## Experiments and Current Work

We are intending to perform some experiments to help achieve our objectives. These experiments should enable us to explore the following:

* Which ensemble learning technique yields the best prediction result in a multilingual setting.
* The performance of each of the used base classifiers as compared to the ensemble classifier.
* The transferability of the ensemble classifier to other languages exploiting a shared embedding space such as XLM-R.
* The difference in performance of the ensemble classifier when transferred to a parallel (machine translated) Arabic corpus and another unparallel Arabic corpus.
* Since Arabic is typologically different than English, therefore, we aim at examining the approach with a third language that is closer to English.

Therefore, we will follow the below steps and record our insights accordingly. These steps are:

* Obtain an English labelled dataset for disaster tweets (CrisisNLP).
* Pre-process the data including data cleaning and preparing the data for binary classification (relevant/not relevant).
* Extract features that might contribute to enhancing the performance.
* Use a multilingual embedding language model (e.g., BERT, XLM-R).
* Apply one or more of the ensemble learning techniques for the classification.
* Compare the performance of each of the ensemble techniques as well as the individual base classifiers.
* Transfer the best performing technique to a parallel (translated CrisisNLP) vs. unparallel Arabic (Kawarith) corpora. Parallel data can be obtained using machine translation of the original English dataset.
* For generalisation check, we will apply transfer learning of the final model to a third language.

### **Data Handling and Pre-processing**

After a thorough investigation of the available datasets, we finally decided to consider **CrisisNLP** as an English benchmark dataset and translate it to Arabic using machine translation to generate a parallel Arabic corpus. Furthermore, we intend to use **Kawarith** dataset as an unparallel Arabic corpus that can be used to compare the performance of our model to the parallel data.

While exploring the CrisisNLP dataset, the following notes have been taken into consideration:

1. The data is saved as separate files for each disaster forming a total of 12 .csv files of data labelled by paid workers and 13 overlapping disasters’ files labelled by volunteers. These files contain the tweets text and labels of 9 classes. The number of samples varies for each disaster forming an average of 2000 tweets per disaster.
2. The data is imbalanced in terms of relevance labels in a way that the relevant tweets exceed the irrelevant ones. This is expected in such a multiclass dataset with 8 different labels categorising the relevant tweets’ content (e.g., casualties, cautions, donations, emotional support, etc.) opposed to 1 label for irrelevant tweets.

We initially intend to pre-process the datasets in terms of removing stop words and punctuations and binarizing the multiclass labels into 0 and 1 representing irrelevant and relevant samples respectively. We also need to obtain a combined dataset of all disasters in one file in order to train the model. To do so, we should balance each dataset before constructing the final dataset. Once we have that in hand, we can obtain an Arabic machine-translated parallel dataset.

Kawarith dataset, on the other hand, consists of 7 different .csv files, each containing multiclass labelled disaster tweets. The average number of samples per disaster is 1700 tweets. The authors have availed a binary classification of the data as well as processed tweets in terms of punctuation and URLs removal. However, the data is still imbalanced in terms of the number of relevant tweets to irrelevant ones.

The initial method followed for training the model to classify relevant and irrelevant tweets was by re-labeling a multi-class CrisisNLP dataset into irrelevant(label 0) if original dataset had label as not\_related\_or\_irrelevant and into relevant(label 1) if original dataset had any of the other 8 labels namely-

sympathy\_and\_emotional\_support, infrastructure\_and\_utilities\_damage, displaced\_people\_and\_evacuations, donation\_needs\_or\_offers\_or\_volunteering\_services, missing\_trapped\_or\_found\_people, injured\_or\_dead\_people, caution\_and\_advice, and other\_useful\_information.

