**What dataset are you using and how do you plan to collect it?**

**Data Description**

We sourced the dataset used in this study from GitHub. The authors provided the final, preprocessed dataset which is made up of 1000 disaster related news articles. Each article is labelled at word-level with 13 categories which include NaturalHazard, Location, Date, Floods, Death\_and\_Toll, AffectedPopulation, InfrastructureDamage, CollapsedStructure, Fire, PowerOutage, RoadBlocked, MissingPersons, and WaterShortage.

The articles were collected through the AYLIEN News API, and another 9800 articles from CNN were scraped using Octoparse and they were used to train the specialised word embeddings. The dataset has around 321000 labelled words in total, which were split into 80% training data and 20% testing data. They created a gold standard corpus by manually annotating 50 articles from the original dataset, and the rest of the articles were automatically annotated using machine learning tools on Tagtog and then they were verified by domain experts.

The data cleaning process was done by removing blanks and duplicates, and word tokenisation but they kept the punctuations. To improve the performance of the model they converted numbers to text and added features like part-of-speech (POS) tagging with NLTK and casing information to help identify proper nouns in the data. A custom skip-gram Word2Vec model was then trained using the CNN articles for the word embeddings as well as character-level embeddings, and this created 100-dimensional vectors for each token.

**What method/algorithm are you proposing? Are there existing implementations? How to modify/improve them?**

**Methodology proposed by the authors**

The project was treated as a sequence labelling problem and each word token is labelled as ‘B’ for the beginning of an entity phrase, ‘I’ for their position inside the entity, and non-relevant entities are labelled as ‘O’. The model combines a few features as an input to the model, and these include a custom Skip-Gram Word2Vec for the contextual word embeddings, character-level CNN embeddings, and POS tagging and casing using NLTK.

The features are then used in the Bi-directional Long Short-Term Memory (BiLSTM) network that has 100 hidden neurons so that it gets the the past and the future contextual information. They added a Multihead SelfAttention layer at the top layer to help the model recognise long-distance relationships between entities. This layer then goes through the Conditional Random (CRF) layer which uses the ‘Viterbi’ algorithm to make sure that label sequencing is accurate and consistent. The outcome of the CRF layer then gives us the final tagging score for each disaster entity.

The model was trained using grid the search method, a process that is usually used to optimise hyperparameters. The hyperparameters that were selected for training this model are a learning rate of 0.001, 50 epochs, the rmsprop optimiser, 150 hidden neurons in the dense layer, a ReLu activation function, a dropout rate of 0.2, and early stopping after 3 epochs. Training this model took about 2 hours on Google Colab GPU. The model was evaluated using the Precision, Recall, Accuracy and F1-score metrics and compared to the baseline BiLSTM, BiLSTM-ATTN, and BiLSTM-CRF models.

The BiLSTM-ATTN-CRF model may have outperformed the other models, but it is much slower and has high computational costs. To improve this model, we will explore a much faster and simpler method which combines DistilBERT, a distilled version of BERT, by Sanh et al. (2019) with Low-Rank Adaptation (LoRA) by Hu et al. (2021). The purpose of this combination is to create an efficient NER model which will be fine-tuned for our disaster-related data. The model is much smaller and quicker to train than the model proposed by the authors.

**Reference**

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Hu, E. S., Shen, Y., Wallis, P., Zeyuan Allen-Zhu, Li, Y., Wang, S., & Chen, W. (2021). *LoRA: Low-Rank Adaptation of Large Language Models.*

‌