Advance Machine Learning Final Project
Cristina Mac Gregor
Gonzalo Pons
Darshan Sumant

Systematizing news reports on conflict incidences

This report presents the results of the analysis conducted during the quarter applying neural network techniques to analyze text data related to conflict. It will cover:

1. The data
    a. The ACLED project
    b. Main variables
    c. Descriptive statistics
    d. Core pre-processing
2. Literature review
3. The objectives
    a. Related literature (as covered in our initial proposal)
    b. The analysis
    c. Caveats
4. Additional considerations to be addressed through future work.

I. Data

I.I ACLED

The armed conflict location events dataset (ACLED) is a project that has as an objective to disaggregate main conflicts, as are usually conceptualized by the literature on conflict analysis, into specific instances of the confrontations between the involved parties. To create and update this dataset, analysts first collect information from secondary sources, such as news articles or reports. This information is then coded by a first reviewer. Two more reviewers check this information to ensure quality. ACLED provides detailed guidelines for each of these steps.

Each unit of observation corresponds to a confrontation that can be catalogued under one of four main categories: battles, violence against civilians, riots/protests, or remote violence, and there is a lot of information regarding each recorded conflict. Some of the main variables included are:

- *Latitude and Longitude*
- *Administrative Units in which the event took place*
- *Region*
- *Type of event (four broad categories)*
- *Sub-type of event*
- *Actor1: One of the actors involved in the recorded conflict*
- *Actor2: The other party to the conflict. It is important to note that Actor1 and Actor2 are not necessarily in a determinate order and who is in which variable does not correspond to who initiated the attack.*

- *Interaction - types of actors interacting in the event*
- *Number of fatalities*
- *Notes: This is the section of our data that consists of text. It includes a snippet of text that captures some details on the event recorded. Initially we thought that this variable included the entirety of the text from the news report that first covered the reported incident, but it is actually a brief explanatory description made by the people putting together the dataset. The challenge of this is that sometimes there is information captured by the other variables that does not necessarily come from the text, so not all the information that we are going to try to predict is perfectly encompassed by the strings of text that we will process.*

Some examples of how the text looks like are the following:

[177] *'Troops have shot dead a commander of Algerias most radical guerrilla faction who had been sought for scores of killings, a pro-government newspaper said. LAuthentique said Hamou Eulmi, known by his nom de guerre Zinedine, was killed Tuesday near a mos'*

[192] *'25 March 1999 The Globe and Mail Two girls aged 2 and 3 and a woman were among nine victims who had their throats cut overnight by suspected Islamic extremists at an isolated farm south of Algiers, security services said yesterday.'*

[1123] *'One militant was also killed Thursday in a clash with security forces in Beni Beshir, close to Skikda.'*

[19293] *'Violent fight between Seleka rebels and rioters from the Gobongo in Bangui'*

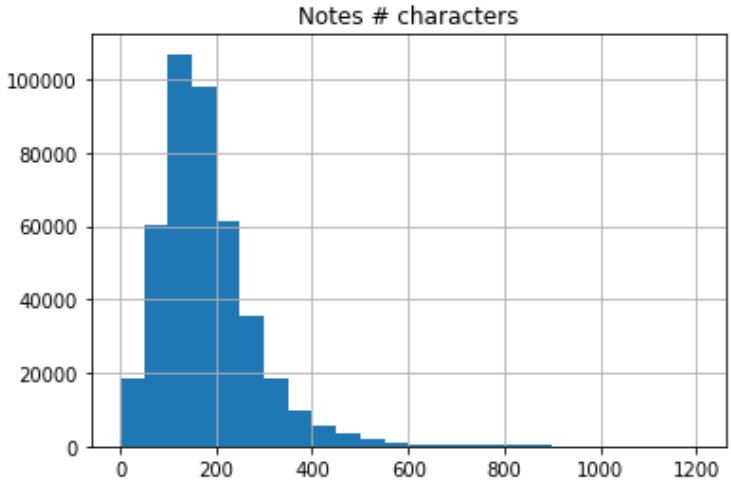
I.II Descriptive

Other relevant information about the data is the geographical and time scope of it. The data reported does not represent all regions for all periods of time. Information about conflict in Africa is reported since 1998; information about Asia approximately since 2005 (depending on the sub-region), information about the Middle East since 2010, and information about Europe, since 2018. It is also important to note that for each of these periods there is a spike on the number of conflicts reported by the dataset overall, which does not necessarily mean that in prior years this conflict did not exist, but that increased coverage and the expansion of regions covered by the project combine to create this increment.

I.III Core pre-processing

⅕ of the observations had text columns that had less than 100 characters. In order to preserve only text with more content to it, for all the supervised analysis conducted, we kept only the observations with more than 100 characters of text.

Figure 1: Distribution of the number of characters per notes



II. Literature review

We found a series of articles that worked with *name entity recognition problem, topic modelling* and *relation extraction*. Even though we didn't find papers that solve the exact same problems, we focus here in three papers that solve a similar problem in comparable contexts.

1. ***Source 1: Identifying civilians killed by police with distantly supervised entity-event extraction. Katherine A. Keith, Abram Handler, Michael Pinkham, Cara Magliozzi, Joshua McDuffie, and Brendan O'Connor. Proceedings of EMNLP 2017.***

Abstract: The authors aim to extract names of persons who have been killed by the police from news text, using EM-based distant supervision with logistic regression and convolutional neural network classifiers. They describe this task as combining *event extraction* and *relation extraction,* by describing events, their arguments, and the semantic relation between entities.

Data: The news documents were collected from continually querying Google News throughout 2016 with using keywords related to police and fatality. This searches were restricted mostly to the US.

Pre-processing: They use spaCy NLP package to segment sentences and extract entity mentions. This included splitting the data in to sentences, removing duplicates of sentences, removing sentences with less than 5 tokens or more than 200, among other data cleansing tasks. Regarding entity mentions, they extract mentions identified as "persons" by spaCy's named entity recognizer and that have a (first name, last name) pair as analyzed by the HAPNIS rule-based name parser. To prepare sentences for modeling, candidate mention spans are collapsed to a special "Target" symbol, while other names are mapped to a different "Person" symbol. Additionally, to improve precision, the data was filtered to include only sentences with both a police and fatality keyword.