But the scores after encoding the original dataset into relevant and irrelevant and training teh classifier was very poor as thye dataset was firstly very unbalanced- class 0 ie. irrelevant class was a lot less than class 1 ie. relevant class. Moreover the measures taken to balance the dataset ie. adversarial example generation and oversampling did not help as the irrelevant class was ten or more times less than the relevant class. Thus after training the classifier, the results were skewed as the precision, recall and F1 score of the relevant class came out to be fine while that of the irrelevant class came out to be 0 everytime.

To solve this issue we decided to do the relevance classification after training on the multi-class dataset instead of first converting the dataset to relevant and irrelevant classes. This small decision in itself helped a lot to solve the issue of unbalanced classes and improved scores greatly.

The reason why the classifier was not giving good scores the first time was primarily down to the fact that the data was very unbalanced and also that the combined data was highly confusing for the classifier to classify as relevant and irrelevant. So this time we combined related data and created 6 classes and trained the classifier on this 6-classed dataset.

The classes we formed were-

* not\_related\_or\_irrelevant [class 0]
* empathy\_and\_advice (by combining sympathy\_and\_emotional\_support and caution\_and\_advice classes) [class 1]
* infrastructure\_and\_utilities\_damage [class 2]
* human\_casualty (by combining displaced\_people\_and\_evacuations, missing\_trapped\_or\_found\_people and injured\_or\_dead\_people classes) [class 3]
* donation\_needs\_or\_offers\_or\_volunteering\_services [class 4]
* other\_useful\_information [class 5]

These new classes were much more balanced in general and in some cases we oversampled, undersampled, added adversarial examples to balance certain class labels and created our final datasets for training the classifier.

Datasets Sizes:

* 2013\_Pakistan\_eq:1021
* 2014\_California\_Earthquake:951
* 2014\_Hurricane\_Odile\_Mexico\_en:1012
* 2015\_Nepal\_Earthquake\_en:2383
* 2014\_India\_floods:1756
* 2015\_Cyclone\_Pam\_en:2609
* 2014\_Philippines\_Typhoon\_Hagupit\_en:2872
* 2014\_Chile\_Earthquake\_en: 1212
* 2014\_Pakistan\_floods: 1139

Size of the final combined dataset: 14955

### **Feature Extraction**

We aim at extracting features that effectively contribute to accuracy improvement. To obtain these, we are going to explore the effect of a set of features and decide whether to eliminate those that affect or do not enhance the prediction accuracy. Some of the features we plan to extract are

## Methodology

Once we have the datasets ready in terms of pre-processing and balancing, we are going to use a multilingual BERT-based embedding to convert the text into vectors to be fed in our model. As an initial experiment, we will examine our approach using a basic neural network model and try to adjust the parameters and fine-tune it until we obtain the best result. This model will then be evaluated on unseen Arabic test data (parallel and unparallel data). The final fine-tuned model will then be compared with different ensemble techniques to examine our hypothesis that the use of ensemble learning will lead to better results than using an individual base model.

**Need For Multilingual Models**

In this task we are doing relevance classification of a multi-lingual crisis related tweet corpus. We have data from languages including English, Arabic, French, Chilean to name a few. Thus we use a multi-lingual model and its corresponding tokenizer to handle the task.

The distinction between mono-lingual models and multi-lingual models is that a mono-lingual model is trained by masked language modelling on a corpus containing data of only one language. Whereas a multi-lingual model is trained on corpus containing multiple languages at the same time. This helps such models be more expressive and understand language ,semantics, relationships between languages in a better way.

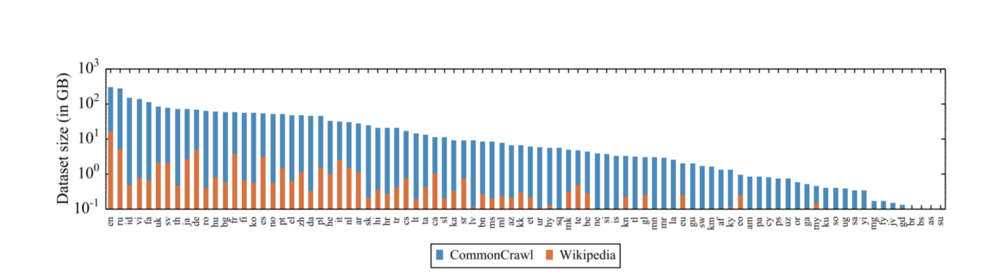
Examples of mono-lingual models include- RoBERTa-base(English only), French BERT(French only), German BERT(German only), etc.

