Final Report - Peace Speech Project

Hongling Liu (hl3418), Haoyue Qi (hq2180),
Xuanhao Wu (xw2744), Yuxin Zhou (yz3904), Wenjie Zhu (wz2536)

Dec 18th, 2021

I. Problem Definition and Progress Overview

In the current world of rising conflicts and wars, peacekeepers are actively conducting research on hate speech analysis in seeking to maintain robust and peaceful communities [1]. Such hate speech dataset is accessible on the internet including but not limited to news articles, blogs and social media posts. Meanwhile, peace speech, which is also an important yet available piece of language assets, commonly gets neglected. Capstone students in the past years had worked with Professor Coleman's team to conduct pilot studies that showed promising powers of peace speech in distinguishing the peacefulness of societies. This year, we continue the previous research with a much richer dataset and two new objectives.

First, we explored the performance of more state-of-art NLP models on classifying English articles from high-peace and low-peace countries. We presumed that a high performance model would indicate the existence of language differences in the articles from countries with different peace levels. Different from the study conducted by previous years' capstone students, we primarily focus on encoding sentences using contextual approach like BERT rather than bag-of-words method, and then using the generated embeddings as inputs to explore different classification modeling approaches. Next, we dove into studying how the interrelationship among words affect the previous classification results by evaluating the models' performance over randomly shuffled articles. A drop in the performance would then show the ordering of words in one article to be a strong peacefulness indicator.

Our progress on the above two objectives are intertwined and going in parallel according to the procedure logic shown in *Figure 1-1*. In this final report, we will briefly summarize the first two of the four steps in the procedure pipeline, and put more focus on finalizing our analysis on the experiment outcomes, and give some additional findings and side notes to conclude this project. You may find the details of the technical methodologies in our previous progress reports.

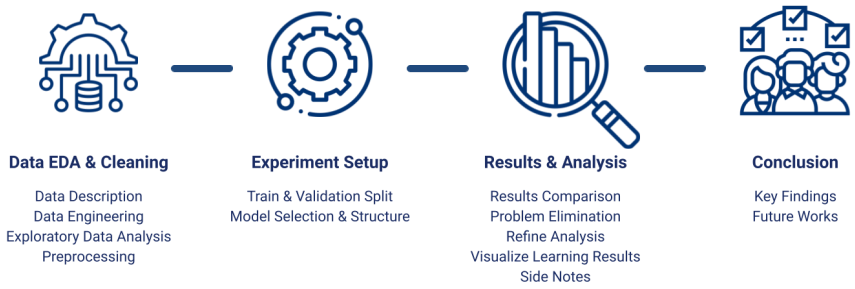


Figure 1-1 Procedure Overview

II. Data EDA & Cleaning

II.I. Data Description

The data we studied are English news articles and blogs served by LexisNexis and are stored on the AWS S3 bucket as .json files in two separate directories: *highPeace* and *lowPeace*. The *highPeace* is a directory with news articles from the top 10 high peace countries around the world based on the *rule-of-5*[1] metric, and the same logic follows for the other folder (see *Table 2-1* for the list of countries). Each .json object is one data point. There are ~33M articles (.json files) from high peace countries and ~57M articles from low peace countries. Among the columns contained in each file, we are interested in the *title*, *content*, *wordCount*, and *country* columns. We used AWS SageMaker cloud computation platform throughout this project, since it has a more compatible connection with S3 bucket and a configurable number of GPUs and other computational resources.

Low Peace			High Peace		
● Afghanistan	● Iran	● Uganda	● Austria	● Denmark	● Norway
● Congo	● Kenya	● Zimbabwe	● Australia	● Finland	● Sweden
● Guinea	● Nigeria		● Belgium	● Netherlands	
● India	● Sri Lanka		● Czech Republic	● New Zealand	

Table 2-1: List of Article Countries in Real Data

II.II. Data Engineering

As introduced above, each data point is stored as an individual .json file with less than 20 KB per file on average. This storage format could cause a so-called *Small Files Problem* when loading the data, which means the milliseconds to open, read, and close each file could accumulate to a large runtime overhead if a program has to execute those steps in high frequency. The existence of this problem adds difficulties for us to view the dataset entirely and perform further analysis and operations.

We proposed to read in a large set of files from the real dataset at once and save them together as one large file for re-use in the rest of this project to avoid encountering the *Small File Problem* repeatedly. We also made several adjustments to our code to reduce the initial small file fetching latency when compiling this target large file: 1) use UltraJSON (*ujson*) package instead of the traditional json package for faster content loading; 2) parallelize the I/O bound loading task via multithreading; 3) switch to a larger notebook instance (*ml.m5.4xlarge*) with a wider network bandwidth and a larger memory space. The optimized program gave us a ~24x speed up overall. In concern of the budget, we had to limit the amount of data we scanned and studied in later steps to 1M per peace category (2M in total) and only kept the columns of interest needed for lightweight file transfer and storage. But with a relaxed budget constraint, one can re-use the pipeline we constructed here to extract and compress more of the data files.

[1] For each country, scores for indices such as Global Peace Index, the Positive Peace Index, etc., are computed. Then countries are placed into low-, mid-, and high-peace categories for each index. If a country has at least 5 indices categorized as low-peace, and no index in high-peace, then the country is low-peace. Same logic follows for high-peace.

II.III. Exploratory Data Analysis (EDA)

Given the runtime constraint, it would be a challenge to perform an EDA on all 90M data points. Thus, we proposed the iterative EDA method in *Figure 2-1*. After a few iterations of this process, the data distribution we observed would converge to its true value within a confidence interval due to the increasing sample size per step. By studying the convergence trend, we could estimate the overall data distribution from the EDA result on just the 2M data extracted in II.II:

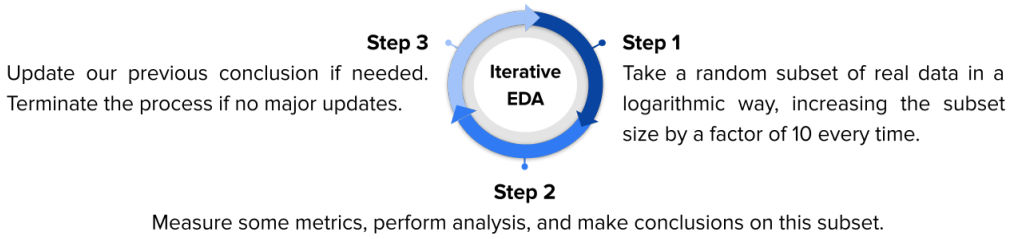


Figure 2-1: Iterative EDA Process

As mentioned above, our fields of interest are *title*, *content*, *wordCount*, and *country*. Having noticed that the article distribution across countries has been collected and analyzed before, we focused more on the word count distribution in our EDA. *Table 2-2* is a summary on the sample article count distribution, sample average word count, and the 95% confidence interval (C.I.) for the mean word count per country over the sampled 2M data points we scanned in II.II. The data distribution is highly imbalanced across countries within each peace group. In the low peace country group, non-Indian data only makes up less than 10% of the category total, and only 1 data point from Guinea appears among the 1M low peace samples. The highPeace country group is slightly more balanced, but the dominance of Australian data is still obvious. Though we were lacking data from some countries, we still luckily have enough samples from the majority of countries to narrow down the width of the word count C.I. to ~30 words. The valid wordCount C.Is in left tables barely overlap with any of the wordCount C.Is in the right table, suggesting that there might be a length difference in articles from low peace and high peace countries.