Learning models: both feature-based logistic regression and convolutional neural networks combined with both “hard” distant label training and “soft” Expected Maximization (EM) joint training. For the CNN approach a stochastic gradient descent for the negative expected log-likelihood is performed using pre-trained word embeddings for initialization.

Metrics: Area under precision recall curve (AUPRC) and F1 scores.

Evaluation of results: The models predict entity-level labels and the authors compare these with a dataset of Fatal Encounters which has more than 18,000 entries of victims’ name, age, gender, race, in addition to location, cause and date of death. This comparison is done with two *distant supervision training* paradigmas (“hard” and “soft”).

2. ***Source 2: Learning to Extract International Relations from Political Context. Brendan O'Connor, Brandon M. Stewart, and Noah A. Smith. Proceedings of ACL 2013.***

Abstract: The authors present an unsupervised approach to event extraction, through a probabilistic model that infers latent frames and a representation of the relationship between political actor pairs. It aims to produce tuples with the form (*source (actor)*, *receiver (actor)*, *timestamp*, *predicate path (action)*)

Data: 6.5 million newswire articles from the English Gigaword 4th edition (1994–2008, Parker et al., 2009), and a sample from the New York Times Annotated Corpus.

Pre-processing: syntactic pre-processing. CoreNLP is used for POS-tag and parsing the articles. Name Entity Recognition (NER) is done deterministically by finding instances of country names from the CountryInfo.txt dictionary from TABARI. To identify the verbs and their direction, they use the “CCprocessed” version of the Stanford Dependencies (de Marneffe and Manning, 2008). Verb paths are identified by looking at the shortest dependency path between two mentions in a sentence.

Learning models: Logistic normal topic model, including latent temporal smoothing on the political context prior.

Metrics & Evaluation of results: The authors compare the automatically learned verb classes to pre-existing verb patterns from TABARI (open-source rule-based event extraction system), and demonstrate correlation to the Militarized Interstate Dispute (MID) dataset. The metric they use is the Area Under the Curve metric.

3. ***Source 3: Content-driven, unsupervised clustering of news articles through multiscale graph partitioning. M. Tarik Altuncu, Sophia N. Yaliraki, Mauricio and Barahona. Data Science, Journalism & Media workshop 2018.***

Abstract: The authors showcase an unsupervised approach to content-clustering, at different levels of resolution, by using a recent deep neural network text analysis methodology (Doc2vec) that represents text in a vector form and then applies a multi-scale community detection method (Markov Stability) to partition a similarity graph of document vectors.

Data: The authors use a corpus of 9,021 news articles published by Vox Media during the whole of 2016. The articles correspond to a wide range of topics from politics, to sport, to human interest, with a clear focus on the US. News about the 2016 US Presidential election, Brexit, and the Rio Olympics feature heavily in the corpus.

Separately, the base Doc2vec model is trained using a corpus of 5.4 million Wikipedia articles. This model trained on the Wikipedia corpus is then applied to the Vox Media corpus.

Pre-processing: Tokenization, with removal of stop words, and normalization using NLTK module stemming methods. Meaningless wrapper sentences such as document header, footer, signatures, annotations, etc. were removed with a threshold of 2+ frequency on the sentence tokens.

Learning models: The base Doc2vec model is trained using a corpus of 5.4 million articles from Wikipedia. This is done to ensure a good representation of the general text encountered in news articles. The model thus trained, is then applied to the corpus of 9021 Vox Media articles from 2016 (after the pre-processing to tokenize, clean wrappers, etc.) to create a 300-dimensional paragraph vector for each of the 9021 articles. This paragraph vector encapsulates a variety of semantic & syntactic characteristics.

Pairwise cosine similarity between all possible pairs (${}^{9021}C_2$) of articles is obtained, and for the entire dataset, a min spanning tree (MST) is generated through the MST-kNN method with k = 13 (nodes are articles, and the weighted edges represent similarities between the nodes) which is called the similarity graph $\boldsymbol{G_S}$. a Markov Stability (MS) is applied to $\boldsymbol{G_S}$ in order to extract the multi-scale community structure intrinsic to the graph. MS defines hard clusters based on partition extracted from the similarity graph.

Metrics & Evaluation of results: The quality of clusters generated is evaluated using two alternative approaches - (1) Intrinsic Topic Coherence is measured using the pointwise mutual information (PMI) based on the co-occurrence of the 15 most common words. (2) To compare similarity between partitions of the similarity graph, the authors use normalized mutual information (NMI) based on mutual information between, and the individual entropies of the two cluster assignments.

III. Objectives

We started this project with three main goals that evolved along the way as we delved deeper into the data and identified some limitations to our original proposal. Nonetheless, despite the challenges and changes, our work continued to be structured around three main questions:

1. Can we identify who are the two main actors involved in an instance of a conflict?
2. Can we identify what is the relationship (mainly, the type of violence used) by one actor on another?
3. Can we explore additional patterns in the events reported according to the text available?

In the following section we will explore with detail each of the goals, outlining the initial goal was, what the final question asked ended up being, the methodology and our results, and furthermore, some considerations about what we found and potential for future work.

III. I Goal 1

Initially, this section of our project aimed at identifying which were the specific contending parts in a conflict (e.g. Syria's Military, Boko Haram, village) given the snippet of text we are given. For this, we planned to use Name Entity Recognition (NER) packages, such as SpaCy or Stanford NER, to identify relevant organizations in the text and use this as input to our neural network. Additionally, we wanted to take advantage of the labeled variables "ACTOR1" and "ACTOR2" of our dataset (see Table 1), which could provide us a way to evaluate and train our models.

Table 1: Sample of relevant variables.

