with a hidden size of 128, a dropout rate of 0.3, a batch size of 256, and a learning rate of 0.0002. For Level B, the LSTM model has a hidden size of 50, a dropout rate of 0.1, batch size of 32, and a learning rate of 0.0001. Finally, the Level C LSTM model has hidden size of 50, a dropout rate of 0.1, batch size of 32, and a learning rate 0.0001. We use the Adam optimizer for training.

4.4 BERT

We use BERT as an instance of the Transformer models, where the representation of each token is constructed by attending to all tokens in the input using multiple attention heads. While Transformers do not encode positional information in the same way as recurrent models do, sentence order is still modeled, but using positional embeddings.

In our experiments, we use the base uncased model, which has 12 layers, a hidden size of 768, and 12 attention heads, amounting to 110 million parameters. We use the classifier wrapper of the BERT model from HuggingFace (Wolf et al., 2019). Finally, we fine-tune the BERT model for each task, starting from the same pre-trained base model.

BERT has achieved (nearly) state-of-the-art performance for a number of NLP tasks, showing both high representational power and nice robustness across tasks. By using a pre-trained model and fine-tuning it for our tasks, we enjoy the benefits of transfer learning in a low-resource setup.

We fine-tune BERT for 2, 3, and 3 epochs for Level A, B, and C, respectively, and we use learning rates of 0.00002 for Levels A and C, and 0.00004 for Level C. In order to cope with the data imbalance in Level C, we apply per-class weights on the loss as follows: IND=1, GRP=2, OTH=10. We use the Adam optimizer and a linear warm-up schedule with a 0.05 warm-up ratio.

4.5 Democratic Co-training

Democratic co-training is a semi-supervised technique that can be effectively used to create large datasets with noisy labels when provided with a set of diverse models trained on a supervised dataset. This approach has been successfully applied in tasks like time series prediction with missing data (Mohamed et al., 2007), the early prognosis of academic performance (Kostopoulos et al., 2019), or in the health domain (Longstaff et al., 2010).

In our work, we employ democratic co-training (Zhou and Goldman, 2004) to create semi-supervised labels for all three levels of *SOLID* using *OLID* as our seed set. In democratic co-training, distant supervision is conducted by an ensemble of models with different inductive biases:

1. Train N *diverse* supervised models $\{M_j(X)\}$, where $j \in [0, N]$ on a labeled dataset $X = \{(x_i, y_i)\}$, where $i \in [1, |X|]$

2. For each example x'_i in the unannotated dataset $X' = \{(x'_i)\}, |i \in [1, |X'|])$ and each model M_j, predict the confidence p'^j_i for the positive class.

We train all models on the *OLID* dataset. We use a 10% split of the training dataset for fine-tuning. The performance of the individual models on the *OLID* dataset is shown in Table 2.

BERT is the best model for Level A. The PMI model performs almost on par with the LSTM models, which we attribute to the size of the dataset and to the fact that a simple lexicon of curse words would be highly predictive of the offensive content present in a tweet. The FastText model has the worst performance, which is still relatively high compared to the rest of the models.

BERT is best for Level B as well, followed by LSTM. PMI and FastText perform even worse as the task becomes harder for frequency and n-gram based approaches.

At Level C, the overall performance of the models gets even worse as both the size of the dataset becomes smaller, and the task gets harder. The BERT and the LSTM models outperform FastText and PMI, with BERT being the best model.

5 The *SOLID* Dataset

In this section, we describe the process of collecting a large dataset of over 12 million tweets used

Model	**Level A**	**Level B**	**Level C**
BERT	0.816	0.705	0.568
PMI	0.684	0.498	0.461
LSTM	0.681	0.657	0.585
FastText	0.662	0.470	0.590

Table 2: Macro-F1 score of the models in the democratic co-training ensemble on the *OLID* test set.

Level	**Label**	***OLID***		***SOLID***	
		Train	**Test**	**Train**	**Analysis**
A	OFF	4,640	240	1,448,861	1,090
	NOT	9,460	620	7,640,279	2,807
B	TIN	4,089	213	149,550	850
	UNT	551	27	39,424	1,072
C	IND	2,507	100	120,330	580
	GRP	1,152	78	22,176	190
	OTH	430	35	7,043	80

Table 3: Train and Test/Analysis data distribution for the *OLID* and the *SOLID* datasets.

for training and analysis. All of the data was labeled using the democratic co-training approach described in the previous section.