country	sample_perc	avg_wordCount	wordCount_CI_95
Afghanistan	0.2328	221.809708	(213.6184, 230.001)
Congo	0.4985	489.747041	(477.5319, 501.9622)
Guinea	0.0001	3121.000000	(nan, nan)
India	91.7565	422.663411	(421.0335, 424.2933)
Iran	2.3680	484.964443	(479.5853, 490.3436)
Kenya	0.5157	607.522397	(594.3597, 620.6851)
Nigeria	2.7022	579.226482	(571.1824, 587.2705)
Sri Lanka	1.3659	831.227542	(810.2969, 852.1582)
Uganda	0.2902	575.023777	(560.7757, 589.2719)
Zimbabwe	0.2701	566.279156	(551.8202, 580.7381)

country	sample_perc	avg_wordCount	wordCount_CI_95
Australia	76.0521	884.916874	(882.2838, 887.55)
Austria	0.4210	1240.742993	(1200.2169, 1281.2691)
Belgium	5.3665	703.747545	(683.1177, 724.3774)
Czech Republic	0.5935	415.039596	(402.2787, 427.8005)
Denmark	1.3845	1281.044854	(1252.5792, 1309.5105)
Finland	1.0927	1807.282694	(1767.3396, 1847.2258)
Netherlands	0.8204	1566.497075	(1530.0766, 1602.9176)
New Zealand	9.0610	526.680709	(522.3605, 531.0009)
Norway	1.2701	1851.772459	(1815.7198, 1887.8251)
Sweden	3.9382	1688.450307	(1673.9758, 1702.9249)

Table 2-2: Data Distribution Overview
(Left: Low Peace Countries, Right: High Peace Countries)

We performed the iterative EDA step on the word count and plotted the boxplot for each peace group per iterative step in *Figure 2-2*. As more samples were included in the analysis, more large-value

samples appears for both groups, but the quartile lines bounding the box always stabilized at ~(130, 260, 420) for the low peace group, and ~(240, 495, 1150) for the high peace group. The long growing tail points out a possibility that the true distribution of word counts has an infinite variance, which makes sense in reality since there is no hard upper limit posed on news article length in general.

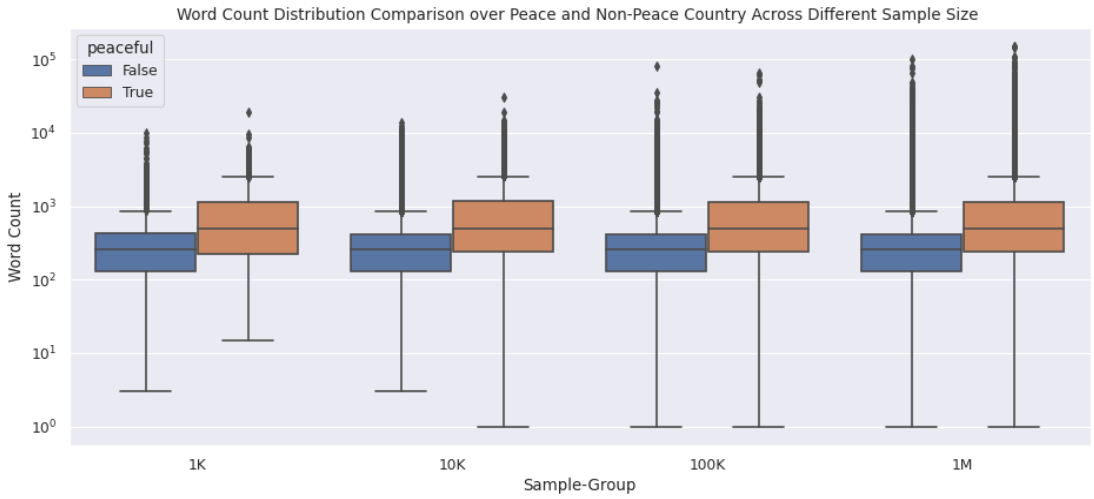


Figure 2-2: Iterative Boxplot per Peace Group

We suspected earlier that high peace countries might have longer articles than low peace countries. To validate this hypothesis, we performed a one-sided distribution-agnostic Wilcoxon signed-rank test. The result is shown in *Figure 2-3*. The p-value is so small that it's rounded to 0 in the output. Thus we have strong evidence to accept this hypothesis.

```
scipy.stats.wilcoxon(hp_df.wordCount, y=lp_df.wordCount,
                     zero_method='wilcox', correction=False, alternative='greater', mode='auto')
```

```
WilcoxonResult(statistic=374641694823.0, pvalue=0.0)
```

Figure 2-3: Wilcoxon Signed-rank Test Result

Ultimately, the EDA results enlightened us with the following two suggestions on the next-step procedures:

1. **Class Balancing Approach**

   As our objective is to create a classification model to distinguish between high peace countries and low peace countries, the imbalance between the number of articles from different countries would cause a class imbalance, and thus augment model's overfitting tendency, lead to a bad performance on the test set among metrics like accuracy, precision and recall. The theoretical data balancing approach suggests taking the same amount of articles from each country using resampling. Since there is an equal number of countries in each peace group, the outcome dataset would be balanced between countries within each peace group as well as across peace groups. However, the obstacles we discussed in section II.II impedes us from finding enough data points for underrepresented countries, such as Guinea in *Table 2-2*. Though balancing data by country was our top choice, we switched to using the alternative approach in most cases if not mentioned explicitly, which is to balance the dataset across peace groups. The data undersampling procedure

that produced the 2M dataset was designed to produce a class balanced dataset, and any further undersampling we performed in later steps on this 2M data would preserve this class balance.

2. **Appropriate BERT Token Length**
   Training BERT text encoders from scratch is resource costly. The pre-trained off-shelf ones usually take a unified input length capped by 512 tokens. Unifying the length to maximum token length preserves data information seen by the encoder to the maximum. However, if low peace countries tend to have shorter articles, then the [PAD] tokens for filling articles length will be more frequently appearing in inputs from low peace countries than in those from the high peace group. Though BERT's attention mask is applied to ignore the [PAD] tokens, McCormick, the founder of McCormick.AI, noted in his tutorial blog post [2] that it is still an unknown question of whether the [PAD] tokens have any effects in BERT's prediction result due to the difficulty in explaining deep learning models. So we shrunk the input length to 128 to confound these input length differences in two peace groups and speed up the BERT training time on our large dataset.

II.IV. Preprocessing

Article title is a summary with dense information related to the content. The sentences the model made predictions upon were the concatenations of article titles and contents. The natural language samples in our dataset were published news articles and have been pre-cleaned by LexisNexis. From a data-quality perspective, there are barely any typos and missing data in this dataset. But additional cleansing was necessary for data to meet the requirements of our experiment and to be compatible with BERT.

Many encoders belonging to the BERT family are powerful enough to interpret inputs with special characters and numbers. Furthermore, according to Alzahrani et al.[3], BERT performs the best on a binary classification task when its input is raw text. Camacho-Collados et al.[4] also achieved a similar conclusion for general neural network models in topic categorization and sentiment detection tasks. Though preprocessing techniques such as removing stopwords is commonly seen in handling inputs for other NLP models, such methods destructs the original input context and results in losing information. For example, the negation stopword "*not*" is often related to a shift in sentiment and might be related to the peacefulness pattern we are looking for. Since we would start with fine-tuning a BERT-family encoder that has been pre-trained on raw natural language, overly preprocessing our data would only negatively impact its ability in learning such text patterns. Yet, there are many named entities in the articles that might disclose the information such as the name of a city, an author who frequently writes reports for a country, or an organization that's doing business in a country. To avoid the encoder model from learning those biased aspects of the data, we removed that information from our input. Besides that, we converted all British English to American English so the spelling of the word won't disclose the region where the article comes from, and the two versions of a word can be mapped onto the same embeddings which reduces the sparsity in text mapping.

