Can We Achieve More With Less?
Exploring Data Augmentation with Toxic Comment Classification
CS221 Project Progress Report

Chetanya Rastogi, Fang-I Hsiao, Nikka Mofid
{chetanya, fihsiao nmofid}@stanford.edu

1 Introduction

Data is the bottleneck of machine learning. There are millions of interesting problems in the world but, without enough data, high accuracy classifiers to solve these problems can not be built. In the converse, for problems where data is abundant labeling can take days. Thus, in this project, we aim to tackle this data divide and explore if high accuracy classifiers can be built from small datasets using a combination of data augmentation techniques and machine learning methods. We are working to develop a model to detect and classify toxic speech in comments to help web moderators fight back against online harassment and cyberbullying, and also protect data annotators from the psychological stress of having to label an extremely large, graphic dataset.

2 Model

In this project, we aim to explore and compare the performance of data augmentation techniques in tandem with different machine learning methodologies. To solve this problem, we will be using the "Wikipedia Toxic Comments" dataset [9]. The data contains a list of ∼158k Wikipedia comments and six binary labels for the kind of hate speech each comment qualifies as. For our task, we take any kind of toxicity as a positive class and the rest of the comments as the negative class. Furthermore, we split the data into train and test and keep the test set aside which will only be used for evaluation purposes.

To evaluate the performance of data augmentation, we further sample 5% of the data from the train set which serves as the small training set for our baseline(without data augmentation) and the entire train set serves as the oracle. We evaluate the performance of our augmented datasets against our learning algorithm: logistic regression, SVM, and a bidirectional LSTM. For this project, we have chosen to focus on Easy Data Augmentation and BackTranslation as our primary augmentation algorithms as they are two of the most popular and effective data augmentation methods [10]. We use the following metrics to evaluate our results: Recall and F1 score. This is because the data is highly imbalanced (only 10% of the comments are toxic) making the Recall and F1 score the most pertinent metrics for this problem.

3 Learning Algorithms

We will be using three learning algorithms of varying complexity as our classifiers. We will compare the effect of our data augmentation algorithms on these learning methods.

3.1 Logistic Regression (LR)

Logistic regression is a simple learning algorithm used for classification that assigns observations to a discrete set of classes. It is different than linear regression in that it uses the logistic sigmoid function to return probability value which can be mapped to two or more discrete classes [5]. We chose to use logistic regression as one of the learning algorithms to analyze the performance of data augmentation because it is simple and very commonly used for classification tasks like this one so the performance results are very relevant to those undertaking similar projects with small or limited datasets.

3.2 Support Vector Machines (SVM)

SVM is a discriminative classifier which given labeled data, finds an optimal hyperplane in an N-dimensional space that uniquely categorizes the data points. There are many hyperplanes or "decision boundaries"

that can be chosen when splitting the data into classes and an SVM works by finding the plane that has the maximum margin or distance between data points of both classes. The points that are closest to the hyperplane are support vectors and they are used by the SVM to maximize the margin of the classifier [1]. We chose to use an SVM as one of the learning algorithms to analyze the performance of data augmentation because it is more expressive than logistic regression and highly interpretable in contrast to a neural network. Due to its wide use for such problems and slight increase in complexity, we thought that an SVM would provide an excellent performance comparison to logistic regression and, as we will see in the next section, the more complex bidirectional LSTM.

3.3 Bidirectional LSTM (Bi-LSTM)

LSTM is a common deep learning model for natural language processing tasks. It is a recurrent neural network with a more complicated architecture. We use bidirectional LSTM which reads the text forward and backward and then combines these two features to make the prediction [6]. We chose to use bidirectional LSTM as one of our learning algorithms to analyze the performance of data augmentation because unlike SVM and Logistic Regression (LR), LSTMs can capture the sequence of tokens which adds an extra layer of information not captured by the bag-of-words model of SVM and LR. Thus, it allows us to see the performance of data augmentation on a more complicated and precise learning algorithm which may be applied to such a classification task.

4 Data Augmentation Algorithms

We plan to implement different data augmentation techniques and compare their effect in terms of the performance boost on different types of learning methods(described in section 2).We chose to focus on Easy Data Augmentation and Backtranslation because they are two of the most popular and effective methods of data augmentation [10].