Examples of multi-lingual models include- mBERT(can handle over 50 laguages), XLM-100 and XLM-RoBERTa(can handle over 100 laguages).

**Model Description and Training (XLM-RoBERTa)**

The Facebook AI team released the[**XLM-RoBERTa**](https://arxiv.org/pdf/1911.02116.pdf) model in November 2019 as an update to their original XLM-100 model. Both are transformer based encoder language models and rely on Masked Language Modelling(MLM) objective to train and can process text from 100 separate languages. The main difference between the two was in the amount of data used to train them. XLM-RoBERTa has been trained on a significantly larger corpus obtained by cleaning data from CommonCrawl and in total took up 2.5Tb of space. XLM-100 in contrast was trained on the Wiki-100 corpus which was orders of magnitude smaller than CommonCrawl. This was particularly important in the performances of the low-resource languages which had very less data in Wiki-100 but significantly more data in CommonCrawl. The RoBERTa part in XLM-RoBERTa comes from the fact that the training routine of this model was same as the monolingual RoBERTa model, specifically the sole training objective is Masked Language Modelling.

XLM-RoBERTa uses a large shared SentencePiece model to tokenize text from any of the 100 supported languages instead of having a slwe of language specific tokenizers as was the case in XLM-100. Also validation perplexity was no longer used as the stopping criteria during training since the researchers found that the downstream performance continues to improve even when perplexity does not.



Hyper-Parameters Used:

Epochs- 3

Batch Size- 4

Learning Rate- 2 x e-5

Maximum Sequence Length- 512

Warmup Steps- 100

Weight Decay - 0.01

## Evaluation

We will use the accuracy, precision, recall and F1-measure and macro-average to evaluate the performance of the different models.

Accuracy is just the percentage of the predictions that were made correctly expressed as a decimal number between 0(0% of predictions were correct) and 1(100% of the predictions were correct).

***Accuracy = (# of correct predictions) / (# of total predictions)***

To connect Accuracy to the other metrics, we first define True Positive(TP), True Negative(TN), False Positive(FP) and False Negative(FN).

In binary classification if positive(relevant in our case is 1) and negative(irrelevant in our case is 0) then,

TP = # of predictions where predicted = 1 and original label = 1

TN = # of predictions where predicted = 0 and original label = 0

FP = # of predictions where predicted = 1 and original label = 0

FN = # of predictions where predicted = 0 and original label = 1

Thus the new definition of accuracy is:

***Accuracy = (TP + TN) / (TP + TN + FP + FN)***

Other Metrics for Sensitivity and Specificity

Sensitivity is the same thing as recall or the true positive rate(TPR), and in lame terms it is the fraction of true positives to the total predictions that were correct.

***Sensitivity = Recall = TPR = TP / (TP + FN)***

Precision is the ratio of the true positives to the number predictions that are positive.

***Precision = TP / (TP + FP***)

F1 score is the harmonic mean of the recall and precision and takes values between 0 and 1.

***F1-Score = 2 x (Recall x Precision) / (Recall + Precision) = TP / (TP + ½ (FP + FN))***

We can see that if either recall or precision is 0, F1-score is also zero

Macro-averages are the average of the pre-class Recall, Precision and F1-Scores.