The entire preprocessing pipeline is in *Figure 2-4*. Note that the English NLP models of different sizes from SpaCy don't have a significant performance variation in named entity recognition (NER) tasks as shown in *Table 4-1*, we used the small model *en-core-web-sm* for NER purpose to save

computation resources. As stated, the removed targeted entities were those that might disclose country-related information: organizations, geographical names, and person names. The output of this pipeline were English sentences ready to be fed into a pre-built preprocessor paired with each BERT family encoder. Both the pre-trained encoders and the paired preprocessors were off-shelt from Tensorflow Hub and Hugging Face depending on the specific chosen member of the BERT family, which we will discuss in a later section III.II.

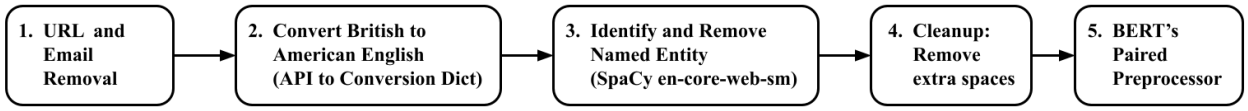


Figure 2-4: Preprocessing Pipeline Design for Encoders in BERT family

Model	Size	Tokenization Accuracy	NER precision	NER recall	NER F1
en-core-web-sm	12 MB	1.00	0.84	0.83	0.84
en-core-web-md	43 MB	1.00	0.85	0.85	0.85
en-core-web-trf	438 MB	1.00	0.90	0.90	0.90
en-core-web-lg	741 MB	1.00	0.85	0.85	0.85
Source: SpaCy documentation (https://spacy.io/models/en)					

Table 2-3: Comparison between SpaCy NER models of different sizes

There are two major types of tokenizers and two versions of encoders in the BERT family: WordPiece or SentencePiece used with cased or uncased encoders. Horan, the Machine Learning Engineer at Intercom, summarises a detailed comparison between the types of encoders in his blog tutorial [5], and we will not go into too much detail here, but rather we present *Figure 2-5* here to highlight the major differences. The sample tokenization results from BERT-base-uncased, BERT-base-cased and RoBERTa-base models are listed from top to bottom. The prior two uses an uncased/cased WordPiece tokenizer while the last one uses a SentencePiece tokenizer. Note how the cased and uncased tokenizers tokenize the word "Powerful" differently. Depending on which version of the BERT we use, its paired preprocessor will lower-casing all input letters for us if necessary. Then it will send the sentence to the tokenizer to perform a subword tokenization based on maximum likelihood estimation which leads to results similar to but more flexible than stemming and lemmatization. For example, the word "*embedding*" is splitted into: "*em*"(prefix), "*##bed*"(root), "*##ding*"(suffix), "*##s*"(plural) by wordPiece tokenizer, or : "embed"(prefix), "d", "ings"(suffix) by sentencePiece tokenizer. Each of the tokens have its own word embedding, position embedding, and segment embedding. The encoder sums up those embeddings internally to resemble the meaning of this word in the context it appears in. Therefore, we don't need to worry about stemming and lemmatizing words ourselves.

```
from transformers import BertTokenizer, AutoTokenizer
tz_uncased = BertTokenizer.from_pretrained("bert-base-uncased")
sentence = 'BERT Encoder is a Powerful encoder that produces word embeddings!'
print(tz_uncased.tokenize(sentence))
```

```
['bert', 'en', '##code', '##r', 'is', 'a', 'powerful', 'en', '##code', '##r', 'that', 'produces', 'word', 'em', '##bed', '##ding', '##s', '!']
```

```
tz_cased = BertTokenizer.from_pretrained("bert-base-cased")
print(tz_cased.tokenize(sentence))
```

```
['B', '##ER', '##T', 'En', '##code', '##r', 'is', 'a', 'Power', '##ful', 'en', '##code', '##r', 'that', 'produces', 'word', 'em', '##bed', '##ding', '##s', '!']
```

```
tz_roberta = AutoTokenizer.from_pretrained("roberta-base")
print(tz_roberta.tokenize(sentence))
```

```
['BER', 'T', 'ĠEnc', 'oder', 'Ġis', 'Ġa', 'ĠPowerful', 'Ġenc', 'oder', 'Ġthat', 'Ġproduces', 'Ġword', 'Ġembed', 'd', 'ings', '!']
```

Figure 2-5: Sample Tokenization Results

Before preprocessing
Sadia Dada, Director Communications PMPKL, said, "The challenge of littering is not new to our country nor the efforts to combat it. Lack of awareness and infrastructure for disposal are key drivers abetting this bad habit adding that most important is how minor changes in the way we work can create room for each and every individual of society to be a part of it."\n\nPublished by HT Digital Content Services with permission from Daily Times. For any query with respect to this article or any other content requirement, please contact Editor at contentservices@htlive.com
After preprocessing
, Director, said," The challenge of littering is not new to our country nor the efforts to combat it. Lack of awareness and infrastructure for disposal are key drivers abetting this bad habit adding that most important is how minor changes in the way we work can create room for each and every individual of society to be part of it." Published by with permission from.
After shuffling
of important be each of key in to country abetting this it." of that way Director, from. society combat said," and room challenge minor every littering it. bad habit awareness nor part are of for we , how our not drivers disposal the efforts most create adding infrastructure the can for is The Lack and to individual Published permission to changes with by is work new
* Highlighted parts are deleted in preprocessing

Table 2-4: Sample article before preprocessing, after preprocessing, and after shuffling

The output articles of the preprocessing pipeline for BERT are still human readable as the majority of the contexts are preserved. For the shuffling sentences, we simply split the preprocessed sentences into tokens by space, randomly shuffle the tokens, and join them back to sentences by space. *Table 2-4* shows how an example was modified in each step.

III. Experiment Setup

III.I. Train & Validation Split

During our exploration, SpaCy's default named entity recognizer appeared to fail recognizing all of the named entities. Some edge cases include: 1) all capitalized country names, and 2) the adjective form of a country or a location name. In order to alleviate the effect of these named entity residuals,

we employed two approaches for both of our project objective tasks to split the training and validation dataset: traditional random split and split by country (*Table 3-1, Figure 3-1, Figure 3-2*).

	Random Split	Split by Country
Train on Data from...	All countries	A subset of countries
Validate on Data from...	All countries	A subset of countries disjoint from countries in training set