4.1 Easy Data Augmentation(EDA)

4.1.1 Algorithm [10]

EDA makes use of four extremely simple operations to generate new augmented sentences given a basis sentence. The algorithm takes in one sentence from the training set and performs one of the following operations chosen at random:

1. **Synonym Replacement(SR):** Randomly choose n tokens from the sentences that are not stop words. Each chosen token is then replaced by one of its synonym chosen at random.

2. **Random Swap(RS):** Randomly chose n pairs of tokens from the sentence and swap their positions.

3. **Random Insertion(RI):** This step is a slight advancement of SR. Like SR, again choose n tokens at random from the sentence which are not stop words but instead of replacing them with their respective synonyms, insert the synonym (one for each token) at a random position in the sentence.

4. **Random Deletion(RD):** For each token in the sentence, determine whether to delete it or not with a probability p.

Each of the above four operations has an intuition behind it that works on tackling different problems that are mostly encountered while working on a small dataset. SR helps in maintaining the same syntactic and semantic meaning [11] as the original sentence while providing an opportunity to the model to "learn" about new words that might not be present in the original dataset. RI and RS maintain all the original tokens in the sentence but introduce perturbations that helps in regularization and as a result, the model generalizes well when it comes across unknown patterns. RD helps in reducing model overfitting that is

generally the case with a small dataset. By deleting words at random, it makes sure that the model is not "memorizing" particular patterns that it sees in the samll dataset and forces it to explore other features.

4.1.2 Implementation Details

The aim of any data augmentation technique is to produce new data points while preserving the original class label. We implement EDA in such a way that the amount of augmentation a sentence undergoes(number of tokens inserted, swapped, deleted, or replaced) is proportional to the length of the sentence. This is because long sentences are more robust to noise than short sentences because it's very easy to destroy a structure of a sentence having fewer tokens. A parameter (α) dictates how many tokens(n) will undergo RS,SR, RI(n=αl, l being the length of the sentence) and serves as a hyperparameter that needs to be tuned. For the sake of consistency, the probability with which a token is deleted(RD) is also set to α.

We define four functions, one corresponding to each operation, which are called uniformly at random to generate an augmented instance of a given instance. We also provide the flexibility of how many augmented sentences does a user want to generate per sentence. For SR, we make use of python's *nltk* library for defining stop words and finding synonyms while the rest of the functions are fairly straightforward and doesn't require any external library support. Examples of augmented sentences are shown in Table 1

Original Sentence	**Operation**	**Augmented Sentence**
how can you **block** me when you're just an editor	SR	how can you **impede** me when youre just an editor
how can you **block** me when you're just an editor	RD	how can you me when youre just an editor
how **can** you block me **when** you're just an editor	RS	how **when** you block me **can** youre just an editor
how can you block me when you're **just** an editor	RI	how can you block **simply** me when youre just an editor

Table 1: Augmented Sentences using EDA Operations

4.2 BackTranslation

4.2.1 Algorithm

The idea of backtranslation was introduced to generate parallel sentences in two languages for improving the performance of Neural Machine Translation models [8]. The key idea is to use the noise introduced by NMT models to augment training data. Since NMT models try to retain the same semantic meaning as the source language, the generated output is a rephrased version of the same sentence, thus providing new data points for the models to train on.

Backtranslation works by taking a sentence from the training set(source) and translating it to an intermediate language and then translating the generated output back to the source language using an NMT model. This process results in the introduction of noise at both the translation stages and we end up with a new sentence having similar semantic meaning as the original one. The pipeline for backtranslation is as follows:

$$\text{Original Sentence} \xrightarrow[\text{another language}]{\text{translate to}} \text{Intermediate Sentence} \xrightarrow[\text{original language}]{\text{backtranslate to}} \text{Augmented Sentence}$$

4.2.2 Implementation Details

We make use of the paid Google Translate API [2] for this task. A script was written that links with the user account on Google Cloud Platform and makes HTTPS requests to the API. The API supports more than 50 languages. We chose Spanish as our intermediary language as it is the second most spoken language in the world [3] and hence has a high probability that NMT models trained on English-Spanish would have higher performance due to availability of high-quality data. An example of such an augmented sentence is shown below in Table 2

Original Sentence	**Operation**	**Augmented Sentence**
I too agree with your suggestion thanks for taking this on	BT	I also agree with your suggestion thank you for taking this

