NSFW AND PROFANITY FILTER

Anchit Gupta (20171041), Neel Trivedi (20171015), Kunal Vaswani (20171064)

Links
Github repository
Demo video
Project Presentation

1. INTRODUCTION

In this project we have created NSFW and profanity filters for images and text respectively. With the omni-presence of internet and social media, and the right and freedom of "free speech" on online platform, lots of unwelcoming and hurtful content is constantly being posted online either in terms of images or text. With this project we have tried to create two browser extension, which with the help of machine learning, analyzes and filters such content from the webpages in real time and tries to make the internet experience more welcoming and safe.

2. PROBLEM DEFINITION

The problem is to identify and blur profane words and NSFW images on a website using a browser extension. The main aim is to create a fast and reliable browser extension that could identify profane words and NSFW images in real-time on a website. Users will have the option to choose the threshold (profanity control) for the content filtering, 0 for no filtering, and 1 for complete filtering. Also, for now, the majority of categories considered for the NSFW images are nudity and pornographic images. The reason for this is that almost all of the publicly available NSFW image dataset contains images of these categories only. If time permits, we can later incorporate other categories like violent images as well. For text, the problem is further extended to finding fast methods for filtering profane words for a different language and hate speech for text. We have decided to use Hindi as a second language since it's the most familiar second language to us.

2.1. Major Challenges and solutions

2.1.1. Profane words

The beginning approaches on profane word detection used the idea of predicting profane words using a predefined wordlist. This approach is extremely subjective since it doesn't capture context at all. They would result in a word like suck as profane, but it completely depends on the context it is used. Moreover, the bigger question is the lack of ML used in these models in this Machine Learning era. You won't be able to detect extensions of profane words like b1tch without using ML. The above-mentioned models thus show a new series of challenges. Our model shouldn't just detect a word by itself, it should also consider the context it is used in.
Solution: Use word/phrase vectorization techniques and get better embeddings with good information about context too. With words available as vectors, we can use various ML techniques for further classification.

There are multiple existing libraries like profanity-check, profanity-filter for profane word detection too, and we need to also look at an augmented approach of incorporating them along with our task of extending profanity detection for Hindi words and thus produce better results.

2.1.2. Hate Speech

As said earlier, the problem statement could further be extended to filtering hate speech. For hate speech, the dataset should have an even distribution between hate and non-hate speech for better learning.
Solution: We could mix various datasets available for hate speech to create one proper even dataset.

We have many existing papers from SVMs, RNNs, to Transformers, which try to maximize accuracy but the issue that arises is with using complex models is the time taken by them to detect. Since we are building a browser extension, time is of the essence here.
Solution: We need to conduct various experiments to better estimate the complexity of the model versus the time taken to produce results. We would then take the model with the best results in a limited time.

2.1.3. NSFW images

One of the major problems with handling images on different websites is that these images are bound to be of different scales. Hence, the solution (NSFW image classifier) must be

able to handle images of different scales.
Solution: The first most basic solution is to use Image Retargeting methods. This method essentially resizes input images to a specific size, which the model can take as input. Another solution is to use the method employed by R-CNN style segmentation architectures. Which uses Feature Pyramidal Pooling to handle variable size input images.

Another problem that will arise if we wish to include a feature to control the degree by which the extension should blur the NSFW images. So instead of having 0-1 binary classification, users will have the option to set the strictness on the classifier: 0 being allowing all the images, 1 being completely removing(blurring) the NSFW images, and in between values will classify images as NSFW / SFW accordingly.
Solution: For this, along with the classification of image into NSFW and SFW category, we will need to train the model to regress for an NSFW score as well, which will be used to define the inappropriateness of that image at work, and using that score the model will decide whether to blur the image or not according to the scale set by the user.

3. LITERATURE SURVEY AND RELATED WORK

This section contains the description of the datasets and related works used for this project.

- **NudeNet**[1](Link)

  Authors of this idea have created Neural Nets for Nudity Classification, Detection, and selective censoring. They have used CNN based deep networks to train the classifier. They have also created a dataset of around 160,000 NSFW and SFW images across 16 classes. The classes are used for finer classification on top of the binary classification of NSWF and SFW. These classes have labels like : COVERED-BELLY, EXPOSED-BELLY, ... etc.

