1 Baseline

There are several approaches how we can introduce the baseline model. However, I have decided to stick with simpler solution. Idea is to build dictionary based solution which will replace a swear word (or multiple related, forming swear formulation) with safer non-toxic synonym.

1.1 Building Dataset

While there are a lot of “bad words” datasets available online, it is not so trivial to find their safe synonyms. Multiple approaches can be used to find alternatives:

1. **Machine Learning:** use pretrained model to find safe synonym for each word of the dataset
2. **Word Corpus:** use corpus of english words to find synonyms. For instance, WordNet has this feature. However, we have to consider the problem, that WordNet does not itself classify the toxicity of the word, so we have to somehow deal with toxic synonyms.
3. **Manual:** manually derive synonyms. This can work for smaller versions of the datasets.

To address the issues with synonyms we can consider generating several (preferably long enough list) of synonyms, regardless the method we chose. Then, we need an assumption, that there is little to none swear words which are not included in our dataset. Evidently, this assumption is wrong, but approximation is good enough for building reliable dataset. Finally, we can filter the synonyms, omitting the ones which belong to our dataset of unwanted (i.e. swear) words.

This way datamining becomes very easy to follow pipeline:

1. Choose swear words dataset
2. Generate a list of synonyms for each of them, using algorithm of choice
3. Select an appropriate synonym, which is not in swear dataset itself
4. Convert the dataset into the replacement dictionary for the ease of use

1.2 Algorithm

Given the replacement dictionary, we can effectively detect forbidden words in the text and replace them with safer alternatives.

However, we still need to consider several issues.

1. Some sentences may stay unchanged. E.g. swear word is not in our list or highly obfuscated. We **should not** consider these sentences in evaluation. The reason for this, unchanged sentences most probably will have ideal scores on most of the metrics of semantic and norm similarity. This will lead to overalll result better, than it really is, if considered separately from toxicity.
2. Carefully preprocess the sentence. Ideally we want the model to effectively work on raw data

(or with some internal preprocessing). This is an issue if we consider the obfuscated words and words with different capitalization.

3. The performance of this algorithm is highly dependent on the dataset generated, as a result, toxic words list as well as synonyms should be extensive enough to cover most of the dataset.

2 Propositions

Before proposing the models we can use for the task, we need to define the task itself. While text detoxification is a general goal, we can approach it with different ways.

2.1 Possible approaches

1. **Inplace replacements:** just find a bad word and replace with predefined synonym.

   Not a ML algorithm at all, just a simple inplace algorithm used in our baseline model.

2. **Masking:** we can use encoder, trained to predict masked tokens. Masks can be placed instead of synonyms in previous approach.

   For example, we can use BERT for predicting several replacements for masked token and select the most appropriate one, based on predefined metrics. For instance, this approach was called CondBERT. This approach is better in a sense, that it can preseve the meaning by reading the whole context, not only the toxic part.

3. **Paraphrasing:** seq2seq transformation of the given text.

   While most probably the model of choice will need fine-tuning, which quite heavy operation on well performant models, it is still possible. Variety of models is incredible: LLaMA, GPT2, GPT3, T5 and others.

4. **Others:** multiple other approaches.

   There are approaches combining discriminator with generator, to teach the model preserve the style (e.g. non-toxic), combining discriminator with paraphraser (to ensure preservation of the meaning), using seq2seq supervised transformer as translator, since they are limited in offensive instruments and many other.

2.2 Model Criteria

With such wide variety of instruments it becomes increasingly difficult to select the proper tooling. We need more constraints on our solution, to optimize it, since simply trying to reach the best performant models will lead to infinite tuning of SOTA techniques. Moreover, it will require enourmous hardware (of financial for cloud-based solutions) investments. Therefore, we need to define several criteria for the selection of approach.

1. Solution from scratch will require a lot of computational power for training, while lacking even a fraction of power of existing transformers. Consequently, we shall use pre-trained models.

2. The approach of choice should either be incredibly pretrained, requiring little to none fine-tuning, or good enough to capture specifics of the task from a small fraction of the dataset. This balance of pretrained perfomance and fine-tuning amount should be as optimized, as possible.

3. While the solutions computational demands are very important, we should not forget about the performance of the final model. It should produce robust predictions in short time.

Each approach has own pros and cons in terms of derived criteria, so a proper research of tooling is required.

3 Metrics

The choice of metrics can greatly affect the outcomes of the research. Here we try to introduce the list with some of the metrics we might consider to use in this research.

3.1 Semantic Similarity

We can use word embeddings to measure the semantic similarity of the result with the original text. Embeddings can be either trained by ourselves, or be pre-trained (e.g. GloVe, etc). Vectorizing the results and taking the cosine similarity can give a significant insight into the similarity of the given sentences.

This metric might produce poor results if the quality of embeddings is not satisfactory. This can be especially an issue in case of baseline model. Simple replacement of words can break the original sense of the sentence with unsuitable synonym. This in turn will lead to incorrect feature capture.

3.2 BLEU

BLEU — Bilingual Evaluation Understudy.

The idea of the metric is to capture matching n-grams in the predicted sequences. It is very widespread metric for natural language processing. Especially in translation tasks. We can and probably should consider the task of detoxification as translation from non-toxic language to a toxic one.

3.3 METEOR

METEOR — Metric for Evaluation of Translation with Explicit Ordering.

This metric is less widespread. However, it addresses many issues of BLEU, resulting in much more balanced view of translation quality. The metric captures semantic equivalence better, then BLEU, by considering synonyms, stemming, being more flexible.

However, advantages come at cost of more computationally intensive algorithm, which might become an issue on large chunks of data.

3.4 Discriminator Network

Given the large enough dataset of reference sentences and ideal paraphrasings, we can train dicriminator. The main purpose of discriminator is to detect whether the text was written by human or machine. But with enough finetuning on dataset references, we can theoretically bias it towards non-toxic sentences.

Chances are, if the discriminator struggles to claim that the text is machine generated, then the paraphrasing quality is relatively good.

4 Conclusion

After definition of possible approaches, all what is left is to pick one or several and compare them. We have models, metrics and the plan for obtaining the dataset.

We have to solve the issue with computational power, either using colab or kaggle.

Finally, we have to seemlessly integrate all the parts of the work locally, to make a uniform architecture capable of solving assignment. But for importantly, representing final product ready for deployment and presentation.