Detecting Sensitive Content in Spoken Language

Rahul Tripathi
Amazon
Seattle, USA
rahtripa@amazon.com

Balaji Dhamodharaswamy
Amazon
Seattle, USA
dhbalaji@amazon.com

Srinivasan Jagannathan
Amazon
Seattle, USA
sjaganna@amazon.com

Abhishek Nandi
Amazon
Seattle, USA
annandi@amazon.com

***Abstract*—Spoken language can include sensitive topics including profanity, insults, political and offensive speech. In order to engage in contextually appropriate conversations, it is essential for voice services such as Alexa, Google Assistant, Siri, etc. to detect sensitive topics in the conversations and react appropriately. A simple approach to detect sensitive topics is to use regular expression or keyword based rules. However, keyword based rules have several drawbacks: (1) coverage (recall) depends on the exhaustiveness of the keywords, and (2) rules do not scale and generalize well even for minor variations of the keywords.**

Machine learning (ML) approaches provide the potential benefit of generalization, but require large volumes of training data, which is difficult to obtain for sparse data problems. This paper describes: (1) a ML based solution that uses training data (2.1M dataset), obtained from synthetic generation and semi-supervised learning techniques, to detect sensitive content in spoken language; and (2) the results of evaluating its performance on several million test instances of live utterances. The results show that our ML models have very high precision ($>>$90%). Moreover, in spite of relying on synthetic training data, the ML models are able to generalize beyond the training data to identify significantly higher amounts ($\sim$ 2x for Logistic Regression, and $\sim$ 4x-6x for a Neural Network models such as Bi-LSTM and CNN) of the test stream as sensitive in comparison to a baseline approach using the training data ($\sim$ 1 Million examples) as rules. We are able to train our models with very few manual annotations. The percentage share of sensitive examples in our training dataset from synthetic generation using templates and manual annotations are 98.04% and 1.96%, respectively. The percentage share of non-sensitive examples in our training dataset from synthetic generation using templates, automated labeling via semi-supervised techniques, and manual annotations are 15.35%, 83.75%, and 0.90%, respectively. The neural network models (Bi-LSTM and CNN) also use lower memory footprint (22.5% lower than baseline and 80% lower than Logistic Regression) while giving improved accuracy.

***Index Terms*—machine learning, sensitive content, neural networks, synthetic training data**

I. Introduction

Voice services like Alexa, Google Assistant, Siri, etc. enable voice driven applications, engaging conversations, and interactive question & response sessions. At the same time, because there is effectively no limit to the types of interactions with the users, there is a risk of negative emotions, spread of offensive content, and exposure to age-inappropriate information. We refer to any content that can create negative surprise to users as *sensitive* content. Sensitive content, over the long run, can negatively impact the user experience of conversing with a voice service. This paper addresses the kind of *textual* sensitive content found in spoken language that may occur in user interactions with voice services and provides a solution to detect such content in real time. A list of sensitive categories that we want to detect along with a brief description is below:

- *Profanity*: profane words and phrases.
- *Insult and Personal Offense*: statements and leading questions that are insulting to any person or entity.
- *Geo-Political*: contentious geo-political topics, actively debated in the news.
- *Sexual and Anatomical*: explicit, NSFW, sexual content for mature audiences.
- *Weapons, Violence, and War*: explicit graphical violence with excessive detail and gore.
- *Race, Minorities, and Gender*: hate speech, offensive or stereotypical content.
- *Religion, Faith & Spirituality*: spiritual guidance and opinions on religion.

We are interested in detecting whether a given utterance text belongs to any of the above sensitive categories. In this work, we do not address the fine-grained category-based classification of sensitive content, $i.e.$, we only explore the binary classification of whether a particular utterance text contains sensitive content or not. We develop ML models for this binary classification problem.

ML models, particularly DNNs (deep neural networks), require large volumes of training data. However, sensitive topics can occur very sparsely in language. Hence, obtaining training data for sensitive content can present a significant challenge. We present a template-based approach to generate training data and describe its use to train machine learning models. We also outline a statistical method to ensure that the synthetic training data meets acceptable data quality standards.

We used the template approach to bootstrap the ML models, and used active learning techniques to improve them. After bootstrapping an initial set of models with synthetic data, we relied on automated labeling of live utterances to generate additional non-sensitive examples for the training data. Our approach is a significant innovation in developing NLP classifiers with sparse data classes. Another benefit of our approach is that it reduces the need for manual review of live utterances to generate labels needed for the training data. Reducing the need for manual review of live utterances not only decreases costs, but also is beneficial for privacy and security in the context of voice services.