Table 2: Augmented Sentences using Backtranslation (BT)

5 Preliminary Results and Analysis

As mentioned in Section 2, because of the highly imbalanced nature of our data set (only 10 percent of the comments are toxic) we use Recall and F1 score to evaluate our dataset. Table 3 and 4 summarise our preliminary results. Clearly both the data augmentation techniques show improvement over the baseline with an average F1 score improvement of 3% on EDA and 2.3% on backtranslation over all the 3 methods. The same is true for Recall as the EDA shows an average improvement of 6.3% while backtranslation gives an average boost of 6%, showing that data augmentation can improve the performance of classifiers.

To understand these improvements, we plot the graphs of feature importances (Refer Appendix B) for all three classifiers. As evident from the figures, both EDA and backtranslation help in boosting the importance of the top features to the level obtained in the oracle(full dataset). Also simple classifiers such as LR and SVM receive a major boost from both the augmentation techniques whereas Bi-LSTM shows a greater improvement in performance with backtranslation than with EDA. This can be attributed to the fact that since EDA is more suitable for bag-of-words model as it treats every token independently and disregards the semantic structure whereas backtranslation retains the semantic structure and hence provides better augmentation for Bi-LSTM model.

	LR	SVM	Bi-LSTM
Baseline	0.677	0.724	0.764
EDA	0.736	0.745	0.772
Backtranslation	0.708	0.744	0.784
Oracle	0.801	0.814	0.829

Table 3: F1 Score Comparison

	LR	SVM	Bi-LSTM
Baseline	0.527	0.598	0.686
EDA	0.629	0.662	0.707
Backtranslation	0.576	0.641	0.767
Oracle	0.724	0.739	0.795

Table 4: Recall Score Comparison

6 Next Steps

We further plan to study and analyse our results to provide an analytical backing to our intuition. Moreover we plan to experiment by combining both the techniques and compare it with a null hypothesis where we just make multiple copies of the training data without any augmentation. We also plan to implement Backtranslation in another language in order to get a better idea of its effects and see whether the choice of language effects the algorithms performance. In addition, if time permits, we would like to explore if we could extend the paradigm of Positive-Unlabeled Learning [4] to utilise unlabeled data for augmentation.

References

[1] *CH 2: SVM Theory*. https://bit.ly/2KsERg2.

[2] *Google Translate Setup: Quickstart Guide*. https://cloud.google.com/translate/docs/basic/setup-basic.

[3] *List of languages by number of native speakers*. https://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers.

[4] Bing Liu et al. "Partially Supervised Classification of Text Documents". In: *Proceedings of the Nineteenth International Conference on Machine Learning*. ICML '02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 387–394. ISBN: 1-55860-873-7. URL: http://dl.acm.org/citation.cfm?id=645531.656022.

[5] *Logisitic Regression: ML Cheatsheet*. https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html.

[6] *LSTM*. https://skymind.ai/wiki/lstm.

[7] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation". In: *In EMNLP*. 2014.

[8] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Improving Neural Machine Translation Models with Monolingual Data". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 86–96. DOI: 10.18653/v1/P16-1009. URL: https://www.aclweb.org/anthology/P16-1009.

[9] *Toxic Comment Classification Challenge*. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/discussion.

[10] Jason W. Wei and Kai Zou. "EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks". In: *CoRR* abs/1901.11196 (2019). arXiv: 1901.11196. URL: http://arxiv.org/abs/1901.11196.

[11] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-level Convolutional Networks for Text Classification". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 649–657. URL: http://dl.acm.org/citation.cfm?id=2969239.2969312.

Appendices

A Experimental Setup

Figure 1 shows the structure of our classification pipeline. The raw text is first preprocessed to remove special characters such as @, $, ?, etc and is converted to lower case to reduce the vocabulary size. Since many comments try to hide the toxicity by putting special characters or by changing the spelling we also normalize the text using certain regular expressions. After the preprocessing is done, we run the data augmentation scripts to increase the training set size.

For EDA, we experiment with multiple values of α and found that at a value of 0.1-0.15 we get optimal results. As described in section 4.1.2, we make use of *nltk* library for finding stopwords and synonyms. We generate 9 augmented sentences per training sentence thus, increasing the size of the training set by 10x.