### **Results**

The initial method followed for training the model to classify relevant and irrelevant tweets was by re-labeling a multi-class CrisisNLP dataset into irrelevant(label 0) if original dataset had label as not\_related\_or\_irrelevant and into relevant(label 1) if original dataset had any of the other 8 labels namely-

sympathy\_and\_emotional\_support, infrastructure\_and\_utilities\_damage, displaced\_people\_and\_evacuations, donation\_needs\_or\_offers\_or\_volunteering\_services, missing\_trapped\_or\_found\_people, injured\_or\_dead\_people, caution\_and\_advice, and other\_useful\_information.

But the scores after encoding the original dataset into relevant and irrelevant and training teh classifier was very poor as thye dataset was firstly very unbalanced- class 0 ie. irrelevant class was a lot less than class 1 ie. relevant class. Moreover the measures taken to balance the dataset ie. adversarial example generation and oversampling did not help as the irrelevant class was ten or more times less than the relevant class. Thus after training the classifier, the results were skewed as the precision, recall and F1 score of the relevant class came out to be fine while that of the irrelevant class came out to be 0 everytime.

To solve this issue we decided to do the relevance classification after training on the multi-class dataset instead of first converting the dataset to relevant and irrelevant classes. This small decision in itself helped a lot to solve the issue of unbalanced classes and improved scores greatly.

The reason why the classifier was not giving good scores the first time was primarily down to the fact that the data was very unbalanced and also that the combined data was highly confusing for the classifier to classify as relevant and irrelevant. So this time we combined related data and created 6 classes and trained the classifier on this 6-classed dataset.

The classes we formed were-

* not\_related\_or\_irrelevant [class 0]
* empathy\_and\_advice (by combining sympathy\_and\_emotional\_support and caution\_and\_advice classes) [class 1]
* infrastructure\_and\_utilities\_damage [class 2]
* human\_casualty (by combining displaced\_people\_and\_evacuations, missing\_trapped\_or\_found\_people and injured\_or\_dead\_people classes) [class 3]
* donation\_needs\_or\_offers\_or\_volunteering\_services [class 4]
* other\_useful\_information [class 5]

These new classes were much more balanced in general and in some cases we oversampled, undersampled, added adversarial examples to balance certain class labels and created our final datasets for training the classifier.

The experiments that we performed were on the following datasets:

* 2014\_Hurricane\_Odile\_Mexico\_en dataset.

The classification scores on the six classes are as follows:

## precision recall f1-score support

## 0 1.000 0.865 0.928 37

## 1 0.857 0.698 0.769 43

## 2 0.519 0.730 0.607 37

## 3 0.811 0.789 0.800 38

## 4 0.773 0.773 0.773 44

## 5 0.294 0.286 0.290 35

## accuracy 0.697 234

## macro avg 0.709 0.690 0.694 234

## weighted avg 0.719 0.697 0.703 234

Finally we classify all the test predictions that are classified as 0 as irrelevant and the remaining five classes(ie. 1,2,3,4, and 5) as relevant.

## **precision recall f1-score support**

## **0 1.000 0.865 0.928 37**

## **1 0.975 1.000 0.987 197**

## **accuracy 0.979 234**

## **macro avg 0.988 0.932 0.958 234**

## **weighted avg 0.979 0.979 0.978 234**

Clearly we get very good scores as the individual class Precision, Recall and F1 scores all exceed 85% and the macro-average scores all cross 93%.