II. Related Work

Sensitive content, and more particularly offensive content, in online discussion communities has been explored in recent

978-1-7281-4493-1/19/$31.00 ©2019 IEEE
DOI 10.1109/DSAA.2019.00052

works. Cheng et. al. [5] studied various forms of antisocial behavior in online communities, the factors that contribute to the emergence of such behavior, and developed a classifier to predict antisocial behavior based on a user's first ten posts. In [4], antisocial behaviors such as *trolling* are attributed to negative moods and discussion context, and a predictive model is developed to identify whether a user, given indicators such as user's mood and discussion context, will behave unacceptably (as per community guidelines) in a given post. Warner et. al. [18] present an approach to detect hate speech in online text by treating it as a word sense disambiguation problem: it is the association of certain stereotypical words and not merely their presence that indicates a hate speech. A comprehensive survey on automated hate speech detection is reported in [17]. Yenala et. al. [21] addressed the detection of inappropriate/offensive content in search queries and user-conversations in messaging systems. Similar to our work, they also found DNN models to be the best performing for inappropriate/offensive content in these two tasks. Our work differs from theirs in that (a) we focus on utterance texts in spoken language during their interaction with voice services, (b) our training datasets are much larger (in millions as opposed to thousands)–we describe a method to generate such large training datasets, and (c) we report performance on live utterances in a voice service.

III. Challenges in building an automated sensitivity detection solution

Sensitivity in conversational text is difficult to detect because it requires looking into the context and the intent of the conversation. Regular expression-based solutions, such as *keyword or phrase based rules* that merely look at presence of certain words in a specific order, are not sufficiently accurate to detect different categories of sensitive content. For example: *should women be allowed to vote* is a sensitive text whereas *in what year have women been allowed to vote* is a non-sensitive text, even though both of them have a considerable overlap of words. Rule based systems do not generalize well and exceptions to rules can make them complex, difficult to maintain, and hard to scale. Hence, we were motivated to seek machine learning based solutions for this problem. For training the ML models, we need sufficient amount of ground-truthed training data. Certain sensitive categories such as *profanity, insult and personal offense, and sexual* have publicly available datasets that can be leveraged for building a supervised classifier [1], [6], [9]. In our research, we could not find large labeled datasets for other categories such as *geo-political, weapons, violence & war, race, minorities & gender, and religion, faith & spirituality*. Furthermore, the amount of sensitive content in day to day conversations with voice services may be lower than the amount of non-sensitive content, i.e., sensitive content can be a sparsely occurring class of content. Therefore, creating a large dataset with enough representations of each of the sensitive categories can be a challenging task. For example, if we need 10^4 sensitive examples to train a model and the sensitive content occurs in around 4% of the population, we need to manually review and label around 2.5×10^5 examples in the population to find these sensitive examples, which is not a cost-effective process. We have developed a method to overcome this problem. In Section IV, we explain our approach of creating labeled datasets for building supervised ML models for the sensitive content detection problem.

IV. Dataset Construction and Workflow

A. Methodology

Supervised machine learning classifiers for sensitivity detection in user interactions with digital assistants require substantial amount of labeled (sensitive and non-sensitive) training examples to avoid overfitting. This data requirement is even higher for neural network models that have large ($50K$ or more) number of parameters. In our model explorations, we experimented with different proportions ($25:75$, $50:50$, $90:10$) of sensitive and non-sensitive examples in our training data set and compared precision[1] of the initial models across those choices on a small test dataset. We found that having close to $50:50$ split between sensitive and non-sensitive content in the training dataset gives better precision than any other choices. So we embarked on creating a training dataset with almost equal mix of sensitive and non-sensitive data as the primary source.

We initially tried external, publicly available datasets from sources such as Reddit [1] and Kaggle [9] to train our models. However, we found that the trained models had extremely low coverage on sensitive categories such as *geo-political, weapons, violence & war, race, minorities & gender, and religion, faith & spirituality*, which were not represented in these datasets but are very relevant to our test dataset containing conversation texts. Therefore, we decided to create the training and the validation datasets from within the domain of our voice service. Along this direction, we next tried a keyword-based search in conversation texts (e.g., search of all occurrences of the word *evil*) to identify potentially sensitive content for each category. However, our yield of the sensitive class with this keyword based approach was very low and so the cost of manually reviewing and labeling the training data for sensitive class was very high. This led us to explore an approach to generate synthetic training data based on actual spoken language. We started with a few hundred manually identified spoken language examples of each kind (sensitive, non-sensitive) and then generated variations of those using a template-based method. In this approach, we take a base text (called a *template*), identify entities and keywords in the template (called *slots*), and then generate variations by changing the entities and keywords (e.g., synonyms and antonyms) in the text.