Table 3-1: Initial Two Split Approaches

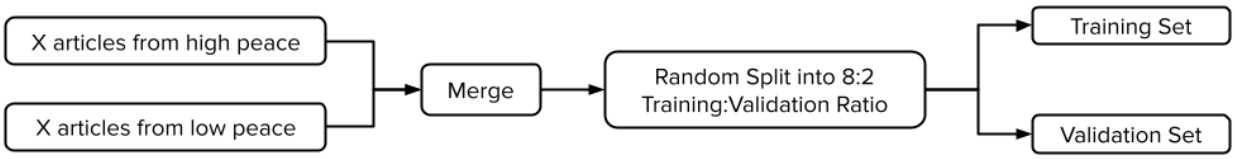


Figure 3-1: Random Split Illustration

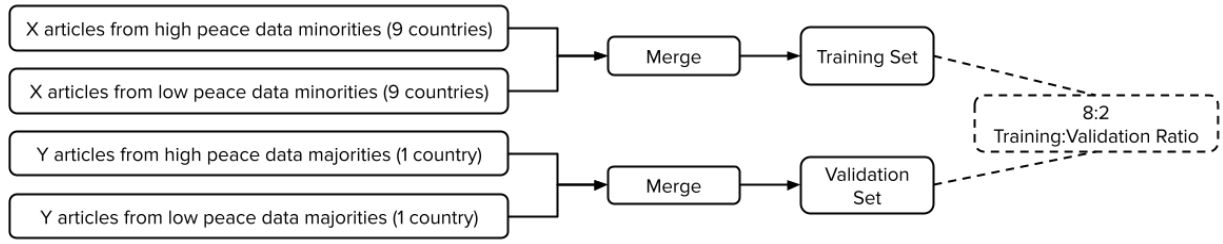


Figure 3-2: Split-by-Country Illustration

By using split by country approach, any named entities that have high correlation with a country in the validation set would not be learned during the training time, and hence would not be key features in the model's decision function. Residual named entities most frequently occurred in two countries with the most samples in each class: India and Australia, which we refer to as the "majorities" in the following report. Thus articles from the majorities constituted our validation set, while the samples from the rest of the countries, which we refer to as "minorities", were in our training set. Our presumption here is that if all countries in the same peace group all share some common language structures, then this approach should not weaken the strength of the model's performance.

III.II. Model Selection & Structure

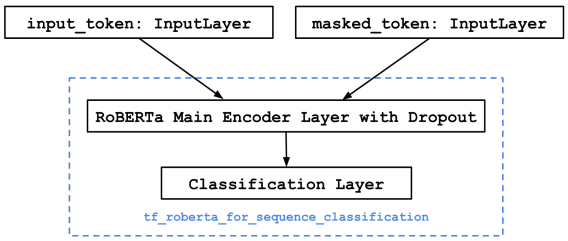
BERT, as a state-of-the-art technique, has many pre-trained variations available on Tensorflow Hub and Hugging Face open source libraries that cover a wide range of sizes, languages, and fields of expertise. BERT's performance is directly impacted by the level of similarity between the data used for pre-training and the actual data. We did some research online and narrowed down the candidates to the following three models listed in *Table 3-2* that are suitable for our news classification task.

Model	Platform	Dataset	Parameter	Word Bank	Past Performance
RoBERTa-base	HuggingFace (Official)	Pretrained on: ● English Wikipedia, BookCorpus, CC-News, OpenWebText, Stories	125M	50K	SST-2 Score: 94.8 MRPC Score: 90.2
experts/bert/wiki_books/onli	Tensorflow	Pretrained on: ● English Wikipedia, BookCorpus Fine-tuned on: ● QNLI Task	110M	30K	MRPC Score: 90+
textattack/distilbert-base-uncased-ag-news	HuggingFace (OpenSource)	Pretrained on: ● English Wikipedia, BookCorpus Fine-tuned on: ● Adversarial Attack Approach on News Category Classification	66M	30K	ag_news sequence classification: 94.7

Table 3-2: BERT Encoder Candidates

We finally picked the RoBERTa-base as our first choice, since this model has many solid papers that are supporting it, and it is managed by the HuggingFace Team which makes it more reliable than open source models. It could have a higher ability in interpreting news articles due to 63M CC English news articles in its training set. In addition, we got more information on its past performance on different General Language Understanding Evaluation (GLUE) tasks. We supposed that if the articles truly have some peaceful indication power, then the peacefulness might be captured by the article's sentiment. The model's high score on the SST-2 (binary sentiment analysis) task qualifies it. Meanwhile, it also has a relatively prominent performance on the MRPC (English news article semantic binary comparison) task, which directly reflects its capability in encoding information in the news articles. Although this model is the heaviest among all three candidates, its capability and reliability outweigh this disadvantage.

This pretrained model needed to be fine-tuned on our downstream article classification task in order to maximize its performance. We used the well-packed *TFAutoModelForSequenceClassification* function from HuggingFace transformers to load in the RoBERTa model we selected. This ModelForSequenceClassification object class would take care of adding the classification head to the RoBERTa's main encoder layer for us. So configuring this object according to our requirement during initialization is the only setup step, and no further model structural modification took place. This basic fine-tuning structure took two inputs: tokenized input IDs and attention masks. Inside the model, there is a RoBERTa-base encoder layer (Figure 3-3: Main Layer), a dropout layer with 0.1 dropout rate, and a final dense layer with width 2 (Figure 3-3: Classifier Layer), which is the number of class in our sample dataset.




```
Layer (type)                 Output Shape              Param #
=================================================================
roberta (TFRobertaMainLayer) multiple                  124055040
_________________________________________________________________
classifier (TFRobertaClassif multiple                  595975
=================================================================
Total params: 124,651,015
Trainable params: 124,651,015
Non-trainable params: 0
_________________________________________________________________
```

Figure 3-3: BERT fine-tuning model structure and summary

In the original paper of BERT [6], BERT encoder had shown its trait of overfitting tendency. J. Devlin et al. uses only 3 epochs to fine-tune BERT on down-stream GLUE tasks. In order to expose the model with more data variations while reducing overfitting, we trained the encoder with only 1 epoch using a large set of data for the most of time when fine-tuning the encoder using the above structure. The learning rate was scheduled dynamically by Adam optimizer with initial value set to 3e-5. We employed the *linear learning rate scaling rules*[2] proposed by P. Goyal et al. [7] when adjusting batch size. Detailed discussion on the selection of hyperparameters could be found in our 2nd report.

The above fine-tuning pipeline could be seen as passing the encoded sentence embeddings to a logistic regressor to handle the final prediction work. This final predictor could also be optionally switched to other models. Two powerful classification models we considered are XGBoost and SVM. Their pros and cons are all listed in *Table 3-3*. The [CLS] token embedding, which is the embedding of the first input token of each sentence to BERT encoders, represents the embedding of the entire input document. We extracted this embedding and sent it to those models to see how the performance could be improved. We will discuss their performance in detail in a later section.

Model	Pro	Con
Logistic Regression	● Easy to implement (Model fine-tuning structure)	● Performance depends on whether data is linearly separable
XGBoost	● High performance ● Make no assumptions of the data	● Can overfit data ● Requires more computational resources ● Interpretability depending on input features
SVM	● Effective in high dimensional spaces.	● Might not converge on large dataset ● Might be difficult to find a good kernel function

Table 3-3: Comparison between models

IV. Results & Analysis

IV.I. Results Comparison

We trained RoBERTa on 8K class-balanced samples and validated it on 2K class-balanced samples for each of the 10 stratified folds on both the original and the randomly shuffled sentences, then calculated the mean and C.I. for each metric listed in the plot below over the 10 validation evaluations. *Figure 4-1* shows all of the performance statistics distribution we observed.

There are three observations and corresponding conclusions we drew on this result:

1. The classification accuracy for all tests are ~70% and above, meaning that the encoder and logistic regressor pair forms a weak learner on our task.
2. The performance on the original is better (~5%) than on the randomly shuffled sentences under the same train-test split approach, meaning that the word context is not as important as words' individual meaning to the model's decision making process.

[2] Linear learning rate scaling rule: When the minibatch size is multiplied by k, multiply the learning rate by k.

3. The performance on the Random Split dataset is significantly better than on a Split-by-Country one, meaning the countries in the same peace group might have different language features.