ACTOR1	INTER1	ACTOR2	INTER2	INTERACTION	NOTES
GIA: Armed Islamic Group	Rebel Groups (2)	Military Forces of Algeria (1999-)	State Forces (1)	12	26th Feb 2001- BBC Mon-Large military offensive all over the country sees 9 soldiers and 6 GIA killed
Unidentified Armed Group (Algeria)	Political Militias (3)	Civilians (Algeria)	Civilians (7)	37	A 40-year-old repentant who answered to the name of Hamid Doghman was assassinated this past Tuesday 4 March at about 2000 hours not far from downtown Zemmouri 16 kilometres east of Boumerdes by a militant group made up of between four and six elements. The attack targeted this ex-Islamist who had signed his repentance in August 1995.

However, we encountered significant challenges with this approach. First, the off-the-shelf packages were not always tagging the organizations accurately, sometimes missing organizations or relevant actors. Second, as we can see from the first row of Table 1, the labeling

process of ACTOR variables sometimes incorporated information that wasn't available in the notes. In this case, for example, there is no indication of the Military Forces of Algeria other than the word "soldiers". Finally, there was an inherent complexity of predicting two variables (ACTOR1 and ACTOR2) that incorporated exactly the same information and categories; whether GIA and Military is ACTOR1 or ACTOR2 is completely arbitrary.

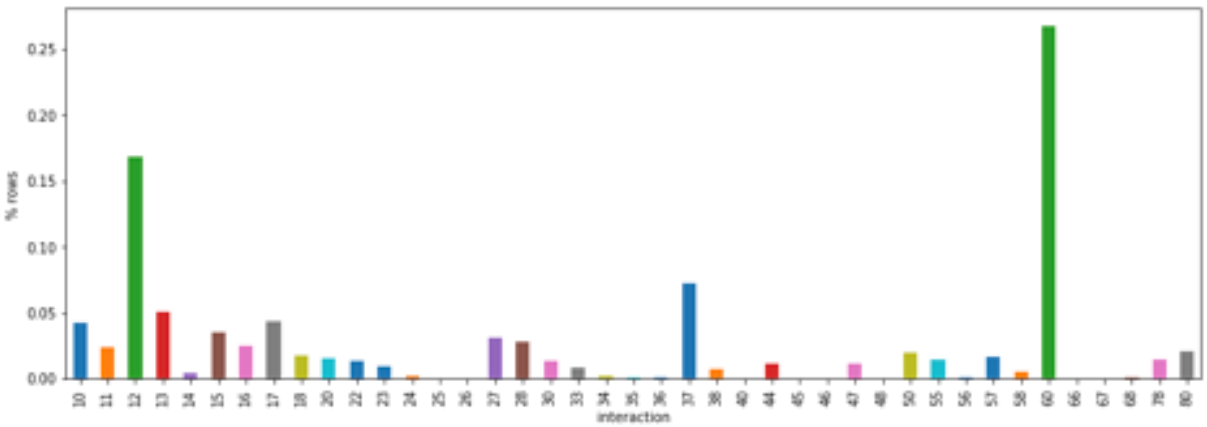
In light of these challenges, we decided to take a similar, but simpler approach to identify actors. In this case, we decided to predict instead the variable "INTERACTION", which categorized each conflict event as an interaction between two types of actors. Hence, instead of trying to identify GIA and Algeria Military Forces separately, we would identify if the specific NOTE was a Rebel Group vs State Force. INTERACTION code is created by concatenating the codes of each INTER actor. For example, here State Forces have code 1 and Rebel Groups code 2, resulting in INTERACTION code 12. Appendix 1 shows the 41 different codes of INTERACTION. As mentioned, the order of ACTOR1 or ACTOR2 is arbitrary, and so code 12 and 21 would mean the same, and therefore they preserve only the lowest of the two (i.e. there is no code 21, only code 12).

Additionally, instead of using NER for our input, we rely on pre-trained word embeddings as input for our neural network. Specifically, we use medium-sized SpaCy word2vec embeddings pre-trained in Wikipedia ('en_core_web_md'). This function provides vectors of 300 embedding dimensions for complete sentences.

Pre-processing

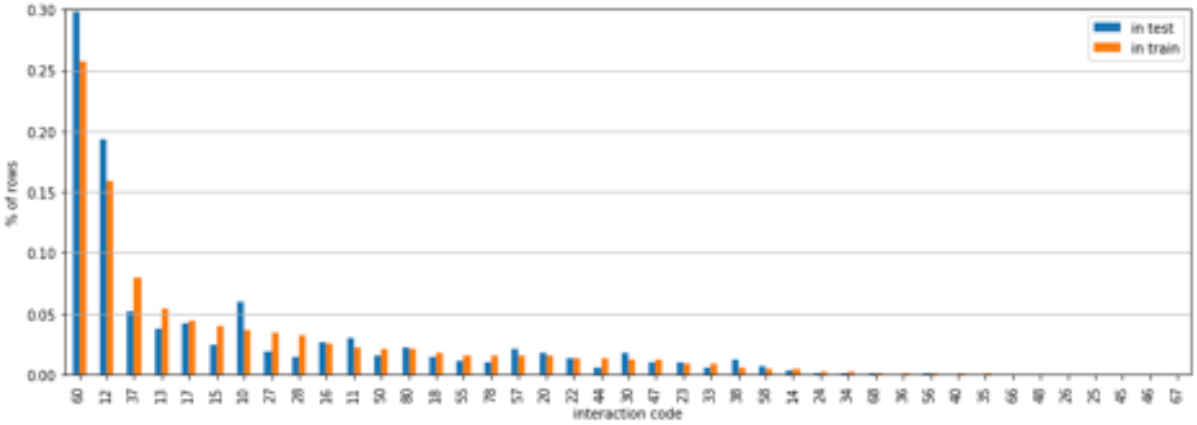
We first excluded notes of less than 100 characters long to ensure at least a meaningful amount of text data. Additionally, we excluded three strange INTERACTION codes that were not included in the ACLED codebook, which likely means that it was either very recent, rare, or simply an error. These were 88, 70 or 77. Following the INTER codes, this would mean "Other Actor vs Other Actor", "Sole Civilian Action" and "Civilian vs Civilian", respectively. This resulted in 417,003 events or rows. Figure 2 shows that there are large imbalances of codes like "60- SOLE PROTESTER ACTION", which represents around 26% of the events, and "12- MILITARY VERSUS REBELS", with around 17%. There are also some codes like "25- REBELS VERSUS RIOTERS" and "26- REBELS VERSUS PROTESTERS" that have less than 1%.