Methods of synthesizing new data to overcome class imbalance or to reduce model overfitting has found many applica-

[1]For our models, we focused more on precision than other traditional measures such as recall or F1-score. This is because the perceived impact of incorrectly classifying a non-sensitive content as sensitive is higher in a data population where the sensitive class is sparsely populated, than incorrectly classifying a sensitive content as non-sensitive.

tions in machine learning. For imbalanced class problems, the Synthetic Minority Over-sampling Technique (SMOTE) [3] is shown to improve the accuracy of classifiers for the minority class. SMOTE allows creation of synthetic examples of the minority class by interpolating between numeric features of existing examples of the same class. However, this method is not helpful for textual data since interpolation is likely to generate non-sensical texts. In image applications, data augmentation using label-preserving transformations are often used to reduce overfitting on image data (e.g. [12]). The closest to our usage of templates for synthetic data generation in natural language applications is Dixon et al. [7]. In that paper, the authors generated a synthetic *test* dataset using templates to evaluate unintended bias in machine learning models. To the best of our knowledge, templates have not been used previously for augmenting the *training* dataset in applications related to detecting sensitive or offensive content.

B. Data collection

We collected positive (sensitive or minority class) and negative (non-sensitive or majority class) examples from spoken language examples using the approach below. As we discussed earlier, sensitive content can be sparse, traditional sampling methods (e.g., uniform sampling) to find them may be futile. So, we relied on keyword-directed searches such as using *power words* or *negative-sentiment* bearing words to increase our yield of identifying sensitive content from the interested categories. This ensured that the data being annotated is rich in sensitive content from these rare sensitive categories. We then manually annotated around five thousand examples as sensitive or non-sensitive according to the sensitivity classification outlined in Section I. We developed a sentence similarity tool and ran it to identify content that are similar to the content we already annotated to be sensitive. The tool used average WordNet [8], [14] synset based Wu-Palmer similarity [19] of most similar words in two sentences to calculate a sentence similarity score. We used a minimum threshold of 0.80 sentence similarity score to identify similar sentences. After iterating over this process multiple times, we identified more examples of sensitive content across various rare sensitive categories that could be used for template generation. Each of these examples were structurally unique (e.g., *Adam is a moron* and *Geena is an idiot* would not be considered structurally different from each other). These included sensitive content from categories such as sexual, insult, violence, racism, and profanity. Overall, we were able to collect around a thousand examples with a ratio of $3:1$ sensitive:non-sensitive classes from these sources.

C. Creating templates and generating variations from them

For each of the collected examples, we then manually identified keywords and entities (called *tokens* or *slots*), tagged them into token groups, and created templates based on them. For instance, for a text such as *are you not allowed to tell the truth about XYZ*, we marked "tell", "truth", and "XYZ" as tokens, identified tell and truth as [synonym] and "XYZ" as [public_figure] token groups, and created the template: *are you not allowed to "tell"[synonym] the "truth"[synonym] about "XYZ"[public_figure]*. We also maintained an *entity* database for each token group and a *name-variation* database for select individual names (e.g., entity database for [public_figure] includes a list of influential persons and the name_variation database holds various common names of a select individual, e.g., Smith, John Smith, J. Smith). For synonyms and antonyms variations, we developed a Wordnet [8], [14] synset based tool that looks at the parts of speech of a given text to find the right variations. Next, for each template, we varied entities/keywords according to their token group—synonyms and antonyms of a keyword based on its part of speech in the text and possible entities using entity and name-variation databases for other token groups—to generate variations from the template. Some examples of templates and their generated variations are listed in I.