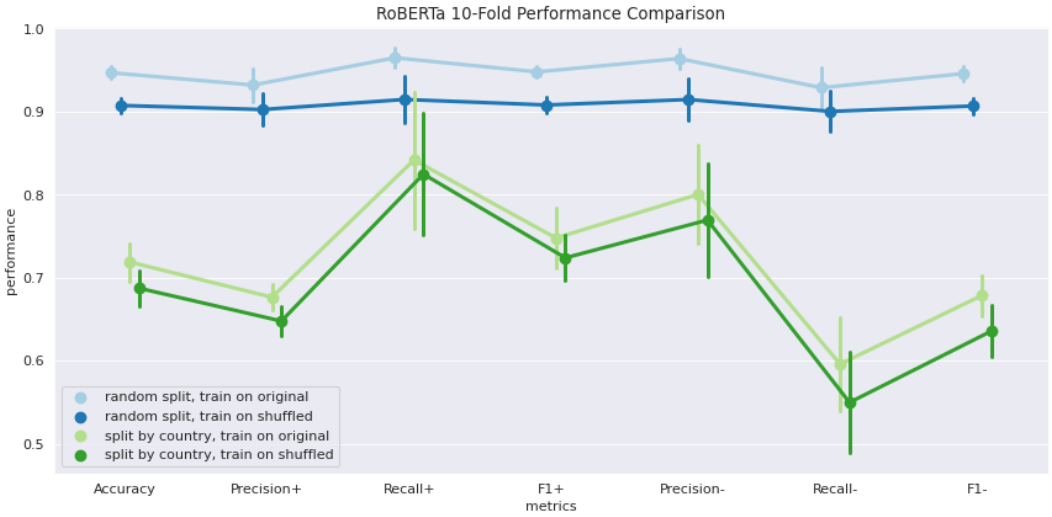


Figure 4-1: RoBERTa Fine-Tuning Performance Comparison
(+ : High Peace Class; - : Low Peace Class)

There are two hypotheses which could explain the performance drop mentioned in observation 3) above. 1) The imbalance between samples from the majorities vs. minorities in both train & validation set under a Random Split method may drive the model to just learn the language structures from the two majorities instead of all countries, and causing the Random Split model to have very good trian & validation performance due to the overwhelming proportion of majorities' data in each set, or 2) as we mentioned above, the countries in the same peace group are not sharing common language structures, and high/low peace countries could be high/low peace in different ways, thus causing the split by country model to have very bad performance when unable to see the peace indicators that are country-specific.

IV.II. Problem Elimination

To see which of the above two hypotheses is correct, we devised another test to see how the model performs on average over all countries. We took a fixed number of articles from the minorities, and measured the median of articles from each country in the minorities. Then we added into this dataset the median number of articles from each of the majorities, which are Australia and India, and performed a random train-validation split over this compounded dataset (*Figure 4-2*). As shown in *Table 4-1*, the performance of the model is roughly the same as when performing the Random Split.

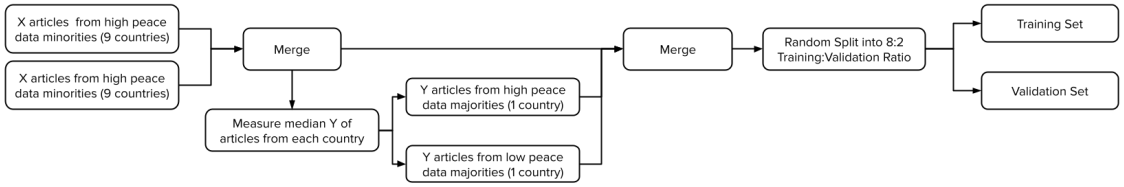


Figure 4-2: Country-balanced Random Split Illustration

		Loss (%)	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
Original Sentence	Train	17.76	93.13	93.89	96.88	95.36
	Validation	11.35	95.77	95.96	98.29	97.11
Shuffled Sentence	Train	27.49	89.08	89.78	95.93	92.75
	Validation	19.43	92.63	93.96	95.98	94.96

Table 4-1: RoBERTa Performance Over ~13.6K Training and ~3.4 K Validation
(Country-balanced Random Split)

We further shrunk the number of samples from minority countries to 10K, repeated the above procedure 10 times, and calculated the mean and C.I. for each metric to see the significance in the performance similarity. *Figure 4-2* is an illustration for the procedure. We see from *Figure 4-3* that though the performance before & after balancing the data is at about the same level, the performance scores disperse wider when including less Australian and Indian data.

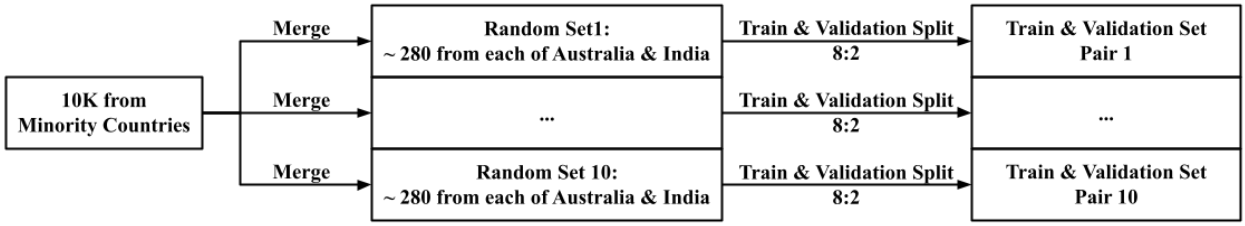


Figure 4-2: Procedure Illustration

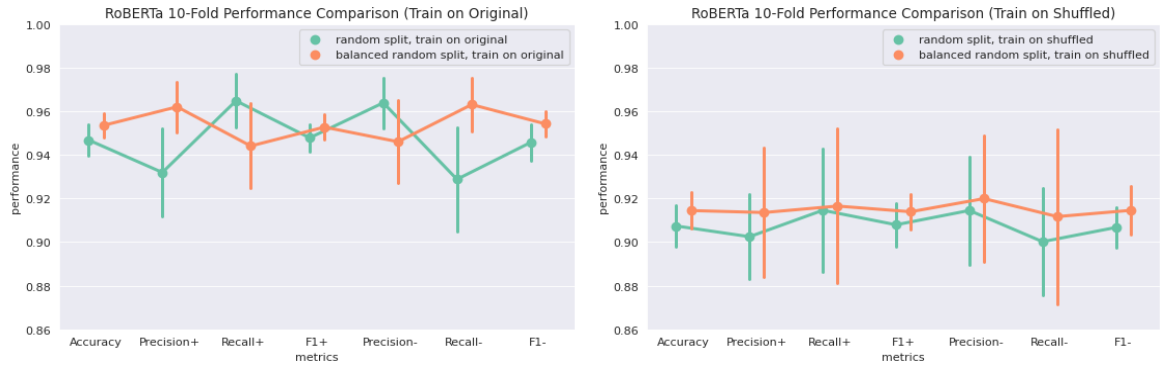


Figure 4-3: RoBERTa Fine-Tuning Performance Comparison
(+ : High Peace Class; - : Low Peace Class)

IV.II. Refine Analysis

The good performance result above indicates the data imbalance between countries is not the main cause of the performance drop. This left us with the second hypothesis to be the only cause for now. High/low peace countries could be high/low peace in different ways. An analogy is when training the image classifier to classify dogs vs. cats, if all training images of dogs are Husky, then the model

might fail to classify the image of Corgi during the validation process. Accordingly, there might be different types of language structures within each peace group.

In addition, the model's performance on shuffled sentences is again lower than the unshuffled ones by ~5%, which is consistent with our previous observation on how word ordering affects predictions. Interestingly, when splitting by country, the F1+ is significantly higher than F1- independently from whether the sentences are shuffled or not. In other words, F1- drops more than F1+ when training the model on minorities compared to those on the Random Split dataset. A possible explanation is that the high peace data, or articles from Australia, in the validation set has more shared language features with the high peace data in training, while the low peace data, or articles from India, in the validation set has less shared language features with the low peace data in training.