5.1 Large-Scale Dataset of Tweets

We collected our data from Twitter using the Twitter streaming API[6] via Twython[7]. In order to download truly random tweets, we searched using the 20 most common English stopwords such as *the*, *of*, *and*, *to*, etc. Using such stopwords ensured that we were more likely to obtain English tweets as well as a nice variety of random tweets while staying within the rate limiting constraints imposed by Twitter. We kept the stream running the entire time and we continuously chose a stopword at random but based on its frequency in a large monolingual corpus[8]. We then collected 1,000 tweets for that stopword. Thus, tweets including more frequent stopwords were collected more frequently. A full list of the stopwords and their frequency is shown in Appendix A.1.

We used *langdetect*[9] to select English tweets and we discarded tweets that were less than 18 characters or less than two words long. We substituted all user mentions with `@USER` for anonymization purposes, and we further ignored tweets with URLs as those tend to be less interesting and not offensive. This procedure allowed us to collect over 6,000 quality tweets in 40 minutes. In total, we collected over 12 million tweets. We kept 9 million as training data, which we release publicly, and we created an analysis dataset from the remaining 3 million tweets with more remaining for future testing and analysis. All of our data was labeled in a semi-supervised manner using the democratic co-training approach described in Section 4.5.

5.2 Semi-Supervised Training Dataset

To create our semi-supervised training set, we computed the average and the standard deviation of the confidences predicted by each of the models in the democratic co-training setup, as described in Subsection 4.5. Therefore, the semi-supervised dataset *SOLID* $= \{(x'_i, p'_i) | i \in [1, |\textit{SOLID}|]\}$ where p'_i = avg$(\{p'^j_i | j \in [1, N]\})$, std$(\{p'^j_i | j \in [1, N]\})$ is the aggregated single prediction. The distribution of the scores across the dataset is shown in Appendix A.2.

In particular, we computed the scores based on the confidences for the positive class at Levels A and B and the confidences for the IND, GRP, and OTH classes at Level C. We performed the above aggregation step instead of just using the scores of each model in order to avoid overfitting to any particular model in the ensemble. Our intention for having both the average and the standard deviation was to stimulate different aggregation schemes of the semi-supervised data and, at the same time, to disallow model approximations or any assumptions based on the models.

First, we annotated all the tweets in a semi-supervised manner with the labels of level A. Then, we selected the subset of tweets that all models were confident were likely offensive (BERT and GRU $\geq$ .5, PMI and FT=OFF) to annotate for Level B. Finally, we chose the tweets likely to be TIN at Level B with a standard deviation lower than 0.25 for Level C. The size and the label distribution across the datasets can be found in Table 3 and examples of tweets and their confidence for each model can be found in Table 4.

[6] https://developer.twitter.com/en/docs
[7] https://twython.readthedocs.io/en/latest/
[8] Project Gutenberg: https://en.wiktionary.org/wiki/Wiktionary:Frequency_lists#Project_Gutenberg
[9] https://pypi.org/project/langdetect/

Level	Text	BERT	LSTM	FT	PMI	AVG	STD	Label	E/H
A	@USER he fucking kills me. he knew it was coming	0.919	0.958	0.852	0.509	0.809	0.177	OFF	E
	His kissing days are over, he's a pelican now!	0.659	0.304	0.568	0.523	0.514	0.131	NOT	H
	i think we're all in love with winona ryder	0.060	0.038	0.017	0.480	0.102	0.155	NOT	E
B	Guess I'll just never understand the fucking dynamics	0.901	0.569	0.001	0.617	0.522	0.327	UNT	H
	@USER Government is a bunch of bitches.	0.013	0.221	0.000	0.397	0.158	0.164	TIN	E
	@USER Give me the date. Fuck them other niggas Bro I'm irritated as fuck	0.882	0.666	0.983	0.701	0.808	0.131	TIN	E
C	@USER He was useless stupid guy	0.807	0.915	1.000	0.480	0.801	0.197	IND	E
	It's like mass shootings is the reg in this shit hole country!	0.826	0.479	0.693	0.570	0.642	0.131	OTH	H
	Getting these niggas tatted is a overstatement are ya dead serious	0.700	0.691	0.770	0.491	0.663	0.104	GRP	H