TABLE I
TEMPLATES AND GENERATED VARIATIONS

Template	**Variations**
why do people "hate"[antonym] "XYZ"[political_figure]	why do people hate ABC why do people love DEF
are you not allowed to "tell"[synonym] the truth about "XYZ"[political_figure]	are you not allowed to say the truth about ABC are you not allowed to state the truth about DEF

D. Semi-supervised and Active Learning methods for data collection

Apart from using templates to augment and enrich our training dataset, we relied on other means such as active learning to continuously bring in new data into training set using existing data. For this, we first trained our models with the template based training data. We then replayed an unlabeled dataset, obtained from spoken language examples, through three different models (explained in Section V) for inference. Next we identified the content on which all three models had a low confidence score (≤ 0.3). We automatically labeled this data as non-sensitive because of (1) with a sparsely populated sensitive class, the dataset was likely to have a high % of non-sensitive content, and (2) based on the consensus of three different models, the likelihood of sensitive content was very low. This technique of assigning labels to previously unlabeled data and using it for model training is called *semi-supervised learning* [2], [20]. Finally, we brought this newly labeled data into the training set and repeated this cycle a few times.

We measured the accuracy of our models on the same unlabeled data by manually annotating a sampled set at different confidence thresholds. This helped us in choosing the right threshold required to meet a high level of precision (see Section VI-B), for the models.

TABLE II
TRAINING DATASET

Number of sensitive texts	1054351
Number of non-sensitive texts	1122384
Number of words in vocabulary	51300

E. Quality control

For each of the two classes (sensitive and non-sensitive), we took a uniformly random (with replacement) sample of 270 texts from the generated dataset and manually reviewed to look for texts that: (a) did not fit the class or (b) appeared meaningless. If we found > 3 such examples, we corrected data generation process to avoid those types of texts. We repeated these sampling and quality checks until we had ≤ 3 examples in a random sample of size 270. This is because a random sample of 270 texts for quality checks is sufficient to guarantee $\leq 3\%$ errors with 95% confidence (Lemma 1).

Lemma 1. *Suppose the fraction of bad texts in a collection of texts is p, where $p > 3\%$. For any $n \geq 264$, if we draw a uniformly random (with replacement) sample of size n, then the probability of finding 3 or less errors is less than 5%.*

At the end of this process, we were able to generate around a million examples each for sensitive and non-sensitive classes. These constituted our training dataset. The percentage share of sensitive examples in our training dataset from synthetic generation using templates and manual annotations are 98.04% and 1.96%, respectively. The percentage share of non-sensitive examples in our training dataset from synthetic generation using templates, automated labeling via semi-supervised techniques, and manual annotations are 15.35%, 83.75%, and 0.90%, respectively. Table II summarizes the key characteristics of our training dataset.

V. MACHINE LEARNING MODELS

We experimented with different classifier models and training methods to evaluate which ones would be promising for identifying sensitive content with high precision. For example, we evaluated logistic regression, gradient boosting, xgBoost, random forest, recurrent neural networks, convolutional neural networks, and ensemble methods. In this paper, we present details on three particular approaches–logistic regression, a BiLSTM model, and a convolutional neural network.

Most of our ML models use GloVe [15] word embedding to transform texts into numeric vectors of 300-dimensions as the initial features of the input data. In our neural network models (CNN and BiLSTM), we start with the pre-trained embedding and then fine-tune the word vectors during the model training process.

A. Logistic Regression Model

In this logistic regression model, standard data processing steps including a stemmer is applied to transform text into a stream of tokens. Next a series of transformations are applied to represent the text as a numeric feature vector. Some transformations used are binary indicators for unigrams, bigrams, trigrams and 4-grams, and count based features that are scaled to [0,1]. The model uses sparse multinomial logistic regression learning algorithm and is trained using stochastic gradient descent with AdaGrad optimizer.

B. Bi-Directional LSTM Model

We used the Bi-LSTM model developed for detecting offensive content in Khatri et al. [10], and trained it on the synthetic data discussed above. We used this model to identify non-sensitive training data as described in Section IV-D.

Each word in the input text is transformed to a numeric vector using pre-trained Glove word vectors by passing through an embedding layer. The weights of this layer are hyper-parameters that are fine-tuned during the learning stage. Next the weights are passed as inputs to two LSTM layers: one layer acts on word vectors from left to right and the other from right to left (hence Bi-directional). Assume that the input word vectors are denoted by X_1, X_2, ..., X_{last}. The LSTM layer is a recurrent neural network. At time step t, the left-to-right LSTM takes input (a) the t'th vector from left X_t and (b) the previous step hidden state h_{t-1} and outputs the new hidden state h_t, whereas the right-to-left LSTM takes input (a) the t'th vector from right $X_{\text{last}-t+1}$ and (b) the hidden state $h'_{\text{last}-t+2}$ and outputs hidden state $h'_{\text{last}-t+1}$. The hidden state h_0 and $h'_{\text{last}+1}$ are initialized to random values with zero mean. Each recurrent unit (LSTM cell) is itself composed of 4 gates: forget, input, cell, and output gates. In LSTM, the cell state (output of cell gate) acts as a memory to hold information about the data seen so far while the other layers help to manipulate this information.