Figure 2. % of INTERACTION codes in analytical dataset



The next step was to split the dataset into training, validation and test sets. We used 75% of the events for training (312,752), 12.5% for validation (52,126), and 12.5% for test (52,126). Following other projects in the literature that work with machine learning and news, we took the oldest events for training and the most recent for validation/test. This better resembles the practical applications of these algorithms, which normally are used to categorize new events, incorporating a challenge of languages and text modifying over time. The remaining 25% was randomly split between validation and test to ensure the validation statistics would be representative of the final test ones. Figure 3 shows the distribution of INTERACTION codes between training and test. While there are some differences, these should not influence much.

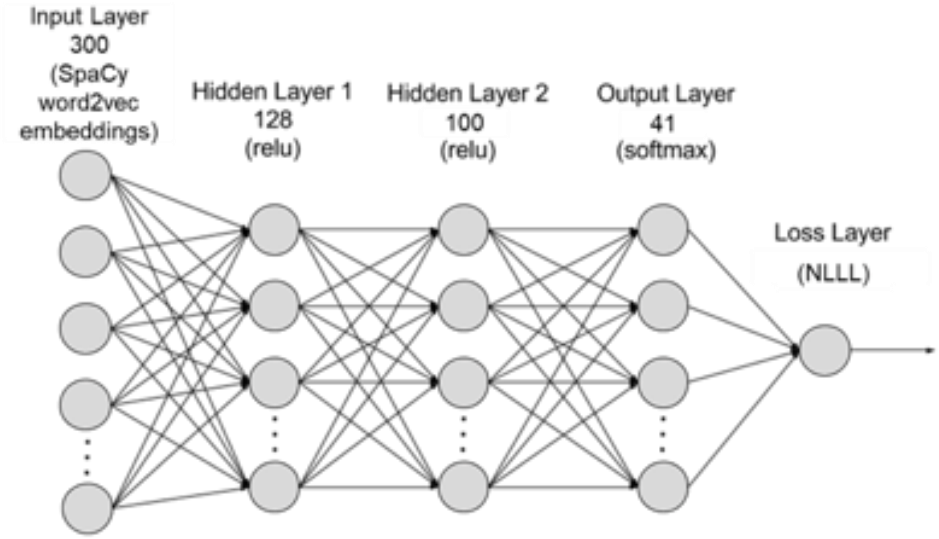
Figure 3 % of INTERACTION codes in training and test



Simple linear logistic regression & 2 hidden layer neural network

We ran a neural network (See Figure 4) that receives as input vectors of SpaCy pre-trained word2vec embeddings with 300 dimensions of each sentence. This passes through two hidden layers of 128 and 100 nodes and with ReLU transformations. The output layer of 41 categories is then transformed with softmax to get probabilities. Then, the result is applied to NLLL loss function. We use Adam optimizer to update the parameters (note that the embeddings are not parameters trained in this process). We established a learning rate of 0.001, although the Adam optimizer modifies this rate throughout the process.

Figure 4. Neural network architecture.



(Adapted from tutorial image)

For training, we go through the entire training dataset twice (epoch = 2). When creating the training dataset, observations were randomly shuffled so the model would not be trained chronologically and the loss would be more informative. Every 5,000 rows the total loss was saved and reset. Every 80,000 rows and at the end of the training dataset (iteration 312,752), the accuracy in the validation was calculated (8 times total). Off these 8 validation accuracy calculations, the best model was saved to prevent overfitting. This best model was used then in the test set to calculate an unbiased estimate of accuracy and analyze the performance of the model.

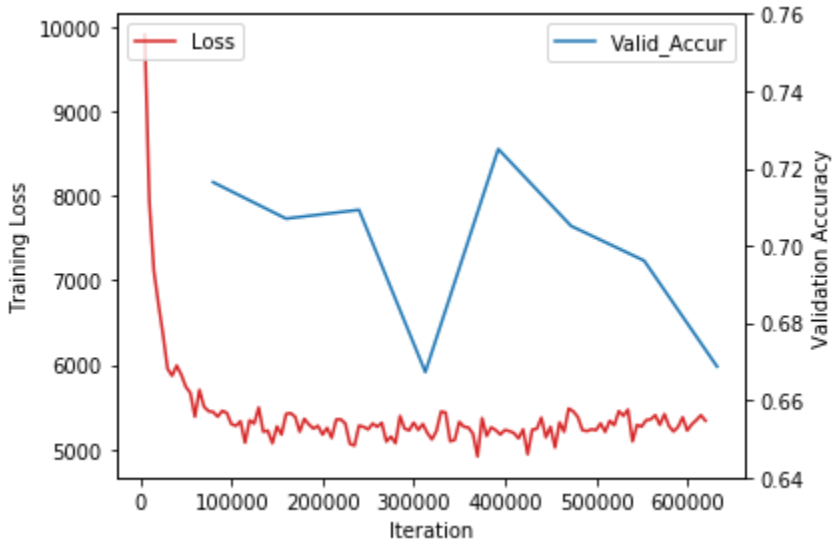
While we did not do a formal tuning of the hyperparameters by running the complete training process with different hyperparameters, we did try larger learning rates and only one epoch, showing that learning was too volatile with this rate and additional training was useful.

Additionally, we performed a baseline model to compare accuracy. We used a simple linear regression neural network that inputs the same vector of 300 pre-trained dimensions and outputs 41 categories, transformed in softmax and passed through the same loss function and optimization.

Results

Figure 5 shows the training loss and validation accuracy in the training process. As we can see, there is a sharp decrease in training loss very early in the process, and then stabilizes but still moving erratically. In terms of accuracy in the validation, the best model is calculated at iteration 392,752 (second calculation of the second epoch), achieving an accuracy of 72.5%. After that, we see how the accuracy starts a steady decline.