Table 4: Training data aggregation examples. The first four columns show the confidence of each of the models with respect to the positive class in Level A and B (OFF, UNT) and only for the corresponding class in C (one example for each of the classes: TIN, GRP, OTH). The label column shows manual annotations, and the last column shows whether the tweet is considered Easy (E) or Hard (H) based on its AVG.

6 Experiments and Evaluation

In this section we describe our experiments and results when training with *OLID* + *SOLID* data compared to just *OLID* on the *OLID* test set.

6.1 Experimental Setup

We used BERT and FastText models from the semi-supervised annotation setup to estimate the improvements in performance when training on *OLID* + *SOLID* . We kept the same FastText models, but we tried to improve the performance of BERT via fine-tuning. We achieved improvements for Levels B and C by upsampling the underrepresented classes: for all of the classes, we sampled K instances, where K is the number of instances for the most frequent class. We further removed the warm-up in Levels B and C as this yielded better results for Levels B and C.

When training on *SOLID* along with the supervised training dataset, we used different schemes for combining the two sources of supervision, *OLID* and *SOLID* , as well as different thresholds for the confidence of the instances in *SOLID* . The models in this set of the experiments were fine-tuned on the 10% validation split of the training dataset used during co-training.

The FastText model used an external command-line tool without any control over the training data for the separate epochs. Therefore, we merged the training split of *OLID* and *SOLID* , randomly shuffled them, and evaluated the same models with the new combined dataset.

For BERT, due to the computational requirements of the Transformers, we subsampled 20,000 tweets from *SOLID* data in Level A. Including more semi-supervised instances did not yield any performance improvements. We further balanced the *SOLID* and the *OLID* datasets in Levels B and C by oversampling the underrepresented classes. During training, we used *SOLID* in the first epoch and the supervised training set in the next two following epochs for Level A. In Level A, the training split of *OLID* was already sufficient to train a string Transformer model. In other words, the *SOLID* dataset served only for initial fine-tuning because using it later in the training process after training with *OLID* yielded worse results. For Levels B and C, we trained for two epochs with the training split of *OLID* and then for one epoch with *SOLID* . At Levels B and C, we observed that using *SOLID* in the initial training epochs and then fine-tuning with *OLID* , did not yield any performance benefits. Furthermore, training with the *OLID* first and then using *SOLID* for the final epochs, yielded substantial performance improvements. In Levels B and C, *OLID* had a small number of training instances, which provided a good initial training signal, but when used in the final training epochs, caused the model to overfit to a specific region without being able to generalize on the test split.

For both models, we selected the most confident examples based on the average and the standard deviation scores. We chose only the instances with an average confidence score for each level as follows:

Level A: $avg(\text{OFF}) < 0.2 \cup \ avg(\text{OFF}) > 0.7$

Level B: $avg(\text{UNT}) < 0.3 \cup avg(\text{UNT}) > 0.7$

Level C: $avg(\text{IND}) > 0.8 \cup avg(\text{GRP}) > 0.7 \cup avg(\text{OTH}) > 0.65$

6.2 Results

In this section we describe our results for the experiments described in the previous section based on combining *SOLID* with *OLID* in the training phase. The results are shown in Table 5,

First, we observe that the results for Level A are improved only for the FastText model. At this level, we find that the BERT model already achieved high performance as the supervised training dataset is larger. As a result, including more semi-supervised data did not improve the model's performance. On the other hand, the FastText model, which had the worst performance at this level, managed to improve its accuracy with a large margin when given *SOLID* along with the supervised training set. Our findings are also in line with the study of Longstaff et al. (2010), who observed that democratic co-training performed well when the initial classifier's accuracy was low.