- **NSFW image dataset by OmerEven-Tzur**[2](Link)

  This is a publicly available NSFW image dataset that contains 63,000 NSFW and 35,500 SFW images. The images are scraped from the internet, and the dataset is provided in the url format. Since this dataset can be noisy, some filtering must be done before actually using these images.

- **Huge NSFW dataset created by Alexander Kim**[3](Link)

  Data scientist Alexander Kim has created an enormous dataset containing URLs to around 37,000 SFW images and 183,000 NSFW images over four different classes. The creators of this dataset evaluate the performance using a ResNet based CNN model.

- **My NSFW dataset by Vareza Noorliko**[4](Link)

  This is a small publicly available NSFW image dataset that contains around 800 NSFW and 800 SFW images.

- **Nsfw-Classify-Tensorflow**[5](Link)

  This is a Github repository which has an NSFW classifier model implemented in Tensorflow. It also provides a script to download NSFW dataset. The model scores an image based on five classes (Neutral, Drawing, Sexy, Porn, Hentai).

- **pytorch-sentiment-analysis** [6](Link)

  This is a Github repository which has various state-of-the-art Text models implemented in PyTorch. We utilize this for our profanity detector.

- **hate-speech-and-offensive-language**[7] (Link)

  A dataset that contains content that is racist, sexist, homophobic, and offensive in many other ways. It is based on the paper "Automated Hate Speech Detection and the Problem of Offensive Language." ICWSM.

- **Toxic Comment Classification Challenge**[8](Link)

  A large number of Wikipedia comments which have been labeled by human raters for toxic behavior. The types of toxicity are: toxic, severe_toxic, obscene, threat, insult, identity_hate.

4. NSFW CLASSIFIER

4.1. Data collection and training procedure

Unfortunately no other data apart from sexual/pornographic images were available for this task.

We have collected the data from multiple sources:

1. not-safe-for-work
2. NSFW-data-source-urls
3. NSFW-data-scraper

The dataset had to be cleaned after downloading, for example:

1. Delete duplicate images
2. Delete corrupt images
3. Delete images with 0 size

We tried two approaches to train the NSFW image classifier:

1. **Transfer Learning**

   In this we have trained **4 different models**, using these as the base model and adding fully connected layers on of it to classify the output into two categories - NSFW and SFW

Model	Accuracy	Inference Time per image
MobileNetV2	**96.35**	**52 ms**
ResNet50	91.28	82 ms
ResNeXt101	92	180 ms
VGG	90.78	106 ms
Ours from scratch	71.69	30 ms

Table 1. Results of different models for NSFW image classification

(a) ResNet50[9]

(b) ResNeXt101[10]

(c) VGG16[11]

(d) MobileNetV2 [12]

2. **Training from Scratch**

   Our model consists of 5 CNN layers and 3 Fully connected layers.

Some other training details:

- **Optimizer**: Adam
- **Batch Size**: 128
- **Learning Rate**: 0.0001
- **Number of Epochs**: 50
- Hardware details of CPU: Processor: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.81 GHz Installed RAM: 8.00 GB (7.89 GB usable) System type: 64-bit operating system, x64-based processor

4.2. Results

Based on the accuracy and inference time taken in Table 1 we decided to use MobileNetV2 as our pretrained model.

5. PROFANITY DETECTOR

5.1. Data collection and training procedure

We used a combined dataset from these two sources:

1. hate-speech-and-offensive-language
2. Toxic Comment Classification Challenge

We kept classes with any kind of toxicity or offensive language as profane for either of these datasets. Our combined dataset has around 140K non-profane sentences and 37K profane sentences. Following are the approaches used to train the profanity detector:

1. **SVM**

   Since the first part of our solution involved making a fast model, we incorporated the simplest strategies for each part of the problem. We used the popular CountVectorizer class which helps in vectorization of each sentence. For fast learning, we used the simplest of models: Linear SVM. scikit-learn's LinearSVC class is used for that. The main reason to use Linear SVM is: it's fast enough to run in real-time yet robust enough to handle many different kinds of profanity.