* **Pakistan Earthquake**

**6 classes**

precision recall f1-score support

0 0.750 0.727 0.738 66

1 0.581 0.529 0.554 34

2 0.880 1.000 0.936 22

3 0.812 0.765 0.788 51

4 0.750 0.818 0.783 22

5 0.561 0.605 0.582 38

accuracy 0.721 233

macro avg 0.722 0.741 0.730 233

weighted avg 0.720 0.721 0.720 233

**two classes**

precision recall f1-score support

0 0.750 0.727 0.738 66

1 0.893 0.904 0.899 167

accuracy 0.854 233

macro avg 0.822 0.816 0.819 233

weighted avg 0.853 0.854 0.853 233

* **California Earthquake**

precision recall f1-score support

0 0.261 0.400 0.316 15

1 0.800 0.889 0.842 27

2 0.765 0.796 0.780 49

3 0.892 0.892 0.892 37

4 0.966 1.000 0.982 56

5 0.711 0.509 0.593 53

accuracy 0.781 237

macro avg 0.732 0.748 0.734 237

weighted avg 0.792 0.781 0.781 237

**2 classes**

precision recall f1-score support

0 0.261 0.400 0.316 15

1 0.958 0.923 0.940 222

accuracy 0.890 237

macro avg 0.609 0.662 0.628 237

weighted avg 0.914 0.890 0.901 237

* **Chile Eartquake**

precision recall f1-score support

0 0.931 0.818 0.871 66

1 0.810 0.791 0.800 43

2 0.952 1.000 0.976 40

3 0.912 0.939 0.925 33

4 0.957 1.000 0.978 22

5 0.705 0.795 0.747 39

accuracy 0.872 243

macro avg 0.878 0.891 0.883 243

weighted avg 0.876 0.872 0.873 243

2 classes

precision recall f1-score support

0 0.931 0.818 0.871 66

1 0.935 0.977 0.956 177

accuracy 0.934 243

macro avg 0.933 0.898 0.913 243

weighted avg 0.934 0.934 0.933 243

* **India Floods**

precision recall f1-score support

0 0.965 0.865 0.912 96

1 0.819 0.983 0.894 60

2 0.927 1.000 0.962 51

3 0.930 0.909 0.920 44

4 0.844 1.000 0.915 54

5 0.812 0.553 0.658 47

accuracy 0.889 352

macro avg 0.883 0.885 0.877 352

weighted avg 0.891 0.889 0.884 352

**2 classes**

precision recall f1-score support

0 0.965 0.865 0.912 96

1 0.951 0.988 0.969 256

accuracy 0.955 352

macro avg 0.958 0.926 0.941 352

weighted avg 0.955 0.955 0.954 352

* **Pakistan Floods**

precision recall f1-score support

0 0.966 1.000 0.982 28

1 0.767 0.719 0.742 32

2 0.565 0.722 0.634 18

3 0.867 0.830 0.848 47

4 0.660 0.733 0.695 45

5 0.608 0.534 0.569 58

accuracy 0.732 228

macro avg 0.739 0.756 0.745 228

weighted avg 0.734 0.732 0.731 228

**2 classes**

precision recall f1-score support

0 0.966 1.000 0.982 28

1 1.000 0.995 0.997 200

accuracy 0.996 228

macro avg 0.983 0.998 0.990 228

weighted avg 0.996 0.996 0.996 228

* **Philippines**

precision recall f1-score support

0 0.963 0.994 0.978 312

1 0.775 0.721 0.747 86

2 0.500 0.611 0.550 18

3 0.921 0.814 0.864 43

4 0.613 0.826 0.704 23

5 0.683 0.602 0.640 93

accuracy 0.857 575

macro avg 0.742 0.761 0.747 575

weighted avg 0.858 0.857 0.856 575

**2 classes**

precision recall f1-score support

0 0.963 0.994 0.978 312

1 0.992 0.954 0.973 263

accuracy 0.976 575

macro avg 0.977 0.974 0.975 575

weighted avg 0.976 0.976 0.976 575

* **Cyclone**

precision recall f1-score support

0 0.929 0.738 0.823 107

1 0.789 0.866 0.826 82

2 0.864 0.864 0.864 88

3 0.882 0.962 0.920 78

4 0.824 0.813 0.819 75

5 0.660 0.717 0.688 92

accuracy 0.820 522

macro avg 0.825 0.827 0.823 522

weighted avg 0.827 0.820 0.820 522

**2 classes**

precision recall f1-score support

0 0.929 0.738 0.823 107

1 0.936 0.986 0.960 415

accuracy 0.935 522