For Level B, the *OLID* training dataset is smaller, and the task is more complex. We observe that including *SOLID* yields performance improvements for both models. We achieve a modest improvement for BERT and a large margin of improvement of 0.121 points for FastText.

Finally, in Level C, the supervised *OLID* dataset is even smaller and the complexity of the subtask is more pronounced, particularly due to three possible labels. Interestingly, including *SOLID* in the FastText model does not yield better results. This might be due to the fact the model already achieves performance on par with the BERT model, and as already stated, democratic co-training performs well when the initial classifier's accuracy is low. On the other hand, the *SOLID* data helps the BERT model and by a large margin of 0.054 points absolute.

Dataset	BERT	FastText
	Level A	
OLID	**0.816**	0.662
+*SOLID*	0.809	**0.720**
	Level B	
OLID	0.712	0.470
+*SOLID*	**0.735**	**0.591**
	Level C	
OLID	0.589	**0.590**
+*SOLID*	**0.643**	0.515

Table 5: Macro-F1 score on the *OLID* test set for the BERT and the FastText models with and without using *SOLID* during training.

7 Discussion

In the experiments and results section, we described our performance using the large semi-supervised dataset as additional training data on the *OLID* test set. However, *OLID* is rather small, particularly for Levels B and C. Therefore, we also annotated a portion of our held-out 3 million tweets in order to be able to analyze the performance in more detail. In particular, we focused on tweets that the model predicted to be Offensive. First, all co-authors of this paper annotated 48 tweets that were predicted to be OFF in order to measure inter-annotator agreement (IAA) using $P_0 = \frac{agreement\_per\_annotation}{total\_annotations*num\_annotators}$. We found IAA to be *0.988* for Level A, with an almost perfect agreement for OFF/NOT. The IAA for Level B was *.818*, indicating a good agreement on whether the offensive tweet was TIN/UNT. Finally, for Level C, the IAA was *0.630*, which is lower but still considered reasonable and Level C is more complicated because it uses a 3-way annotation: IND/GRP/OTH. In addition, a tweet may target more than one label (e.g. both an individual and a group) but only one label can be chosen.

After having observed high IAA, we continued to annotate additional offensive tweets for analysis with a single annotation per instance. We divided our Level A data into four portions based on the models confidence where 0 stands for NOT and 1 is for OFF:

- *if* BERT $\geq .8 \wedge$ PMI $=$ OFF $\wedge$ FT $=$ OFF $\wedge$ GRU $\geq .8$ *then* ***Easy*** **OFF** [284 tweets]
- *if* BERT $\geq .5 \wedge$ PMI $=$ OFF $\wedge$ FT $=$ OFF $\wedge$ GRU $\geq .5$ *then* ***Hard*** **OFF** [835 tweets]
- *if* BERT $< .5 \wedge$ PMI $=$ NOT $\wedge$ FT $=$ NOT $\wedge$ GRU $< .5$ *then* ***Hard*** **NOT** [278 tweets]
- *if* BERT $\leq .2 \wedge$ PMI $=$ NOT $\wedge$ FT $=$ NOT $\wedge$ GRU $\leq .8$ *then* ***Easy*** **NOT** [2500 tweets]

We annotated a total of 1,397 tweets for level A. The portions annotated based on confidence are shown above in square brackets. In addition, in order to properly test Level A (where we only labeled offensive tweets), we also took a random set of 2,500 *Easy* NOT tweets, where we mimic the ratio found in the *OLID* dataset of 2.6 OFF tweets to NOT tweets. The test sizes are shown in Table 3. Of the 1,397 tweets annotated, 307 were

#	Type	Model Prediction	Tweet	Gold Label
1	Easy	OFF	this job got me all the way fucked up real shit	OFF UNT
2	Easy	OFF	@USER It's such a pain in the ass	OFF UNT
3	Easy	OFF	wtf ari her ass tooo big	OFF TIN IND
4	Easy	NOT	This account owner asks for people to think rationally.	NOT
5	Hard	OFF	It sucks feeling so alone in a world full of people	NOT
6	Hard	OFF	@USER We are a country of morons	OFF TIN GRP
7	Hard	NOT	Hate the sin not the sinner...	NOT
8	Hard	NOT	Somebody come get her she's dancing like a stripper	OFF TIN IND

Table 6: Example tweets from the *SOLID Analysis* dataset and the four subsets it is composed of. Shown are the difficulty of each subset, the ensemble model prediction for the examples in each subset, an example tweet's text, and the gold label for that example.