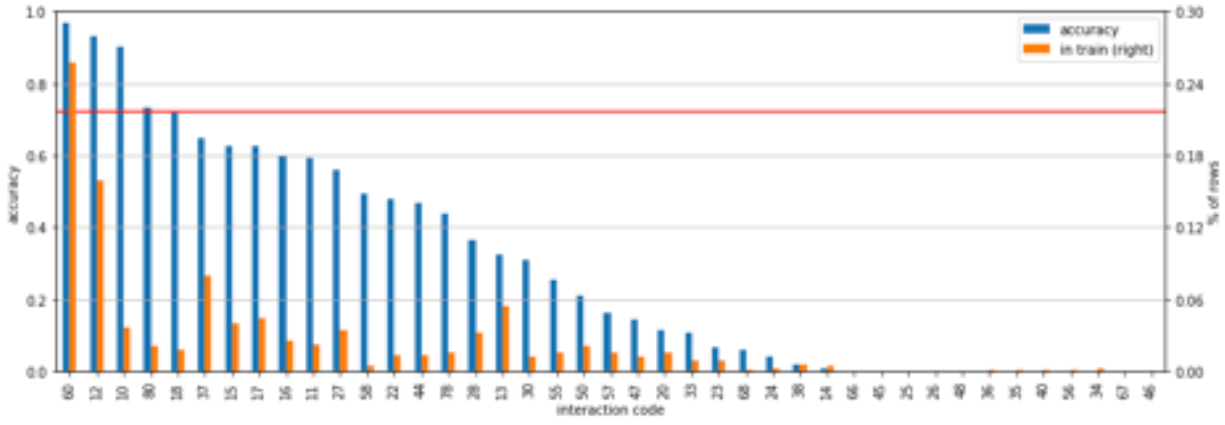
Figure 5. Training loss and validation accuracy.



Using the model with the best performance in validation, we achieved an accuracy of 72.3% in the test set. On the contrary, the simple linear model achieved 50.9% accuracy, evidencing that the additional complexity in our architecture was more successful in classifying 41 categories.

Figure 6 shows the accuracy rate by INTERACTION code. We can see that, although the accuracy was high, this is mainly driven by a few relevant codes. The top 5 codes by accuracy were “60- SOLE PROTESTER ACTION”, “12- MILITARY VERSUS REBELS”, “10- SOLE MILITARY ACTION”, “80- SOLE OTHER ACTION”, “18- MILITARY VERSUS OTHER”. Only 11 codes have accuracy above 50%, which evidence that this model does not perform as well as the total accuracy suggests.

Figure 6. Accuracy and percentage of rows by INTERACTION code.

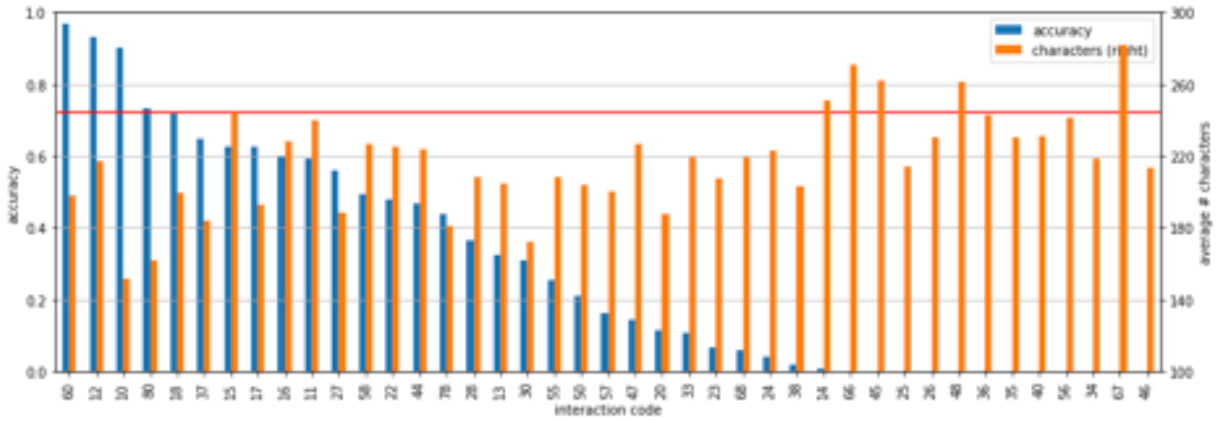


The orange bar in the same figure shows the relative frequency of available training events. It is evident that having an important number of events for training has an effect on the final accuracy, since INTERACTION codes with less than 1% have accuracy almost zero, and the two most accurate codes are also the ones with most events. However, some codes like 80 or 18 show us that its not only about number of training events. Appendix YY shows an example of each of the

top 5 accuracy codes. Our hypothesis is that, in addition to having many observations to train on, there are certain topics and words that make it easier to categorize some INTERACTIONS. For example, code 60 most likely requires the word “protestors” or “protest” to appear. Codes 80 and 18, which incorporates “other actors”, usually involve “coalitions” and are mostly events in the Yemen war. “10 - Sole Military Actions” are usually events were militaries “established bases”. On the contrary, “13- MILITARY VERSUS POLITICAL MILITIA” likely have a variety of topics and different specific actors involved, making it more difficult to predict, even when they have several observations.

The extension of the text could be another important parameter that influences accuracy. Figure 7 shows again the accuracy of each INTERACTION code, but with the average number of characters of the texts in the test data. While the average is influenced as well by the number of available events in the training, we can still see clearly that longer texts do not necessarily contribute to more accuracy.

Figure 7. Accuracy and average # of characters by INTERACTION code.



To better differentiate the effect of the extension of the text, we ran an ordinal least square (OLS) regression with the outcome being a dummy variable of whether the prediction was correct, and the independent variables being dummies of each code (base code “10- SOLE MILITARY ACTION”), a dummy for 2019 (base is 2018) and the number of characters of the string. A detailed output can be found in Appendix 3, showing that all variables were statistically significant. The regression shows that each additional character in the text *decreases* accuracy in 0.03 percentage points (pp). Additionally, it shows that the event being registered in 2019 decreased accuracy in 1.55 pp compared to 2018.