Next, we apply a max-pooling operation on the concatenated hidden states $h_t \circ h'_t$ computed over different steps t. This is an element-wise max operation on the hidden states of Bi-LSTM over various t. The intuition is that each element of a concatenated state $h_t \circ h'_t$, for any t, captures certain features of the input based on the words up to X_t (in either direction) and so max-pooling allows to consider the best features learned across all units t. The output from the LSTM layer is then fed as input to a fully-connected layer that in turn connects to the output layer. The output is a 2-node fully connected layer. Using the *softmax* activation, the output is transformed into probabilities for the sensitive and the non-sensitive classes.

The architecture of the Bi-LSTM model is presented in Fig 1.

C. Convolutional Neural Network Model

We also developed a convolutional neural network (CNN) model based on the architecture proposed by Kim et al [11]. CNNs have found numerous applications in computer vision where they are the building blocks of the best known deep neural networks for several image identification tasks. Kim [11] showed that CNNs can achieve state of the art in various NLP tasks including sentiment analysis.

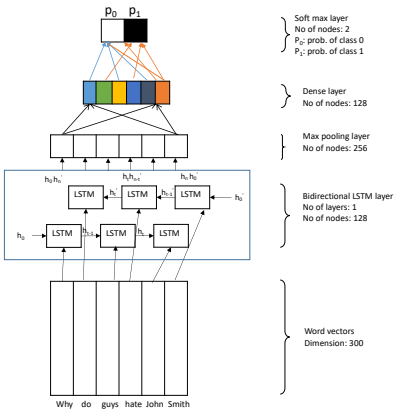

Fig. 1. Bi-LSTM model

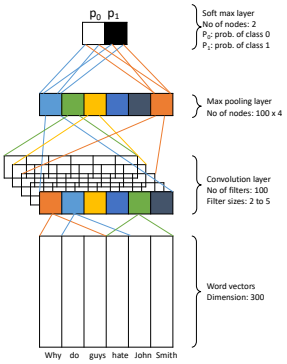

Fig. 2. CNN model

Similar to the Bi-LSTM model, we process input text into a sequence of embedding vectors: one vector for each word in the text. Next a series of filters of various sizes are applied on these vectors. Each such filter of size s results in the application of a convolution operation involving embedding vectors in a sliding window of size s. For example: a filter of window size 3 would take three consecutive vectors, multiply each element with a corresponding weight, add all the terms to a single scalar value, and finally apply a non-linear activation function such as *tanh* or *relu* on the scalar. A particular filter when applied to each possible window of consecutive vectors (words) yields a feature map $\vec{c} = [c_1, c_2, ..., c_{n-s+1}]$, where n is the length of the input sequence and s is the filter size. A max-pooling operation then reduces any feature map to the maximum value $\max\{\vec{c}\}$.

This step is repeated for multiple filters (say 100) and various filter sizes (say 2 to 5). The resulting features are of total size num_filters $\times$ filter_size. The idea behind Max-Pooling is to choose the best feature out of all alternatives. These features are fed to a 2-node fully connected *softmax* layer, which returns probabilities of the sensitive and the non-sensitive classes.

The architecture of the CNN model is presented in Fig 2.

VI. Experiments

A. Baseline Technique

For our experiments, we compare the performance of the ML models against a baseline technique of a purely rule based system. Since the ML models are bootstrapped from synthetic training data, it is conceivable that the ML models do not generalize much beyond this synthetic set. Using the training data as the set of rules in a rule-based sensitive topic detection system therefore provides a good baseline for evaluating the performance of the ML models. The baseline system, by virtue of using a curated set of rules will have a precision of 1, i.e., anything identified as sensitive using this system is guaranteed to be sensitive. The amount of sensitive content detected using this baseline system provides a benchmark for measuring how much additional (or fewer) sensitive content is detected as a result of machine learning. For our experiments, the baseline system had 1,054,351 rules (see Table II).