Type	Model Prediction	Gold Label OFF	NOT	Total
easy	OFF	275	9	284
easy	NOT	0	2,500	2,500
hard	OFF	670	165	835
hard	NOT	145	133	278
Total		**1,090**	**2,807**	**3,897**

Table 7: Statistics about the *SOLID Analysis* dataset and the four subsets it is composed of. Shown are the difficulty of each subset, the ensemble model prediction for the examples in each subset, and the gold label distribution.

determined to be NOT. We annotated an additional 1,922 OFF tweets to use for levels B and C [10] where level C is a subset of the TIN tweets from B.

In total, we annotated 3,319 tweets, and we selected 2,500 random NOT tweets for a total of 5,819 tweets in our Analysis dataset. In all cases, all three levels were annotated and the decision of whether a tweet in Level B/C is *Easy* or *Hard* is still based on its Level A confidence. In the future, we plan to have a larger single set for all levels.

Table 6 shows some example tweets including whether they are *Easy* OFF/NOT (lines 1-4) or *Hard* OFF/NOT (lines 5-8) and Table 7 shows statistics about the dataset. In particular, note that determining the labels for the *Hard* examples is not simple to decide and the model does make incorrect predictions such as in lines 5 and 8. In fact, 25% of the *Hard* OFF tweets that we annotated were determined to be NOT. In contrast, less than 3% of the *Easy* OFF tweets were determined to be NOT.

Model-Dataset	Full	Easy	Hard
Level A			
BERT-*OLID*	0.914	0.987	0.524
+*SOLID*	**0.916**	**0.989**	**0.531**
FastText-*OLID*	0.865	0.944	0.538
+*SOLID*	**0.871**	**0.949**	**0.539**
Level B			
BERT-*OLID*	0.568	0.637	0.516
+*SOLID*	**0.627**	**0.722**	**0.572**
FastText-*OLID*	0.374	0.457	0.314
+*SOLID*	**0.494**	**0.597**	**0.434**
Level C			
BERT-*OLID*	0.635	0.638	**0.632**
+*SOLID*	**0.670**	**0.721**	0.619
FastText-*OLID*	0.372	0.364	0.381
+*SOLID*	**0.537**	**0.542**	**0.522**

Table 8: Experiments on the *SOLID Analysis* dataset, as well as on the easy and on the hard subsets thereof. Shown are macro-F1 scores.

Next, we performed our experiments as described in Section 6.1 on the *SOLID* analysis set. Our goal is two-fold: 1) We focus on the *Easy* vs. *Hard* examples to gain better insight into why some tweets are easier to predict as offensive than others, and 2) *OLID* is small (particularly for B and C). Therefore, showcasing the performance on a larger amount of data can indicate the stability of the test set. The results are shown in Table 8. In almost

[10]The tweets in the analysis set for Level A differ from those in Levels B and C to accommodate the Semeval-2020 Evaluation (Zampieri et al., 2020) task schedule.

all cases including *SOLID* with *OLID* provides an improvement in performance.

The Level A results on the *Easy* tweets achieves almost 99% macro-F1, but for *Hard*, they are below 54%. Furthermore, using *SOLID* provides a nice improvement for the BERT model on the *Hard* tweets which was not evident in *OLID* the test set in Table 5. To provide further insight into why the results are so high for *Easy* OFF tweets in Level A we implemented a a simple curse word baseline using only the absence or presence of 22 curse words such as *fuck*, *bitch*, and *nigga*. A full list of the curse words used in the baseline can be found in Appendix A.1. We found that the majority of the *Easy* tweets can be classified correctly with this baseline with 93.6% F-score. We note that for Level A the full results are much better than on the OLID dataset in Table 5. We expect this is due to our selection of tweets for analysis and that it indicates there are more *Easy* tweets in the analysis set. In contrast, the curse word baseline is not as effective on the hard examples, achieving a macro-F1 score of 58%. These tweets are offensive due to other language such as negative biases rather than the appearance of a curse word such as examples 6 and 8 in Table 6.

