**Detoxify scoring – Negatives**

**Issues with training data: Training data diversity**

Detoxify was trained on Wikipedia discussions and reddit posts/comments and not twitter. Therefore, it may not be as accurate when classifying tweets due to differences in online “culture”. For instance, detoxify was outperformed by the BERT base model when tested on a video game chat toxicity dataset, despite both being fine-tuned to the dataset1.Furthermore, twitter users consistently use more informal language and swear words relative to reddit2. One solution could be to manually classify a couple hundred twitter examples and fine-tune detoxify on them, then use the fine-tuned model rather than the base model for inference.

**Issues with training data: bias and subjectivity**

Additionally, there is the possibility of models being trained on biased training data; If a model is trained on biased training data, such as training data where AAE tweets are more likely to be labelled as toxic relative to non-AAE tweets, this would result in the model being more likely to label AAE tweets as toxic3.

The jigsaw dataset, the dataset used to train the detoxify model, has been widely criticized for using the average score assigned to each text by all reviewers to label texts. Depending on how one sets the threshold, this could mean that if 24 out of 50 people marked a text as being inappropriate, it would still be marked as appropriate. Conversely, lowering the threshold would result in a text marked as appropriate by 40/50 reviewers being marked as inappropriate4. The question remains as to what threshold to use.

**Issues with bias against reclaimed language**

Reclaimed language - language once used disparagingly against POC and queer communities that has been “reclaimed” and used in everyday vocabulary by marginalized communities – is consistently marked as hate speech by hate speech detection models5. This is a particularly important issue with detoxify, which tends to flag sentences with slurs as toxic regardless of the context, and twitter, which has a sizable presence of AAE (African-American English) speakers.

**Detoxify scoring – Positives**

Despite these issues, Detoxify is still miles ahead in terms of accurately classifying hate speech compared to models produced prior to Google’s toxic comment classification challenge. The model is, as far as I’m aware, the only publicly available toxicity detection model stemming from the toxic comment classification challenge. Therefore, while not perfect, it is still a large improvement over previously available models such as perspective API. Furthermore, the model is built using pytorch and transformers, two somewhat intuitive libraries used for machine learning. The Transformers library in particular streamlines LLM classification tasks by using a pipeline, an object that turns complex code into a simple API.

**Stance Detection**

Stance is defined as “a public act by a social actor, achieved dialogically through overt communicative means, of simultaneously evaluating objects, positioning subjects, and aligning with other subjects, with respect to any salient dimension of the sociocultural field’’6. In other words, *stance* is the value judgement that a stance taker holds towards a given proposition. Stance differs from sentiment analysis in that sentiment analysis analyzes a broad emotion in a text, while stance is a position held against an individual or an abstract target, irrespective of emotion. For instance, consider the following sentence:

*“Abortion is evil, a crime against god-- baby-killers, see you in hell!”*

The sentiment here is clearly negative, as indicated by the words “evil”, “crime”, etc. The stance is *anti-abortion.* Consider the second example:

*“A woman’s ability to give birth is a true miracle, and one not to be taken for granted. It is God’s will for one to procreate, even if not on purpose. You are always ready for a baby.”*

The stance here remains anti-abortion, however this is harder to determine. The sentiment, however, is positive.

*Stance detection* is the task of predicting a writer’s stance with respect to a given subject, within his writings6. For us, this would take multiple forms: either pro-vaccine or anti-vaccine, pro-lockdowns or anti-lockdows, etc. Early forms of stance detection came about in the form of mathematical models. Today, stance detection is primarily done through the use of large language models (LLMs). This typically takes the form of one or two methods:

1. **Fine-Tuning:** The first method involves *fine-tuning* a LLM to perform a particular task7 8 . In our case, this would be stance detection. Fine-tuning involves a series of steps: First, data would need to be manually annotated to reflect the stance of each example (usually, a couple hundred to 1000 entries is enough). The data would then be split into training and validation sets, and a LLM would be trained and tested repeatedly on both sets. The result would be a LLM fine-tuned for the specific task of stance detection. We would then run inference on the new model by having it generate outputs for the remainder of the dataset. Some papers have attempted to finetune a instance of LLaMa for stance detection. I highly advise against this due to way LLaMa reads information (left to right instead of bi-directionally) making it computationally heavy and inaccurate for sentiment analysis and adjacent tasks. Instead, I recommend we use BERT.