An important caveat from this analysis is that the labeling wasn’t done as consistently as we thought. For example, we found some instances were military forces of a country were categorized as “1 – State Forces” sometimes, and “8 - External/Other Forces” in others. This could be product of a tension in defining the legitimacy of some governments or the links of some coalitions to certain countries. Regardless, this likely affected our results. Additionally, some NOTES were not as detailed as we would like, and some even included comments from the coders. This certainly affects this and the following objectives.

III.II Goal 2. The relationship between the actors

Given the nature of our project, to identify the relationship between the two contesting actors, is to identify the attack or harm inflicted by one side on another, or mutual aggressions. In our proposal, we mentioned using *relation extraction* methods for categorizing the different ways in which Actor 1 and Actor 2 interact, as well as the direction of the interaction. We wanted to find whether A attacks B, or B attacks A.

We ran into two main challenges for carrying out this goal, parting from the fact that *we do not have labelled data* on the relationship between the actors.

1. We know broad categories for the type of event, as described in the sub-event-type category variable. These categories are descriptions such as “drone strike”, “agreement”, “attack”, “chemical weapon”, “grenade” and “sexual violence”. However, some of this categories, such as *attack,* do not give us precise information regarding the actions carried out by the perpetrating actor. We wanted to try to find the specific actions done by the actors.
2. Moreover, we don’t know the directionality of the attacks. Part of ACLED’s methodology, given that many of the events are part of more intricate conflicts in which actors can be both victims and perpetrators, there is no distinction as such of the parties participating in the events. They are mainly described as *Actor1* and *Actor2,* and for most cases we don’t know who is the aggressor and who is the victim, which proves to be a limitation in labelling the data for the relationship extraction.

To overcome this limitations, we proceeded to perform a semi-automated labelling process followed by a series of preprocessing steps before performing a multi-classification procedure through a Recurrent Neural Network. The process followed was:

Pre-processing

1. Subset our data to cases where we know who the aggressor is: Which in our case means all the attacks in which protestors or civilians were involved, because this are actors that are consistently victims and would not be categorized as that if they were the ones initiating the action (they’d be categorized as Unidentified Armed Groups or Rioters. We only maintain the following INTERACTION codes:

- 16 and 17 - military versus protesters; military versus civilians
- 26 and 27 - Rebels versus protesters; rebels versus civilians.
- 36 and 37 - Political militia versus protesters; Political militia versus civilians
- 56 and 57 - Rioters versus protesters; rioters versus civilians.

Limiting our sample to only this categories, raises a new concern. All the data that we are going to work with follows the same direction (attacking civilians or protestors), so we are not going to be able to make the initial distinction we wished to do, *A* attacks *B* or *B* attacks *A.*

2. Use SpaCy Part Of Speech tagger to find all verbs that appear in the corpus of the collection of documents. We do this using the complete corpus of our data, in order to extract all the lemmatized verbs that appear on any note. We then classify those verbs as related to conflict (the relations that we wish to extract from the text) or not manually, according to our own criteria.

3. We then label each observation from the dataset programmatically using regular expressions according to this list of verbs. If the lemmatized string of a verb appears within the string of the Notes for a particular observation, we identify that as the relationship for that observation. While this is a very imperfect approximation, it allowed us to keep a significant part of our data and give us enough time to focus on the building of the model for the analysis.

4. Update the strings of the text: The last pre-processing step we took, following the work done in (Keith et al, 2017), was to replace in the text of each of the units of analysis the places where the aggressor was mentioned by the literal string "AGGRESSOR" and reciprocally for victims. The objective of this replacement is to provide more information on where each word is with respect to the place of the sentence in which the aggressor and victims are mentioned which can allow the network to learn better or faster. To carry this out we needed to:
    a. Identify all the ways in which aggressors are mentioned in the data: Using regex and the *Actor* reference column, we extracted all the ways in which a group or organization is referred to. For example,'AQAP: Al Qaeda in the Arabian Peninsula' could be referred to as 'AQAP', 'Al Qaeda', or 'AQAP: Al Qaeda in the Arabian Peninsula'. We include all variations in a list of words to be replaced in the string, as well as all potential variations of "Unidentified Armed Groups" such as 'Unidentified Armed Men'. 'Unidentified Group of Armed Men', 'Unidentified Armed Soldiers', etc. We assume that every actor, as long as it is not "civilians" or "protestors", is an aggressor. In total, we come up with a list of 1295 potential strings to be replaced if they were to be found on the text strings.
    b. Identify all the ways in which victims are mentioned in the data: Since there is not as much variation as to the way of referencing the victims, the list of words to be replaced is smaller (length 16) and includes the following words:

Ideally the subset of the data that we would work with is that section that is (i) labelled, (ii) includes 'AGGRESSOR' in the string, (iii) and includes 'VICTIMS' in the string, as well as the initial constraint we introduced of the string being more than 100 characters. This limits our data from over 500,000 initial observations to only 8294 tagged observations. To try to expand a little bit more the analysis, we focus on the results we get from running the model using observations that include in the text *either '*AGGRESSOR' or 'VICTIM' (in order to use 34,556 observations), but we also look at the results with the smaller subset of the data.

LSTM and Bidirectional LSTM

Because we are interested in the relationship between *aggressor* and *victim*, which we have identified in the label, and the string altered so we know in which part of the sentence each of the actors is referenced, we use a recurrent neural network approach. In particular, we implement a Long Short Term Memory (LSTM). LSTMs are a type of RNN that can learn long-term dependencies by only keeping the relevant part of what has been processed of the sentence at each of the stages, solving for the *long-term dependency problem.*[1] The idea with implementing a Bi-directional LSTM is not only processing what comes before a given part of the sequence at a given point, processing the sequence until that point from the start and from the end of the sequence. As compared to the regular LSTM, the bidirectional version will have two hidden layers instead of one.