The difference between *Easy* OFF/NOT and *Hard* OFF/NOT tweets is less pronounced for Levels B and C. In level B the gap is up to 16% absolute. We believe that the curse word imbalance in *Easy* tweets effects level B as well because UNT tweets tend to have curse words, which is evident by the larger number of UNT tweets in the test set such as the example in line 2 of Table 6. In level C, there is usually no difference; TIN tweets may have curse words but that does not necessarily impact who the target is. Finally, the results for all levels vary greatly for both models compared to the results on the *OLID* test set. This indicates that further work is needed to generate a large stable test set.

8 Conclusion and Future Work

We present *SOLID* , a large-scale semi-supervised training dataset for offensive language identification containing nine million English tweets created using an ensemble of four different models. To the best of our knowledge, *SOLID* is the largest dataset of its kind. *SOLID* is freely available[11] to the research community, and it was released as training material for the English track of SemEval-2020 Task 12 (OffensEval 2020) (Zampieri et al., 2020).

We have shown that including *SOLID* yields noticeable performance improvements for levels B and C on the *OLID* test set. Furthermore, we analyze *Hard* and *Easy* tweets and identify that detecting the *Hard* offensive tweets (such as those that do not contain curse words) is still an open challenge. In the future, we hope to release a new larger test set for *OLID* and *SOLID* .

[11] https://sites.google.com/site/offensevalsharedtask/solid

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63.

Pete Burnap and Matthew L Williams. 2015. Cyber hate speech on Twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

Yi-Ling Chung, Elizaveta Kuzmenko, Serra Sinem Tekiroglu, and Marco Guerini. 2019. CONAN - COunter NArratives through nichesourcing: a multilingual dataset of responses to fight online hate speech. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2819–2829, Florence, Italy. Association for Computational Linguistics.

Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving cyberbullying detection with user context. In *Advances in Information Retrieval*, pages 693–696. Springer.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the Web Conference (WWW)*.

Paula Fortuna, José Ferreira, Luiz Pires, Guilherme Routar, and Sérgio Nunes. 2018. Merging datasets for aggressive text identification. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC)*, pages 128–139.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys (CSUR)*, 51(4):1–30.

Spiros V Georgakopoulos, Sotiris K Tasoulis, Aristidis G Vrahatis, and Vassilis P Plagianakos. 2018. Convolutional neural networks for toxic comment classification. *arXiv preprint arXiv:1802.09957*.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Georgios Kostopoulos, Stamatis Karlos, and Sotiris Kotsiantis. 2019. Multiview learning for early prognosis of academic performance: A case study. *IEEE Transactions on Learning Technologies*, 12(2):212–224.

Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC)*.

Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting tweets against blacks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Brent Longstaff, Sasank Reddy, and Deborah Estrin. 2010. Improving activity classification for health applications on mobile devices using active and semi-supervised learning. In *2010 4th International Conference on Pervasive Computing Technologies for Healthcare*, pages 1–7. IEEE.

Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. 2019. Overview of the Hasoc Track at Fire 2019: Hate Speech and Offensive Content Identification in Indo-european :anguages. In *Proceedings of the 11th Forum for Information Retrieval Evaluation*, pages 14–17.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Tawfik A Mohamed, Neamat El Gayar, and Amir F Atiya. 2007. A co-training approach for time series prediction with missing data. In *International Workshop on Multiple Classifier Systems*, pages 93–102. Springer.

Hamdy Mubarak, Ammar Rashed, Kareem Darwish, Younes Samih, and Ahmed Abdelali. 2020. Arabic Offensive Language on Twitter: Analysis and Experiments. *arXiv preprint arXiv:2004.02192*.

Nedjma Ousidhoum, Zizheng Lin, Hongming Zhang, Yangqiu Song, and Dit-Yan Yeung. 2019. Multilingual and multi-aspect hate speech analysis. 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)*, pages 4674–4683, Hong Kong, China. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Zeses Pitenis, Marcos Zampieri, and Tharindu Ranasinghe. 2020. Offensive Language Identification in Greek.