B. Metrics

The metrics of interest to us are: Precision and Recall. Precision measures the fraction of instances predicted to be positive that are in fact positive whereas recall measures the fraction of positive instances that are correctly predicted as positive. More precisely, if TP, FP, and FN denotes True-Positive, False-Positive, and False-Negative instances, respectively, based on the classification decisions by a model, then its precision = TP/(TP + FP) and its recall = TP/(TP + FN).

For detecting sensitive content, it is desirable to achieve very high precision ($e.g., \gg 90\%$) for the sensitive class. This is because taking incorrect mitigation actions on false positives may impact the user experience. For example, if a conversation that does not include an insult is interpreted as containing an insult, this will lead to a poor conversational experience. Moreover, in a population with sparse distribution of sensitive content, false positives will significantly impact a relatively larger number of non-sensitive texts. For the work presented in this paper, we primarily focused on the precision goal.

For a binary classification problem, we show in Lemma 2 (see Appendix) that precision and recall for one class fully determines the same measures for the other class if the distribution of positives and negatives in the test set are known beforehand. We assume that this distribution is roughly known in the population, and so we focused on these metrics for only the positive (sensitive) class.

Lemma 2. *Suppose A and B denote the two classes for any binary classifier. If p and r denote the precision and recall, respectively, for class A and suppose g denotes the class mix ratio (i.e., the ratio of the size of class A and the size of class B) in the test set, then precision P_B and the recall R_B for class B are given as follows:*

$$P_B = \frac{1}{1 + \frac{(1-r)g}{(1-rg(1/p-1))}} \text{ and } R_B = 1 - rg(1/p - 1).$$

Because our work involves evaluation over test sets that easily exceeds several millions instances to be classified, it was not practical to calculate recall (because estimating the false negatives requires manual review of very large number of instances). We therefore used a yield metric that calculates the amount of sensitive examples detected by the model in each data set. For high precision models, higher yields are a good proxy for higher level of recall, i.e., at high precision levels, increase in yield is positively correlated with increase in recall. To normalize the yield across different days of traffic, we present these yield numbers as a multiple of the yield obtained from the baseline technique.

C. Test datasets

We used two separate test datasets comprised of several million spoken language samples obtained from live interactions on two different days. We fed these test data as input to the ML models, and then estimated the precision by manually reviewing and labeling sets of at least 1100 randomly selected samples (with replacement) each from the sets of test examples predicted as sensitive by the ML models. The sample size was sufficient to get highly accurate estimate of precision. We can show using the Central Limit Theorem that ≥ 1068 samples suffice to estimate precision within ± 0.03 additive error with 95% confidence.

D. Results

Our neural network models (Bi-LSTM and CNN) are implemented in MxNet and are trained on single GPU instances (e.g., $p2$ and $p3$). The training time of these models is significantly lower for $p3$ instances than for $p2$ instances, which shows that improvement in GPU capabilities have a direct impact on the training time of our models. For MxNet 1.2.1, the time per epoch for training Bi-LSTM model on $p2$.xlarge instance is 4.5 mins and on $p3$.xlarge instance is 2.25 mins (when all other experimental settings are the same). The corresponding numbers for CNN model (using Gluon MxNet API) are 6.5 mins and 1.35 mins, respectively.

As shown in Table III, except on one occasion (Day-2 dataset) with CNN model, the ML models in general achieved a precision of at least 0.95 in both datasets. The slightly lower precision of the CNN model is compensated by its relatively higher yield compared to the other two models. The Bi-LSTM model has consistently shown high precision (> 0.97) and high yield (> 4) on both datasets. When calculating precision, we excluded some of the utterances that were marked as positive by the model, but were either unintelligible (e.g., "ask forget me poop flowers") or alternately were borderline offensive (e.g., "sing and dance men are better than women").

As shown in Table IV, the Bi-LSTM and CNN models have the smallest memory footprint of all the techniques, achieving approx. 22.5% savings over the rule-based benchmark, and approx. 80% savings over the Logistic Regression model. The low footprint of the Bi-LSTM and CNN models show the power of neural network based learning. As we add more training data, we do not expect the footprint of the Bi-LSTM and CNN models to increase as only weights are updated.