Instead of focusing on tuning many parameters for each type of model, we try to compare the results of a regular LSTM and a bidirectional LSTM, and see if there is any additional value in doing this computation that is computationally more expensive. Both are initialized using the default values with an Adam Optimizer, embedding the words in 10 dimensions through training, and with 40 hidden neurons per hidden layer.

For each one of the two models, we train up to 15 epochs (initially we did up to 50, but found that the point at which the error converged locally was usually around the 10th epoch), we then store the best out of all the epochs performed according to the evaluation on the validation set, along with the trained embeddings. We then apply it to the test set. We compare the performance of both the best model for the LSTM and the best one for the bidirectional LSTM.

Results

Comparing both approaches,[2] we don't see any advantage to using the bidirectional LSTM. In both cases we see an initial large increase in the accuracy (and reduction in the total computed loss) as the number of epochs increase, but that converges quite rapidly to a stable rate of around .72 as can be seen in Figures 8, and 9.

The accuracy evaluated on the test set is of .7342 for the regular LSTM, and of .7254 for the bidirectional approach. Given the small difference in the results, and the computational effort that the second approach entails, we conclude that for the specific goals, and the specific multi-classification problem that we are addressing, it is more adequate to use the regular LSTM.

[1] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[2] And after incorporating changes to the size of the hidden layer to adequately incorporate the results of the bidirectional approach as discussed after our presentation.

Figure 8. Accuracy LSTM in training.

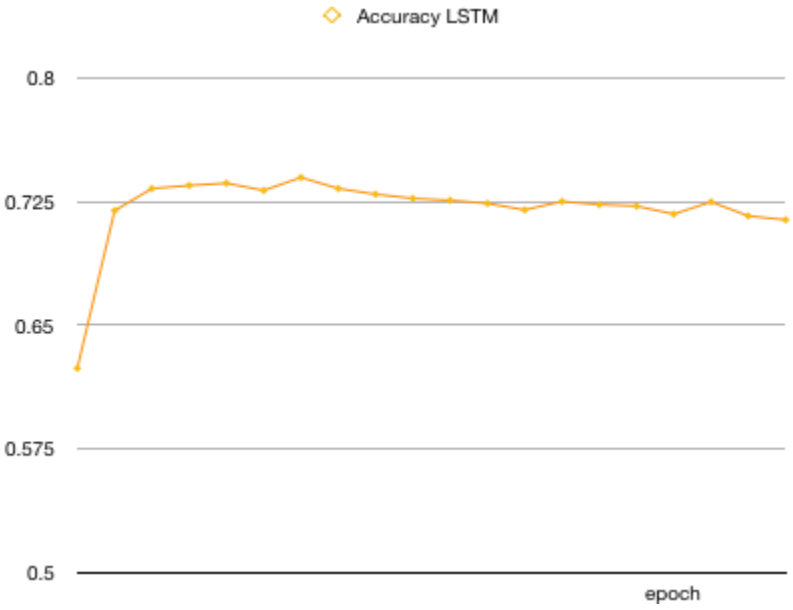
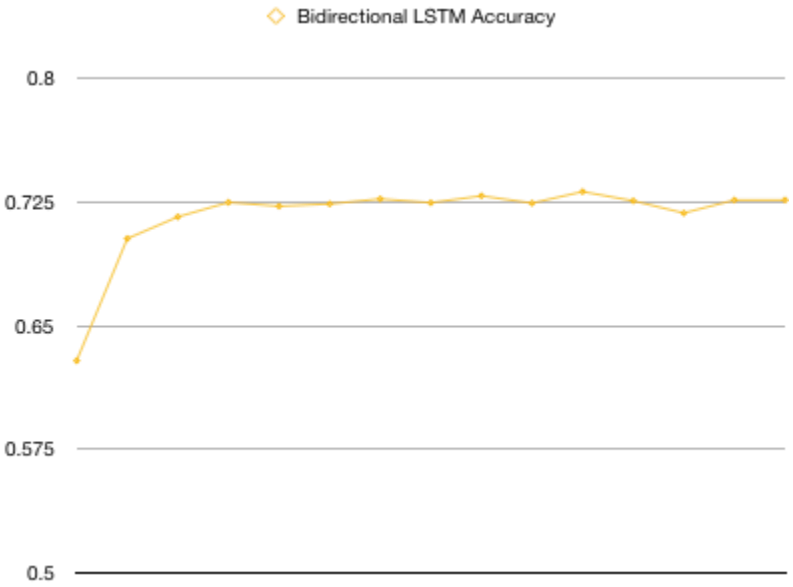


Figure 9. Accuracy Bidirectional LSTM



We proceed to conduct some tuning by trying different optimizers, learning rates, and regularizations. We only train each one with 8 epochs, and we adjust:

- Optimization parameter to be Adam or SGD
- Learning rate to be (start) at .001 or .0001
- Regularization to be 0 or .3

We find that what performs better after evaluating on the test set is the LSTM we initially performed, using an Adam optimizer with a loss rate of .001.

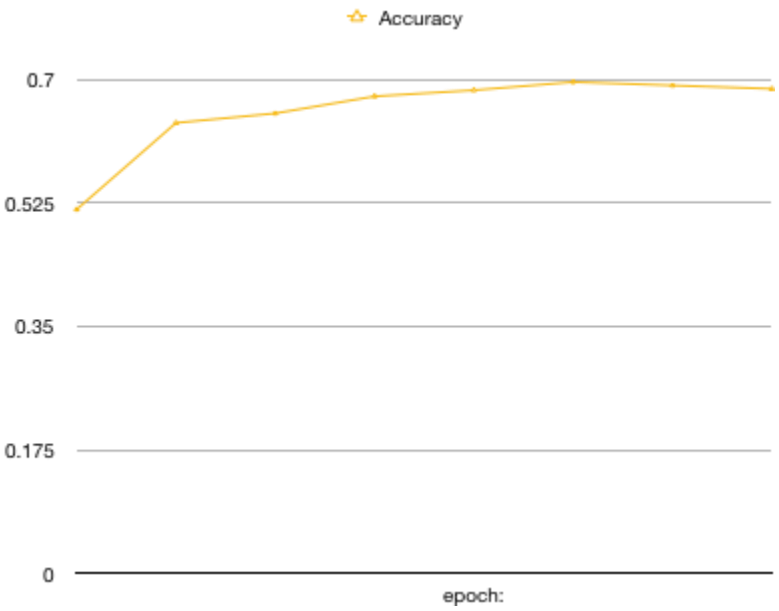
Something interesting to explore is whether we're predicting one specific category better than the others. After analyzing the results we find that the categories for which we're getting a high accuracy (higher than the average accuracy) are the following: injure, kill, attack, assault, fire, plant, arrest, clash, explode, detonate & lead, which are only 11 out of 333 categories that we have, but they are also the labels that appear the most frequently. This shows that our model might be overfitting for some particular categories.