Peter D Turney and Michael L Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, 21(4):315–346.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 shared task on the identification of offensive language. In *Proceedings of the GermEval 2018 Workshop (GermEval)*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technology (NAACL-HLT)*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the type and target of offensive posts in social media. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1415–1420, Minneapolis, Minnesota. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020). In *Proceedings of SemEval*.

Yan Zhou and Sally Goldman. 2004. Democratic co-learning. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 594–602. IEEE.

A Appendices

A.1 Data Collection and Analysis

We described our method for collecting tweets in Section 5 of the main paper. We collect tweets using the most frequent English words based on the large monolingual Project Gutenberg corpus (https://en.wiktionary.org/wiki/Wiktionary:Frequency_lists#Project_Gutenberg). Table 9 shows the top-20 most frequent words in the corpus and their frequency which we used to collect tweets. The normalized value is the percentage of the total frequency for all top 20 words. We randomly pick a number between 0 and 1, and choose the word based on the normalized value. For example, .45 would be "and"

In Section 4 we discussed the simple curse word baseline used to analyze the *Easy* OFF/NOT tweets. Table 10 lists the 22 curse words used in the baseline.

word	**frequency**	**normalized**
the	56,271,872	0.20
of	33,950,064	0.32
and	29,944,184	0.43
to	25,956,096	0.52
in	17,420,636	0.58
i	11,764,797	0.63
that	11,073,318	0.67
was	10,078,245	0.70
his	8,799,755	0.73
he	8,397,205	0.76
it	8,058,110	0.79
with	7,725,512	0.82
is	7,557,477	0.85
for	7,097,981	0.87
as	7,037,543	0.90
had	6,139,336	0.92
you	6,048,903	0.94
not	5,741,803	0.96
be	5,662,527	0.98
her	5,202,501	1.00

Table 9: The top-20 frequent English words. The last column is the normalized value based on the total frequency of all the top words. The random number generated between 0 and 1 will determine which word is chosen.

ass	arse	wtf	lmao
fuck	bitch	nigga	nigger
cunt	effing	shit	hell
damn	crap	bastard	idiot
stupid	racist	dumb	f*ck
pussy	dick		

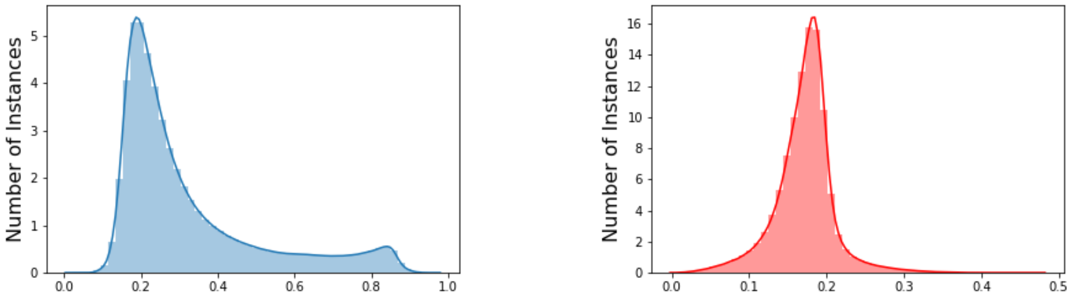
Table 10: The 22 common offensive terms used in the curse word baseline.

A.2 *SOLID* Scores Distribution

Figure 1 presents the distribution of the average and the std scores for Level A produced for each instance, as described in Subsection 5.2. We can observe that the average confidence has one large peak around 0.2 and represents that the models highly agree on a lot of instances from the majority class NOT. We can also find a small peak near the average of 0.8, which corresponds to the consolidated prediction of the OFF class. The standard deviation of the predicted confidence scores is distributed mainly between 0.1 and 0.2.