IV.III. Visualize Learning Results

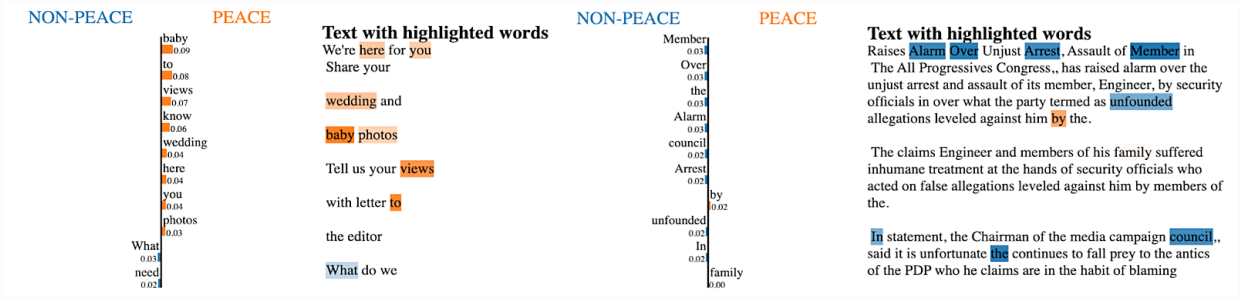


Figure 4-4: LIME Output on Training
(Left: Australia, Right: Nigeria)

We introduced the Local Interpretable Model-Agnostic Explanations (LIME) [8] technique to help visualize and explain what the RoBERTa Encoder had learnt during training. Rather than outputting the model's overall feature importance, LIME highlights the important words in one input sentence at a time using a given model's probability prediction function. We should manually take the context surrounding the words into consideration rather than interpreting the words alone by themselves. As suggested in the output of LIME (*Figure 4-4*), what RoBERTa encoder learned during training agrees with the meaning of peace indicators by human perception.

However, as one of the state-of-art technologies, there are still many limitations on what LIME could achieve (*Table 4-2*). Due to these bottlenecks of LIME, it could only be used as an auxiliary tool to provide some suggestions for us to carry out our next step plan.

Pro	Con
• Make no assumption about the model to be explained. • State-of-arts technology for explaining deep learning models.	• Difficult to quantify feature importances globally. • Need human interpretation on the output for contextual NLP models. • Output is non-deterministic. • Slow to compute.

Table 4-2: Pro and Cons of LIME

IV.IV. Side Notes

We spent some time studying the performance of other classifiers on performing our task given the [CLS] embeddings from fine-tuned BERT. For each train-validation split method, we repeated the procedure in *Figure 4-5*[3] 10 times and plot *Figure 4-6*.

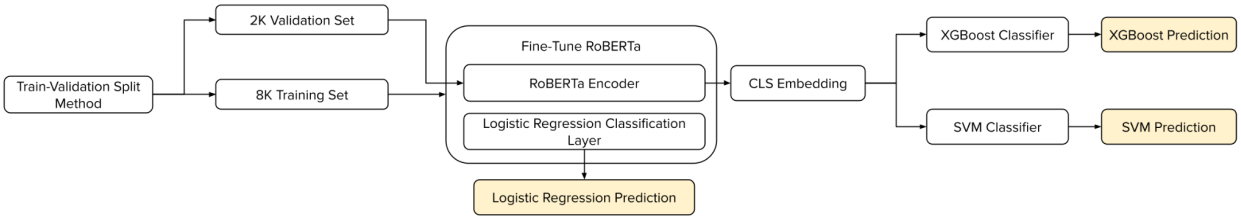


Figure 4-5: Procedure to Measure All Model Performances

From results documented in *Figure 4-6*, we obtained three observations and derived corresponding conclusions that might be helpful for future researches on this topic:

1. Lines' y-axis position per column don't differ by a lot, meaning that models' performance is "capped" by the performance at the fine-tuning stage.
2. Confidence interval at each point is narrower in the bottom two rows per column, meaning that XGBoost and SVM don't significantly improve the performance but add stability to it.
3. Validation accuracy of SVM is slightly higher than that of XGBoost (*not obvious in the plot but is supported by actual stats in the Appendix*), meaning the SVM is slightly more resistant to overfit.

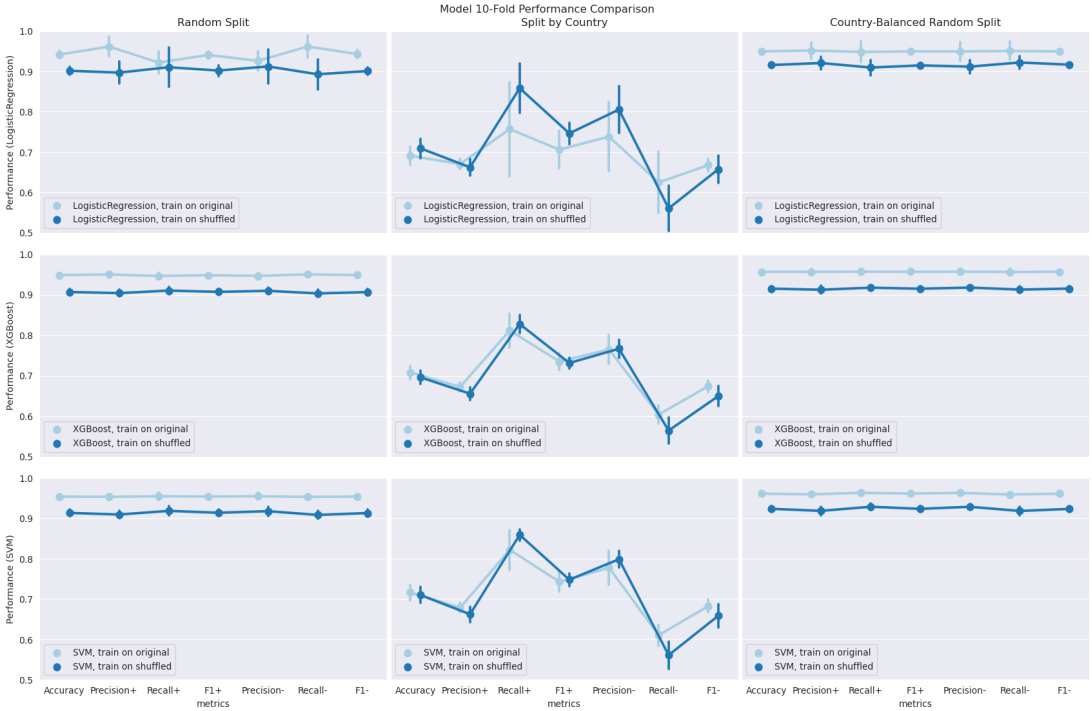


Figure 4-6: Performance Comparison of All Model Structures

[3] SVM pipeline contains a StandardScaler and a PCA to prepare the data for better fitting results.

V. Conclusion

In our project, we raised two main questions. Does the language structure of a country predict its peacefulness? Is it the context in the structure or the individual word meanings that constitutes a sentence that matters in this implication? To find out the answer, we set up three main tests to run original and shuffled sentence classification task against RoBERTa encoders with a classifier layer (mainly the logistic regressor):

1. Random Split: This method is the most traditional approach to a classification problem. The model gets to see data from all countries it is going to make predictions on. The approach ensured the class balance in training and validation set, but ignored the country-wise sample balance within each peace group in both sets.
2. Split-by-Country: Model is not able to see any data from validation country during the training process. This method challenges the model to discover shared language structures between countries within the same peace group. The class balance and country-wise sample balance in training and validation are separately maintained in this approach.
3. Country-balanced Random Split: This method is similar to the random split, but takes care of the country-wise data imbalance within each peace group in training and validation. This method helps the model to put the equal amount of focus on data from different countries rather than on those from the two data majority countries only. This approach maintained the class balance and country-wise sample balance in training and validation.