macro avg 0.933 0.862 0.892 522

weighted avg 0.935 0.935 0.932 522

* **Nepal**

precision recall f1-score support

0 0.922 0.855 0.887 110

1 0.833 0.840 0.837 119

2 0.865 0.970 0.914 66

3 0.864 0.864 0.864 110

4 0.825 0.792 0.808 101

5 0.552 0.570 0.561 93

accuracy 0.811 599

macro avg 0.810 0.815 0.812 599

weighted avg 0.813 0.811 0.812 599

**2 classes**

precision recall f1-score support

0 0.922 0.855 0.887 110

1 0.968 0.984 0.976 489

accuracy 0.960 599

macro avg 0.945 0.919 0.931 599

weighted avg 0.959 0.960 0.959 599

Scores on the combined dataset:

**Final Scores for classification of tweets into Relevant and Irrelevant:**

| **Dataset** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| --- | --- | --- | --- | --- |
| **2015\_Nepal\_Earthquake** | 0.960 | 0.945 | 0.919 | 0.931 |
| **2015\_Cyclone\_Pam** | 0.935 | 0.933 | 0.862 | 0.892 |
| **2014\_California\_Earthquake** | 0.890 | 0.609 | 0.662 | 0.628 |
| **2014\_Chile\_Earthquake** | 0.934 | 0.933 | 0.898 | 0.913 |
| **2014\_India\_floods** | 0.955 | 0.958 | 0.926 | 0.941 |
| **2014\_Pakistan\_floods** | 0.996 | 0.983 | 0.998 | 0.990 |
| **2014\_Philippines\_Typhoon\_Hagupit** | 0.976 | 0.977 | 0.974 | 0.975 |
| **2014\_Hurricane\_Odile\_Mexico** | 0.979 | 0.988 | 0.932 | 0.958 |
| **2013\_Pakistan\_eq** | 0.854 | 0.822 | 0.816 | 0.819 |

Scores for the combined datasets

precision recall f1-score support

0 0.890 0.870 0.880 483

1 0.790 0.779 0.785 299

2 0.830 0.838 0.834 210

3 0.818 0.869 0.843 222

4 0.796 0.833 0.814 239

5 0.639 0.617 0.628 342

accuracy 0.798 1795

macro avg 0.794 0.801 0.797 1795

weighted avg 0.797 0.798 0.797 1795

Second run

precision recall f1-score support

0 0.922 0.855 0.887 483

1 0.763 0.796 0.779 299

2 0.829 0.852 0.840 210

3 0.825 0.869 0.846 222

4 0.816 0.833 0.824 239

5 0.630 0.629 0.630 342

accuracy 0.801 1795

macro avg 0.797 0.806 0.801 1795

weighted avg 0.803 0.801 0.801 1795

2 classes

precision recall f1-score support

0 0.922 0.855 0.887 483

1 0.948 0.973 0.961 1312

accuracy 0.942 1795

macro avg 0.935 0.914 0.924 1795

weighted avg 0.941 0.942 0.941 1795

## Discussion

We can see that the method we applied of not dividing the data into two classes before training gave very good results. Rather we grouped data according to similarity of context from 9 classes to 6 classes and trained the language model to do sequence classification. We then classified the predicted labels into relevant or irrelevant and the scores prove the effectiveness of our method.

In each dataset we can observe that some classes have lower scores than the macro and weighted average and the primary reason for that is the imbalance in the number of training examples of certain classes compared to others thus leading to poorer training and bad classification for the test examples of those classes.

In the future we will look into ways to make the datasets more balanced and solve the issue of lower scores for certain classes as well. Besides, we will also perform experiments using some other multilingual models and create an ensemble to make the overall classification more robust. Finally, we will evaluate the zero-shot performance of our multilingual models trained on English corpus on some other languages supported by XML-R.