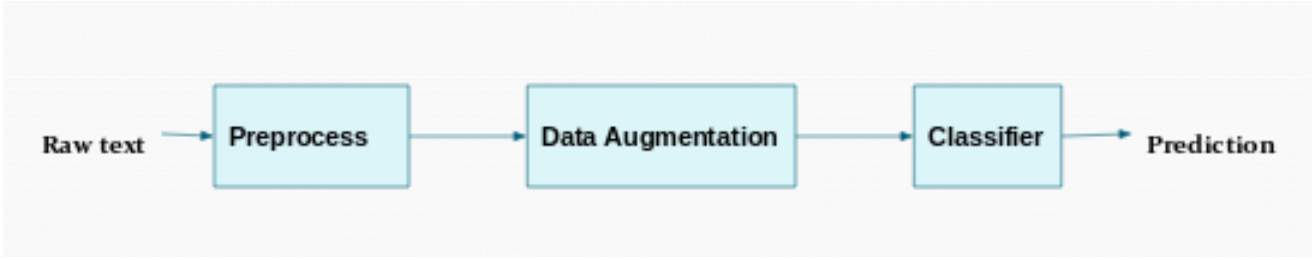
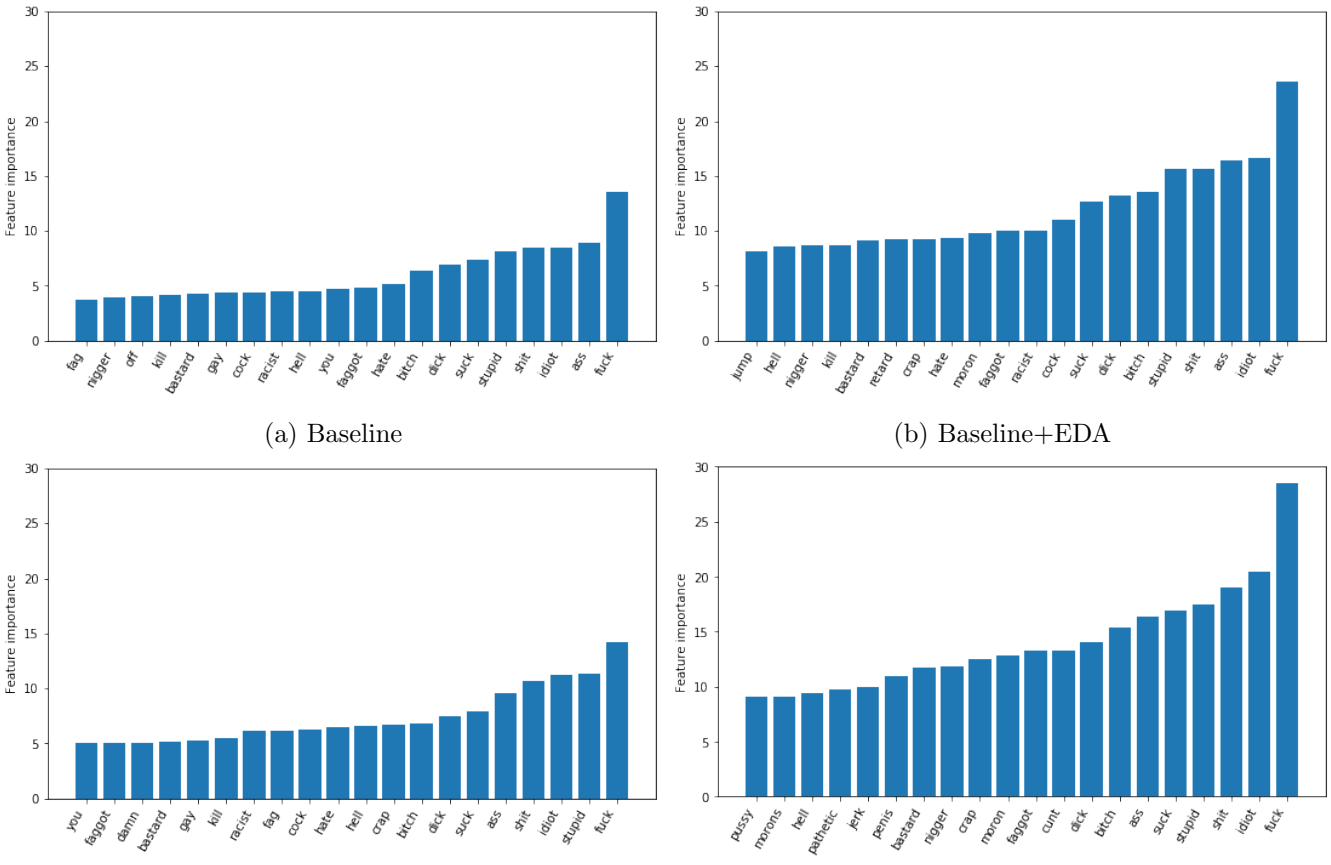