TABLE III
BENCHMARK ON TEST DATASETS

In the Table below, the results are presented after classifying two separate days of traffic (several million spoken utterances on each day) in a voice service using a baseline technique and three different ML models: Bi-LSTM, CNN, and Logistic Regression. We report the precision calculated by drawing a random sample of 1100 instances from the positive sets identified by the models and manually labeling them as true or false positives. Since the test data has millions of instances and the positive set is sparsely distributed, it was not practical to calculate recall. Instead we rely on a yield metric, i.e., the ratio of amount of sensitive content detected by the model to the amount detected by the baseline technique. At high precision values, higher yield indicates higher levels of model learning. As seen below, all three ML models have higher yields in comparison to the baseline technique, with minimal reduction in precision. The Bi-LSTM model has consistently high precision (> 0.97) and high yield (> 4). The CNN model has the highest yield, but its precision is slightly lower (approx. 0.95) than those of Bi-LSTM and Logistic Regression. The Logistic Regression model has high precision, but its yield is significantly lower than that of the neural network models.

Dataset	Model	Precision	Yield
Day-1	Baseline	1	1
Day-1	Bi-LSTM	0.975	4.27
Day-1	CNN	0.958	5.90
Day-1	LogReg	0.9848	1.92
Day-2	Baseline	1	1
Day-2	Bi-LSTM	0.971	4.43
Day-2	CNN	0.945	6.10
Day-2	LogReg	0.958	1.98

TABLE IV
MEMORY FOOTPRINT

Model	Memory	%Savings (relative to the Baseline)
Baseline	80MB	0
Bi-LSTM	62MB	22.5
CNN	62MB	22.5
LogReg	302MB	-277.5

VII. CONCLUSION AND FUTURE WORK

We discussed the relevance of the sensitivity detection problem for voice services. We identified different categories of sensitive content that should be addressed in such applications. We then presented a ML solution that we have developed. The amount of sensitive content can be sparse when compared to non-sensitive content. Therefore, one of the main challenges in designing an ML solution for this problem is setting up the training and validation datasets. We described a template-based training data generation approach that in conjunction with active learning and semi-supervised techniques has worked effectively for this use case. Finally, we evaluated the performance of our ML models on a test dataset containing millions of instances across multiple days of traffic in a voice service.

As a next step, we can investigate the impact of hyper-parameter tuning and the use of ensemble methods. We will also investigate techniques to estimate recall with low sampling, and how to improve recall in a sparse data environment.

Estimating recall is a non-trivial exercise when the population has millions of instances. We will explore applicability of developments in Deep Learning for text analytics such as Attention model [13] and contextualized word embeddings [16].

ACKNOWLEDGEMENT

We thank Gyuri Szarvas for providing us a model based on logistic regression, which we have adapted for the work described in this paper. We also thank Sundeep Teki for his assistance in GPU benchmarking results, and Michael Brunner for identifying the sensitive content categories we described in this paper.

REFERENCES

[1] Baumgartner. Reddit. Online, https://archive.org/details/2015_reddit_comments_corpus, accessed 2019-04-05.

[2] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, Cambridge, Massachusetts, 2006.

[3] Nitesh V. Chawla, Kevin W. Boyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(2):321–357, April–June 2002.

[4] J. Cheng, M. Bernstein, C. Danescu-Niculescu-Mizil, and J. Leskovec. Anyone can become a troll: Causes of trolling behavior in online discussions. In *Computer-Supported Cooperative Work and Social Computing (CSCW)*, 2017.

[5] Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. Antisocial behavior in online discussion communities. In *AAAI International Conference on Weblogs and Social Media (ICWSM)*, 2015.

[6] Davidson. Hate speech and offensive language. Online, https://github.com/t-davidson/hate-speech-and-offensive-language/tree/master/data, accessed 2019-04-05.

[7] Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 67–73, 2018.

[8] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Massachusetts, 1998.

[9] Jigsaw Google. Toxic comment classification challenge. Online, https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data, accessed 2019-04-05.

[10] Chandra Khatri, Behnam Hedayatnia, Rahul Goel, Anushree Venkatesh, Raefer Gabriel, and Arindam Mandal. Detecting offensive content in open-domain conversations using two stage semi-supervision. *CoRR*, abs/1811.12900, 2018.

[11] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[13] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[14] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[15] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[16] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[17] Anna Schmidt and Michael Wiegand. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, 2017.

[18] William Warner and Julia Hirschberg. Detecting hate speech on the world wide web. In *Proceedings of the Workshop on Language in Social Media*, 2012.

[19] Z. Wu and M. Palmer. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138, 1994.