V.I. Key Findings Recap

Comparing the results we obtained from the above three tests, we get the following conclusions:

1. **Findings suggest that linguistic features are a weak predictor of whether a country is high or low peace across countries. Countries may have different linguistic features relating to their level of peacefulness.**

   We observed 95%+ accuracy on Random Split task on the original sentence and a 20% performance drop in Split-by-Country original sentence classification task, which is when the model fails to be exposed to country-specific language structure. We initially suspected the country-wise data imbalance to be one of possible causes of this drop. Because the model could fail to learn from data minorities in the Random Split, but due to the overwhelmingly high amount of data from two major countries in the validation set, such failure was not obvious until we splitted the sets by country. However, such a performance drop was not seen in the Country-balanced Random Split. This eliminated the country-wise data imbalance from the candidates pool, and left us with the only alternative cause we have at this moment: the language features that predict a country's peacefulness could vary from country to country. Such country-specific language features are strong enough that exposing the model to these features could boost its performance by 20% (*switching from Split-by-Country task to Random Split or Country-balanced Random Split task*).

2. **There exists at least a weak language structural linkage between articles from countries in the same peace group.**

We observed a 70% prediction accuracy on the Split-by-Country task when the model's performance was tested against its ability to identify common language features of a cluster of countries. Comparing this score to the 50% score baseline of a binary classification task, there is a 20% improvement on validation as a result of learning on high/low peace countries in the training, indicating the existence of the shared language features in a peace group. These features might appear in only a subset of articles from each peace group or are difficult to be captured so that the current observed improvement is small.

3. **Sentence context makes slight contributions to the peacefulness inference. The meaning of individual words is the main driver in this task.**
   We surprisingly observed an alignment of ~5% performance drop in all models and tasks when switching from predicting on original sentences to shuffled sentences, which is not significant compared to each model's overall prediction performance. Also, the key words highlighted by LIME match the meaning of peace indicator by human perception, which validates the idea that individual word meaning is more important to a model's decision function.

4. **RoBERTa is prone to overfit and sensitive to class imbalance. More complex classifiers reduce the prediction performance dispersion.**
   In all of our tests, we only fine-tune the RoBERTa with 1 epoch to reduce overfit. We mistakenly set the model to fine-tune 2 epochs with a set of class-imbalanced data, where high peace data is twice as much as low peace data in one previous experiment trial (see details in weekly progress report) and observed a performance drop of 20% on a class-balanced evaluation set when classifying shuffled sentences, and roughly 5% only for the original sentences. This reveals the proclivity for overfitting of RoBERTa encoders on class-imbalanced data, especially for hard tasks where it is more difficult to capture the patterns. Meanwhile, using more complex classifiers like XGBoost or SVM in the end seems to be able to reduce the performance dispersion.

V.II. Future Works

Our work on this project provides an overview on how well RoBERTa can capture the language structure in articles from different countries. We used many off-shelf packages and models to assist us on this task. However, these tools might not be flexible enough to align with our needs. As stated earlier, the SpaCy NER package has limitations on identifying country names that appear in some non standard format. Also, there could be cultural vocabularies such as holiday or food names that disclose the information of the article's origin. Future research on this topic might consider training an NER from scratch that better meets the requirements of this research.

Second, though the choice of encoder used in this project is made carefully via multiple comparisons, our choice of BERT encoder partially depends on the premise that peacefulness of a country is reflected sentimentally in one article. If this premise doesn't hold true, then RoBERTa-base might not be the optimal choice. As the performance of different types of encoders may vary a lot on different

tasks, we encourage the future researchers to also explore other branches of the BERT family to see if there is any chance to light-weight the model and improve performance.

Another one of the current challenges is to understand what the model truly learned. As explainable A.I being a popular and demanding topic of research in the machine learning field, there are many other state-of-art techniques we may try to explain our deep learning model. The alternative approach, as we proposed in an earlier report, is to use the models trained on Random Split and Split-by-Country tasks to classify the extreme countries selected by *matched-pair method*[4] and compare the results. The set of extreme countries selected by *matched-pair* methods are disjoint from the set of extreme countries selected by *rule-of-5*. If the model's classification results on those matched-pair countries roughly agree with the manual classification result, then it could also prove that the model has captured the linguistic key to differentiate between high and low peace countries.

VI. Contribution

- Hongling Liu: Main contributor of research and literature review on BERT. Main contributor of exploratory data analysis process and co-contributor of fine-tuning RoBERTA hyperparameters.
- Haoyue Qi: Main contributor of exploratory data analysis and optimizing EDA process. Co-contributor of text data preprocessing. Main contributor of generating prediction results under Split by Country and Country Balanced Random Split experiments.
- Xuanhao Wu: Main contributor of exploratory data analysis and visualization methods and performance metrics. Main contributor of exploring the concept and techniques of lime explainer and using it to extract the important set of decision words used by Roberta model
- Yuxin Zhou: Main contributor of preprocessing on sample dataset and researching on BERT and AWS S3\SageMaker, preprocessing runtime optimization, and running experiments on the Roberta model with different datasets.
- Wenjie Zhu: Main contributor of BERT pipeline implementation and runtime analysis, fine-tuning RoBERTA hyperparameters, and generating the train-validate results & analysis on the articles under Random Split and Split by Country experiment. Help setting up weekly meetings and managing progress.

[4] Cluster the countries using geography first, then select the highest and lowest peace countries in each geographical region, where the lowest peace country in a region is the one with the lowest number of indices classified as high-peace and highest number of indices classified as low-peace. Same logic follows for high-peace.

References

[1] Coleman, P. T., Fisher, J., Fry, D. P., Liebovitch, L. S., Chen-Carrel, A., & Souillac, G. (2020). How to live in peace? Mapping the science of sustaining peace: A progress report. The American psychologist, 10.1037/amp0000745. Advance online publication. https://doi.org/10.1037/amp0000745

[2] "Smart Batching Tutorial - Speed Up BERT Training · Chris McCormick," *Mccormickml.com*, Jul. 29, 2020. https://mccormickml.com/2020/07/29/smart-batching-tutorial/.

[3] E. Alzahrani and L. Jololian, "How Different Text-preprocessing Techniques Using The BERT Model Affect The Gender Profiling of Authors," *arXiv.org*, 2021. https://arxiv.org/abs/2109.13890.

[4] J. Camacho-Collados and M. T. Pilehvar, "On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis," *arXiv.org*, 2017. https://arxiv.org/abs/1707.01780.

[5] Cathal Horan, "Tokenizers: How machines read," *FloydHub Blog*, Jan. 28, 2020. https://blog.floydhub.com/tokenization-nlp/#wordpiece.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv.org*, 2018. https://arxiv.org/abs/1810.04805.

[7] P. Goyal *et al.*, "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour," *arXiv.org*, 2017. https://arxiv.org/abs/1706.02677.

[8] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," *arXiv.org*, 2016. https://arxiv.org/abs/1602.04938.