Figure 1: Toxicity classification pipeline

For backtranslation, we use Spanish as our intermediary language and generate one augmenetd sentence per sentence in the training set thus, increasing the training set by 2x.

For the classification models, *scikit-learn* is used to train LR and SVM models while the Bi-LSTM model is trained using *keras* with tensorflow backend. LR and SVM make use of TF-IDF feature vectors while the Bi-LSTM model uses Glove [7] embedding layer.

B Plots and Figures



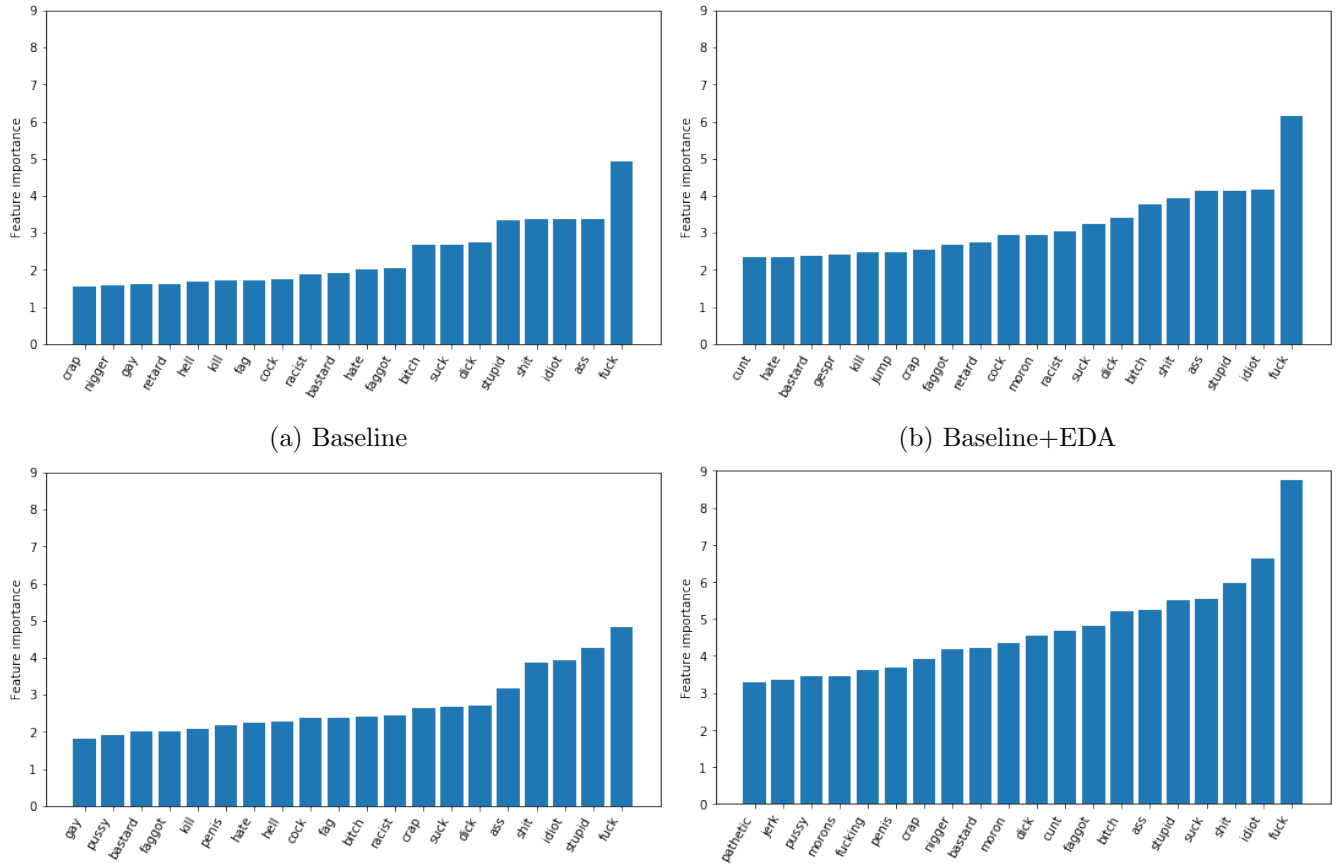