**Pros:**

* I’ve done this before and know how to do it.
* More effective in the long run, since all one needs to do to run the model in the future is call it.
* Gives us accuracy metrics for how well our model performs.

**Cons:**

* Computationally heavy: Fine-tuning required access to an NVIDIA GPU and at least 16 gigabytes of VRAM. Fortunately, I have google collab premium, which should solve this issue, however I cannot guarantee that I will be able to work with datasets as big as what we have on there.
* Requires manually annotating hundreds of tweets, attaching stance values to them.

1. **Prompt engineering:** The second method involves feeding a LLM highly specific prompts tailored to stance detection, called prompt engineering9. This takes the form of providing the LLM with a template and a set of instructions, then feeding it a dataset which it would analyze.For instance, if we wanted to analyze the stance of users towards Kamala Harris’ electoral campaign, our template would be:

<stance> is the stance for the target Kamala Harris

And our instructions would be:

Assume you are an expert social scientist. Analyze the following dataset of tweets to determine stance towards Kamala Harris’ campaign. The stance is one of “against”, “for”, or “neutral”.

**Pros:**

* Less time-consuming than fine-tuning a model.
* **\***Does not require manually annotating a dataset.

**Cons:**

* I’m not familiar with this method and have never tried it.
* \*We have no way of knowing how well our model is performing on the dataset without creating a validation set. For all we know, it could be answering completely wrong: the only way of knowing would be to create a validation set and run our model on it. But at that point, why not just fine-tune?

1. **(This is a grad student thesis, so won’t be able to use in an actual paper)**

Matteo, D. (n.d.). Benchmarking AI Techniques for Toxicity Detection in Online Games. <https://studenttheses.uu.nl/bitstream/handle/20.500.12932/44015/Thesis_MatteoDiana_8255083.pdf?sequence=1>

1. Arazzi, M., Nicolazzo, S., Nocera, A., & Zippo, M. (2023). The importance of the language for the evolution of online communities: An analysis based on Twitter and Reddit. Expert Systems With Applications, 222, 119847. <https://doi.org/10.1016/j.eswa.2023.119847>
2. Sap, M., Card, D., Gabriel, S., Choi, Y., & Smith, N. A. (2019a). The risk of racial bias in hate speech detection. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. https://doi.org/10.18653/v1/p19-1163
3. Adrian-Dip. (n.d.). Detoxify: Python text classifiers that filter inappropriate or offensive text. GitHub. <https://github.com/adrian-dip/detoxify>
4. Welbl, J., Glaese, A., Uesato, J., Dathathri, S., Mellor, J., Hendricks, L. A., Anderson, K., Kohli, P., Coppin, B., & Huang, P. (2021). Challenges in Detoxifying Language Models. Findings of the Association for Computational Linguistics: EMNLP 2021. <https://doi.org/10.18653/v1/2021.findings-emnlp.210>
5. Alturayeif, N., Luqman, H., & Ahmed, M. (2023). A systematic review of machine learning techniques for stance detection and its applications. *Neural Computing & Applications*, *35*(7), 5113–5144. <https://doi.org/10.1007/s00521-023-08285-7>
6. Cruickshank, I. J., & Ng, L. H. X. (2023, September 24). *Prompting and Fine-Tuning Open-Sourced large language models for stance classification*. arXiv.org. <https://arxiv.org/abs/2309.13734>
7. Chuang, Y.-S. (n.d.). *Tutorials on stance detection using pre-trained language models: fine-tuning BERT and prompting large language models*. University of Wisconsin–Madison. <https://www.researchgate.net/publication/372766691_Tutorials_on_Stance_Detection_using_Pre-trained_Language_Models_Fine-tuning_BERT_and_Prompting_Large_Language_Models>
8. Lan, X., Gao, C., Jin, D., & Li, Y. (2024). Stance Detection with Collaborative Role-Infused LLM-Based Agents. *Proceedings of the International AAAI Conference on Web and Social Media*, *18*, 891–903. https://doi.org/10.1609/icwsm.v18i1.31360