2. **LSTMs**[13]

   The major issue from the previous approach is the vectorization of sentences. It doesn't incorporate various contextual information from sentences. To incorporate sequential information we use the lstm model. We resist ourselves from using simple RNNs since they suffer from vanishing gradient problems. We also exploit further tricks by making our lstm model bidirectional and multi layer.We use pre trained 100 dimension glove embeddings trained on 6 billion tokens.

3. **BERT**[14]

   Misspellings are one of the most important issues which lstm models that rely on a vocabulary face. Words like "f**k", "biiiiiiitcchh" are difficult to approach without using subword information. BERT exploits subword information using WordPiece. WordPiece is a subword segmentation algorithm used in natural language processing. The vocabulary is initialized with individual characters in the language, then the most frequent combinations of symbols in the vocabulary are iteratively added to the vocabulary. This subword based approach helps a lot in case of out of vocabulary words.

5.1.1. Training Details for LSTMs

- **Optimizer**: Adam
- **Batch Size**: 128
- **Hidden dimension**: 256
- **Number of Layers**: 2
- **Learning Rate**: 1e-3
- **Dropout**: 0.5
- **Number of Epochs**: 6

5.1.2. Training Details for BERT

- **Optimizer**: Adam
- **Batch Size**: 128

Model	Accuracy	Inference Time per sentence
BERT	**96.34**	**147 ms**
LSTMs	96.17	35 ms
SVM	95.28	1 ms

Table 2. Results of different models for profanity detector

- **Learning Rate**: 1e-3
- **Dropout**: 0.25
- **Number of Epochs**: 6

Hardware details are same as that mentioned in NSFW classifer.

5.2. Results

The LSTM models improve upon SVM models in detecting context. We can see by the following example detection by the LSTM model:

1. Go suck this idiot: Profane
2. Many aphides, puncture the leaves, suck out the saps: Non-Profane

BERT models due to the subword segmentation techniques can classify misspellings like f*ck, fuckkkk, bitchhhh as well. Based on the accuracy and inference time taken in Table 2 we decided to use LSTMs as our pretrained model. We also propose users to use BERT in case of serious checks, where detection of profanity is crucial.

6. BROWSER EXTENSION

One of the requirements of the project was to create a browser extension which will allow users to filter NSFW and profane content from webpages in real time. For this purpose we created two Chrome extensions:

1. **NSFW filter** : This extension filters NSFW images and blurs them on webpages.
2. **Profanity filter**: This extension filters profane text from webpages.

The architectural diagram of the extensions is shown in Figure 1.

As shown in the Figure 1, the browser extensions take the images and text from the webpage and using the backend pretrained ML models decides whether they are NSFW or profane respectively and based on these probabilities, the webpage is updated with blurred content.

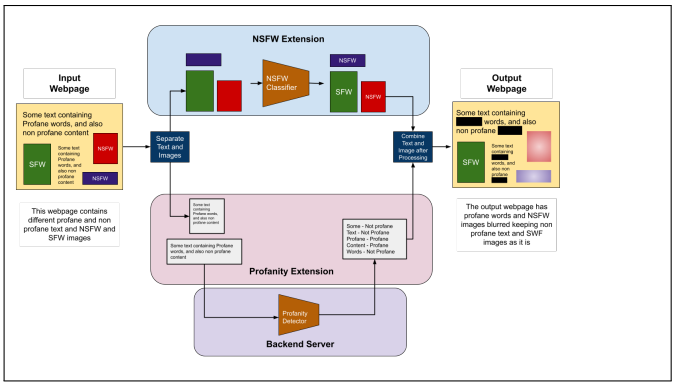

Fig. 1. Architectural diagram of the browser extensions.