[20] Zhu Xiaojin. Semi-supervised learning literature survey. *Computer Sciences TR*, 1530, 2008.

[21] Harish Yenala, Ashish Jhanwar, Manoj K. Chinnakotla, and Jay Goyal. Deep learning for detecting inappropriate content in text. *International Journal of Data Science and Analytics*, 6:273–286, 2018.

VIII. APPENDIX

A. Precision and recall for one class implies the precision and recall for the other class in a binary classifier

Lemma: Suppose A and B denote the two classes for any binary classifier. If p and r denote the precision and recall, respectively, for class A and suppose g denotes the class mix ratio (i.e., the ratio of the size of class A and the size of class B) in the test set, then precision P_B and the recall R_B for class B are given as follows:

$$P_B = \frac{1}{1 + \frac{(1-r)g}{(1-rg(1/p-1))}} \text{ and } R_B = 1 - rg(1/p - 1).$$

Proof. Let $|A|$ denote the size of class A, $|B|$ denote the size of class B in the test set, and N denote the size of the test set. Within Class A, suppose the classifier predicts fraction α in class A and fraction $1-\alpha$ in class B. Within class B, suppose the classifier predicts fraction β in class A and fraction $1-\beta$ in class B. Refer to the confusion matrix shown in Figure 3 for these assumptions.

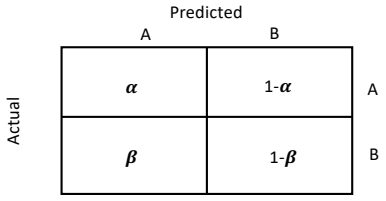

Fig. 3. Confusion Matrix

Then precision P_A for class A is given by

$$\begin{aligned} p &= \frac{\frac{Ng}{1+g}\alpha}{\frac{Ng}{1+g}\alpha + \frac{N}{1+g}\beta}, \\ &= \frac{1}{1 + \frac{\beta}{\alpha g}}, \end{aligned} \tag{1}$$

and recall R_A for class A is given by

$$\begin{aligned} r &= \frac{\frac{Ng}{1+g}\alpha}{\frac{Ng}{1+g}}, \\ &= \alpha. \end{aligned} \tag{2}$$

Similarly, precision P_B for class B is given by

$$P_B = \frac{1}{1 + (\frac{1-\alpha}{1-\beta})g}, \tag{3}$$

and recall R_B for class B is given by

$$R_B \quad = \quad 1 - \beta. \tag{4}$$

Solving Eqs. 1 and 2 for α and β in terms of p, r, and g, and then finally substituting those in Eqs. 3 and 4 gives the desired result. □

B. Sample size estimation for data quality checks

Lemma: Suppose the fraction of bad texts in a collection of texts is p, where $p > 3\%$. For any $n \geq 264$, if we draw a uniformly random (with replacement) sample of size n, then the probability of finding 3 or less errors is less than 5%.

Proof. Assume $p > 0.03$. We will find n for which the above statement is true. Since the sample follows the binomial distribution $B(n, p)$, the probability of finding 3 or less errors is

$$= \quad \sum_{k=0}^{3} \binom{n}{k} p^k (1-p)^{n-k} \tag{5}$$

$$\leq \quad \sum_{k=0}^{3} \frac{1}{k!} (np)^k (1-p)^{n-k} \tag{6}$$

$$\leq \quad \sum_{k=0}^{3} \frac{1}{k!} (np)^k e^{-p(n-k)} \tag{7}$$

$$\leq \quad \sum_{k=0}^{3} \frac{1}{k!} (0.03n)^k e^{-0.03(n-k)} \tag{8}$$

$$\leq \quad 1.1 e^{-0.03n} \left(1 + 0.03n + \frac{(0.03n)^2}{2} + \frac{(0.03n)^3}{6} \right) \tag{9}$$

$$\leq \quad 0.05 \qquad \text{for } n \geq 264. \tag{10}$$

Here, Inequality 6 uses the fact that $\frac{n!}{(n-k)!} \leq n^k$, Inequality 7 uses the fact that $(1-p)^x \leq e^{-px}$ for any $p \in [0, 1]$ and $x > 0$, Inequality 8 uses the fact that $p^k e^{-p(n-k)}$ is a decreasing function of p for $k \leq 3$ and $n \geq 103$, Inequality 9 uses the fact that $e^{0.03k} < 1.1$ for $k \leq 3$, and Inequality 10 can be shown using the computational engine Wolfram Alpha. □