(a) Baseline

(b) Baseline+EDA

(c) Baseline+Backtranslation

(d) Oracle

Figure 2: Feature importance for Logistic Regression



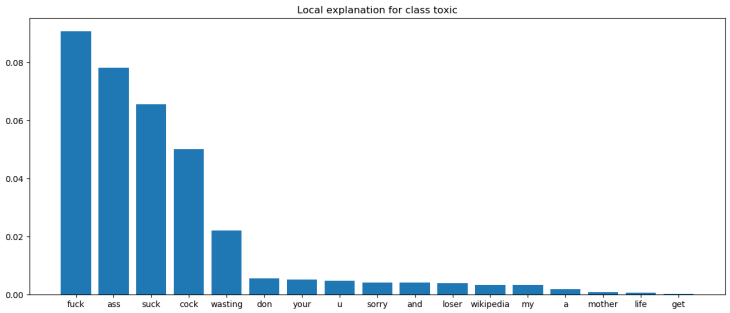
(a) Baseline

(b) Baseline+EDA

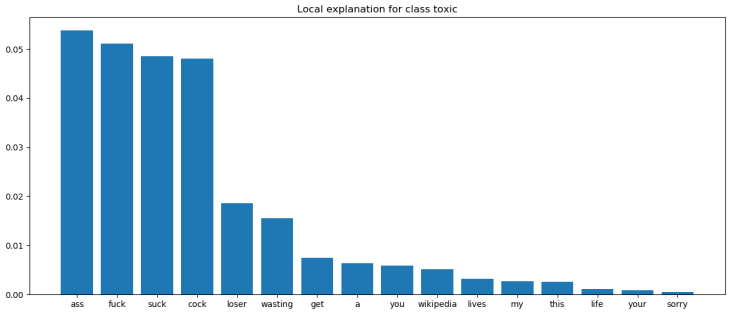
(c) Baseline+Backtranslation

(d) Oracle

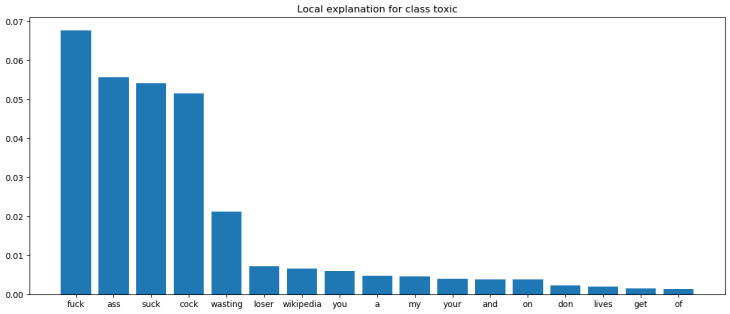
Figure 3: Feature importance for SVM



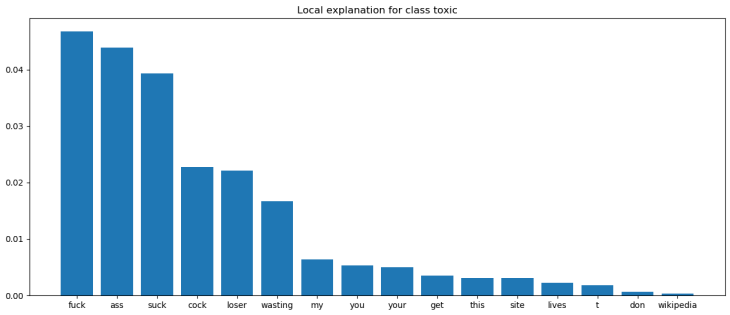
(a) Baseline



(b) Baseline+EDA



(c) Baseline+Backtranslation



(d) Oracle

Figure 4: Feature importance for Bi-LSTM