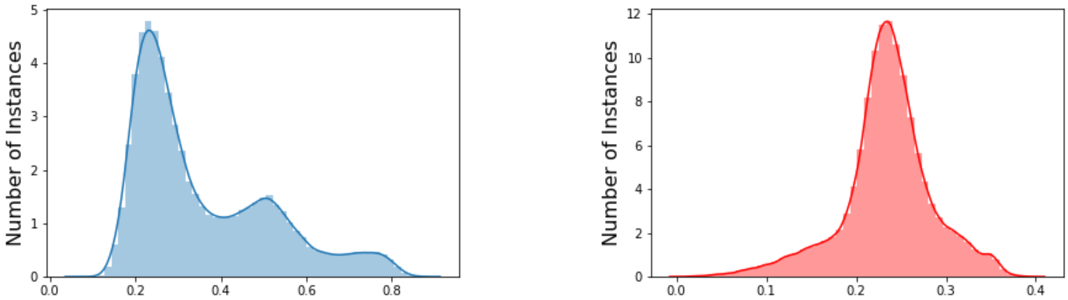
Figure 2 presents the scores for Level B. Apart from the peaks around 0.2 and 0.8, representing models' aggregated prediction for the UNT and TIN classes, we also find that there is a large number of predictions that are clustered around the average confidence of 0.5. The latter indicates that there is a substantially large number of instances that the ensemble is not confident about. This might be a result of the more significant differences in the models' confidences, as indicated in Subfigure 3c.

Finally, Figure 3 presents the distribution of the average and the std scores for Level C for each of the separate classes. We can conclude that the ensemble is usually more confident about instances from the IND class and rarely predicts instances from the other classes.



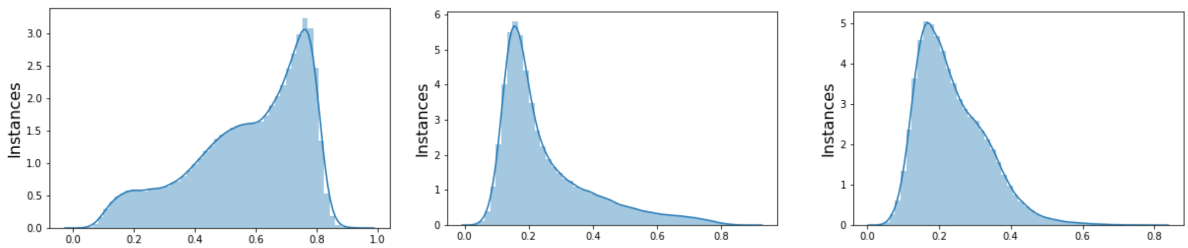
(a) Average Confidence (b) Confidence STD

Figure 1: Distribution of the aggregation scores produced for each instance as over the four confidence scores of each supervised model for Level A.

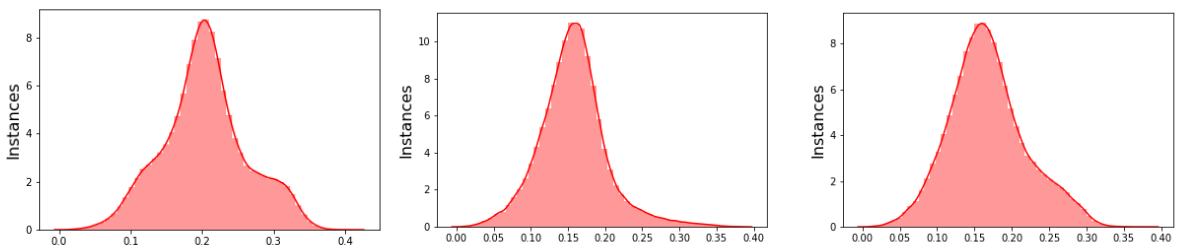


(a) Average Confidence (b) Confidence STD

Figure 2: Distribution of the aggregation scores produced for each instance as over the four confidence scores of each supervised model for Level B.



(a) Average Confidence, class=IND (b) Average Confidence, class=GRP (c) Average Confidence, class=OTH



(d) Confidence STD, class=IND (e) Confidence STD, class=GRP (f) Confidence STD, class=OTH

Figure 3: Distribution of the aggregation scores – average (top) and STD (bottom), produced for each instance as over the four confidence scores of each supervised model for Level C.