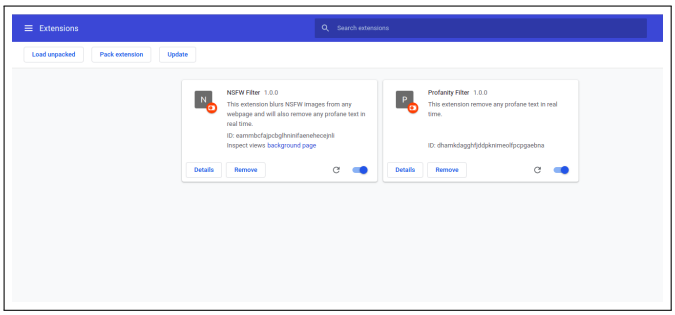

Fig. 2. Extensions added to chrome browser

The procedure to include these extensions into your browser are given on our github repo, here. Figure 2 shows the extensions added to the chrome browser.

6.1. Results of using the extensions

Once the extensions are added to the browser, they will filter out content from all the webpages. Figure 3 shows a sample webpage which containg NSFW images and profane text. Once the extensions are enabled in the browser, such content is blurred as can be seen in Figure 4. We can see that SFW images are as it is while the NSFW images are blurred. Similarly normal text is kept intact whereas profane text like ”fuck you” or ”you cunt” are blurred by the extension. Further upon searching for ”hentai images” or ”sexy images” on google will also give blurred results as shown in Figure 5

7. CONCLUSION

As can be seen by the results, we have successfully created the required deliverable of the browser extensions to filter NSFW and profane content from the webpages. There is still room for improvement in these filed of work. The profanity detector can be extended to include as many languages as possible.

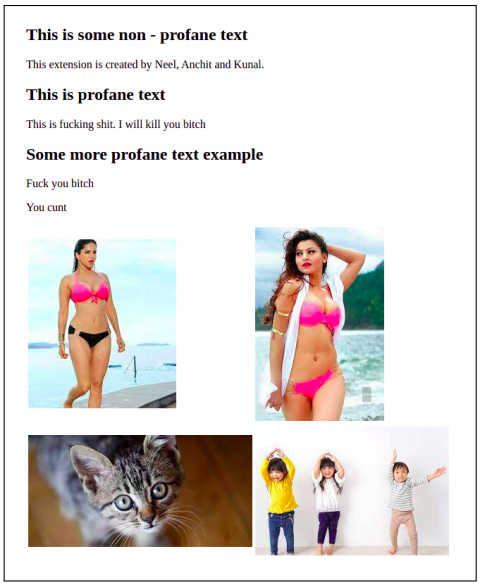

Fig. 3. Sample webpage without using the extensions

Fig. 4. Sample webpage without using the extensions

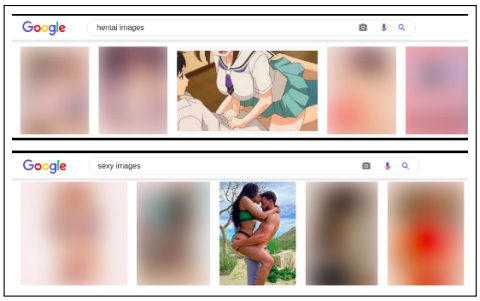

Fig. 5. Blurred NSFW images on google search

Also right now the profanity detector is blurring the profane text, as a future improvement we could implement a system which would replace the profane part of the sentence with non-profane text keeping the meaning of the statement more or less the same.

8. REFERENCES

[1] notAI tech, "notai-tech/nudenet," . 2

[2] OmerEven-Tzur, "Nsfw - not-safe-for-work," Apr 2020. 2

[3] alex000kim, "alex000kim/nsfw-data-scraper," . 2

[4] Vareza Noorliko, "my nsfw dataset," Jul 2019. 2

[5] MaybeShewill-CV, "Maybeshewill-cv/nsfw-classification-tensorflow," . 2

[6] Bentrevett, "bentrevett/pytorch-sentiment-analysis," . 2

[7] T-Davidson, "t-davidson/hate-speech-and-offensive-language," . 2

[8] "Toxic comment classification challenge," . 2

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," 2015. 3

[10] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He, "Aggregated residual transformations for deep neural networks," *arXiv preprint arXiv:1611.05431*, 2016. 3

[11] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. 3

[12] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2019. 3

[13] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 3

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. 3