Appendix

		Accuracy	Precision +	Recall +	F1 +	Precision -	Recall -	F1 -
Original Sentence	Random Split	94.68 (94.13, 95.22)	93.19 (91.70, 94.69)	96.48 (95.56, 97.40)	94.78 (94.31, 95.25)	96.39 (95.53, 97.25)	92.88 (91.10, 94.65)	94.57 (93.94, 95.20)
	Split By Country	71.88 (70.17, 73.58)	67.64 (66.49, 68.79)	84.17 (78.04, 90.29)	74.77 (72.12, 77.43)	80.02 (75.59, 84.45)	59.59 (55.42, 63.76)	67.87 (66.04, 69.69)
	Country Balanced Random Split	95.35 (94.94, 95.77)	96.20 (95.34, 97.07)	94.40 (92.97, 95.85)	95.27 (94.85, 95.70)	94.60 (93.19, 96.01)	96.30 (95.40, 97.21)	95.42 (95.00, 95.84)
Shuffled Sentence	Random Split	90.74 (90.05, 91.43)	90.25 (88.80, 91.70)	91.46 (89.34, 93.58)	90.80 (90.06, 91.53)	91.45 (89.60, 93.30)	90.02 (88.19, 91.84)	90.67 (89.97, 91.37)
	Split By Country	68.72 (67.16, 70.28)	64.77 (63.41, 66.14)	82.47 (77.04, 87.90)	72.38 (70.35, 74.42)	76.94 (71.86, 82.02)	54.97 (50.43, 59.51)	63.57 (61.22, 65.93)
	Country Balanced Random Split	91.45 (90.83, 92.07)	91.36 (89.15, 93.57)	91.65 (89.02, 94.29)	91.39 (90.79, 92.00)	92.00 (89.86, 94.13)	91.17 (88.20, 94.14)	91.46 (90.63, 92.29)

Table 1: RoBERTa 10-Fold Performance Comparison Detailed Numbers
(+ : High Peace Class; - : Low Peace Class)

		Accuracy	Precision +	Recall +	F1 +	Precision -	Recall -	F1 -
Original Sentence	Logistic Regression	94.19 (93.50, 94.87)	96.13 (94.34, 97.93)	92.20 (90.16, 94.24)	94.06 (93.36, 94.77)	92.61 (90.84, 94.38)	96.17 (94.13, 98.21)	94.29 (93.60, 94.98)
	XGBoost	94.86 (94.59, 95.13)	95.03 (94.66, 95.41)	94.67 (94.09, 95.25)	94.85 (94.57, 95.13)	94.70 (04.17, 95.23)	95.05 (94.65, 95.45)	94.87 (94.61, 95.14)
	SVM	95.35 (94.94, 95.77)	95.29 (94.80, 95.78)	95.43 (94.76, 96.10)	95.36 (94.94, 95.78)	95.43 (94.80, 96.06)	95.28 (94.77, 95.79)	95.35 (94.95, 95.76)
Shuffled Sentence	Logistic Regression	90.15 (89.38, 90.93)	89.68 (87.65, 91.72)	91.03 (87.41, 94.65)	90.20 (89.20, 91.20)	91.22 (88.13, 94.31)	89.28 (86.53, 92.03)	90.08 (89.42, 90.74)
	XGBoost	90.70 (90.12, 91.28)	90.42 (89.79, 91.05)	91.05 (90.40, 91.70)	90.73 (90.16, 91.31)	90..99 (90.35, 91.62)	90.35 (89.69, 91.01)	90.67 (90.08, 91.25)
	SVM	91.30 (90.67, 91.92)	90.90 (90.21, 91.58)	91.80 (90.92, 92.68)	91.34 (90.71, 91.98)	91.73 (90.89, 92.57)	90.80 (90.07, 91.53)	91.26 (90.63, 91.88)

Table 2: All Model 10-Fold Performance Comparison Detailed Numbers for **Random Split** Approach
(+ : High Peace Class; - : Low Peace Class)

		Accuracy	Precision +	Recall +	F1 +	Precision -	Recall -	F1 -
Original Sentence	Logistic Regression	69.07 (67.42, 70.72)	67.01 (66.07, 67.96)	75.69 (67.00, 84.38)	70.58 (67.15, 74.02)	73.82 (67.53, 80.11)	62.45 (56.81, 68.10)	66.75 (65.58, 67.91)
	XGBoost	70.81 (69.56, 72.05)	67.22 (66.47, 67.98)	81.19 (78.07, 84.31)	73.51 (71.96, 75.06)	76.49 (73.89, 79.09)	60.42 (58.81, 62.03)	67.43 (66.37, 68.48)
	SVM	71.56 (70.16, 72.96)	67.79 (66.95, 68.64)	82.14 (78.44, 85.84)	74.22 (72.44, 76.00)	77.70 (74.53, 80.86)	60.98 (59.06, 62.91)	68.20 (67.05, 69.35)
Shuffled Sentence	Logistic Regression	70.93 (69.13, 72.72)	66.21 (64.65, 67.76)	85.83 (81.25, 90.41)	74.63 (72.68, 76.57)	80.54 (76.28, 84.80)	56.02 (51.86, 60.19)	65.71 (63.16, 68.27)
	XGBoost	69.61 (68.39, 70.83)	65.56 (64.40, 66.72)	82.78 (81.16, 84.40)	73.15 (72.15, 74.14)	76.67 (75.05, 78.29)	56.44 (54.05, 58.83)	64.96 (63.15, 66.76)
	SVM	70.94 (69.43, 72.45)	66.17 (64.76, 67.58)	85.86 (84.82, 86.90)	74.73 (73.60, 75.85)	79.81 (78.26, 81.37)	56.02 (53.46, 58.58)	65.80 (63.64, 67.95)

Table 3: All Model 10-Fold Performance Comparison Detailed Numbers under **Split-by Country** Approach
(+ : High Peace Class; - : Low Peace Class)

	Accuracy	Precision +	Recall +	F1 +	Precision -	Recall -	F1 -

Original Sentence	Logistic Regression	94.95 (94.56, 95.34)	95.17 (93.67, 96.66)	94.83 (92.90, 96.75)	94.94 (94.52, 95.37)	94.94 (93.23, 96.66)	95.06 (93.34, 96.77)	94.95 (94.55, 95.35)
	XGBoost	95.69 (95.49, 95.89)	95.68 (95.15, 96.20)	95.74 (95.24, 96.23)	95.70 (95.53, 95.87)	95.72 (95.21, 96.23)	95.65 (95.09, 96.22)	95.68 (95.44, 95.92)
	SVM	96.08 (95.90, 96.26)	95.90 (95.55, 96.25)	96.29 (95.92, 96.66)	96.09 (95.93, 96.26)	96.26 (95.85, 96.67)	95.87 (95.53, 96.21)	96.07 (95.87, 96.26)
Shuffled Sentence	Logistic Regression	91.57 (91.14, 92.01)	92.07 (90.95, 93.18)	90.95 (89.53, 92.37)	91.48 (90.94, 92.01)	91.17 (89.95, 92.40)	92.20 (91.02, 93.39)	91.66 (91.29, 92.03)
	XGBoost	91.53 (91.10, 91.95)	91.26 (90.51, 92.00)	91.76 (91.45, 92.08)	91.51 (91.02, 92.00)	91.79 (91.55, 92.03)	91.30 (90.65, 91.94)	91.54 (91.15, 91.93)
	SVM	92.31 (91.81, 92.81)	91.81 (91.03, 92.60)	92.82 (92.19, 93.45)	92.31 (91.77, 92.85)	92.82 (92.31, 93.32)	91.79 (90.98, 92.59)	92.30 (91.82, 92.77)

Table 4: All Model 10-Fold Performance Comparison Detailed Numbers for **Country-balanced Random Split** Approach (+ : High Peace Class; - : Low Peace Class)