injure	**0.990909**
kill	**0.983391**
attack	**0.89168**
assault	**0.884615**
fire	**0.88**
plant	**0.833333**
arrest	**0.803279**
clash	**0.8**
explode	**0.773585**
detonate	**0.75**
lead	**0.726415**
shoot	0.583333
burn	0.536585
beat	0.458333
gun	0.434783
wound	0.3875
shell	0.377358
execute	0.333333
fight	0.288889
bomb	0.277778
protest	0.271186
loot	0.189189

abduct	0.146853
kidnap	0.128571
(…)	

Finally, we also use this best performing model parameters, and train them on the smaller subsample of data that we had, namely the one that had for every observation the text with the characters “AGGRESSOR” and “VICTIM”. The performance of this is actually worse, having an accuracy of .67 when evaluated on the test set. However, this might have been due to timing constraints and perhaps having trained with more epochs the accuracy would have continued on the rise.

Figure 10: Accuracy LSTM - smaller sample of data



II.III Goal 3. Topic Modeling

One of the key goals for the project was also to leverage Topic Modeling for the following purposes:

1. Extract the underlying themes & topics in the ACLED
2. Study these topics to identify the similarities in conflicts around different parts of the world
3. Understand the distributions or balance between such conflict themes across geographies and over a long time period

The questions above, if answered, would help researchers and policy practitioners identify similarities between conflicts that have been hitherto considered unrelated, and hence been studied independently. The Topic Modeling analysis can help bridge studies and learnings across conflicts and their resolutions.

As per our initial proposal we aimed to replicate the recent methodology presented by Altuncu et al. from the Imperial College London. However, a more thorough review of the literature and the implementation of the approach presented, helped us realize that the methodology is a fundamentally new approach to supervised clustering, and would have been unnecessarily intensive for our policy objectives.

We therefore decided to adopt a supervised clustering using a Latent Dirichlet Allocation (LDA) model, which is the most commonly used model for topic modeling. This in turn allowed us to focus more on analyzing the topics extracted. We could, hence, also built a partially automated pipeline for topic modeling & visualization that can help researchers further explore the ACLED.

Methodology

As opposed to goal 1 and goal 2, we work with the full dataset, not filtering out anything since Topic Modeling is inherently an exploratory exercise and any insights from the missing or shorter conflict notes are equally valuable.

The complete process we follow is:

1. We use pre-trained Word2Vec embeddings available with spaCy (‘en_core_web_md’ model) where each word is represented as a 300-dimension vector

2. All conflict notes are tokenized by lemmatizing the words, ignoring the standard ‘English’ stopwords (as defined in spaCy). We also remove any pronouns from the notes after Parts-Of-Speech tagging

3. We leverage Bag-of-Words features for the LDA model with unigrams (one-word tokens only). We deliberately focus on all words, and not just nouns or noun-phrases since topics / themes related to actions (verbs, adverbs, etc.) can also provide important insights. A

continuous BoW approach was initially planned but could not be completed due to the long run-times encountered with the LDA models, and corresponding visualizations.

4. We use the Latent Dirichlet Allocation (LDA) for the Topic Modeling, given that it is the most commonly used and the easiest approach to exploratory topic modeling.

5. The LDA model however implements a 'soft clustering'. We use the pyLDAvis library functionalities to create a D3.js based interactive visualization for each LDA model run. The soft clusters are then converted to 'hard clusters' and we visualize these against other categorical fields available in the ACLED such as region, event type, actor category, etc.

LDA

The Latent Dirichlet Allocation (LDA) approach we use is very similar to a Bag-of-Words Classifier but works with inherent frequency distributions instead of optimizing against a target classification. Based on a given 'number of topics' every conflict note is mapped to each topic through a vocabulary length 'items'.

A sparse matrix (509,157 X 39,799) is then reduced to a denser matrix to assign topic distribution for each conflict note, where 509,157 corresponds to the total number of conflicts or the total observations in the training data; while 39,799 corresponds to the size of the vocabulary after lemmatization & tokenization. The model output corresponds to a (509,157 X no. of topics) matrix, where each row corresponds to the conditional probabilities of topics being related to the given conflict, i.e. p(Topic | Conflict).

Hyperparameters / Key Features

Key features of the LDA Model are

1. <u>the number of topics</u>
   which determines the dimensionality of the intermediate 'embedding' matrices being trained
2. <u>vectorizer type</u>
   this determines how the Bag of Words features are generated, and hence the raw frequency distributions inputted to the Dirichlet distributions. We look at 2 key vectorizer
   a. Count – taking the raw frequency of occurrence of the token
   b. TFIDF – term-frequency inverse-document frequency which also incorporates the distinctiveness of a token
3. <u>max iterations</u>
   this corresponds to the number of 'epochs' run for the model

We discuss the LDA model results in greater detail below. However, we would like to note key observations upfront, as follows:

1. max iterations hyperparameter does not have a significant impact
2. TFIDF vectorizer generates more 'coherent' or more easily explainable topics

Results

The pyLDAvis library enabled us to create D3.js based interactive visualizations for each of the LDA models we run. All of the visualizations are hosted on GitHub pages here and have been hyperlinked below.

We discuss few of the models run to highlight the key findings & relevant insights

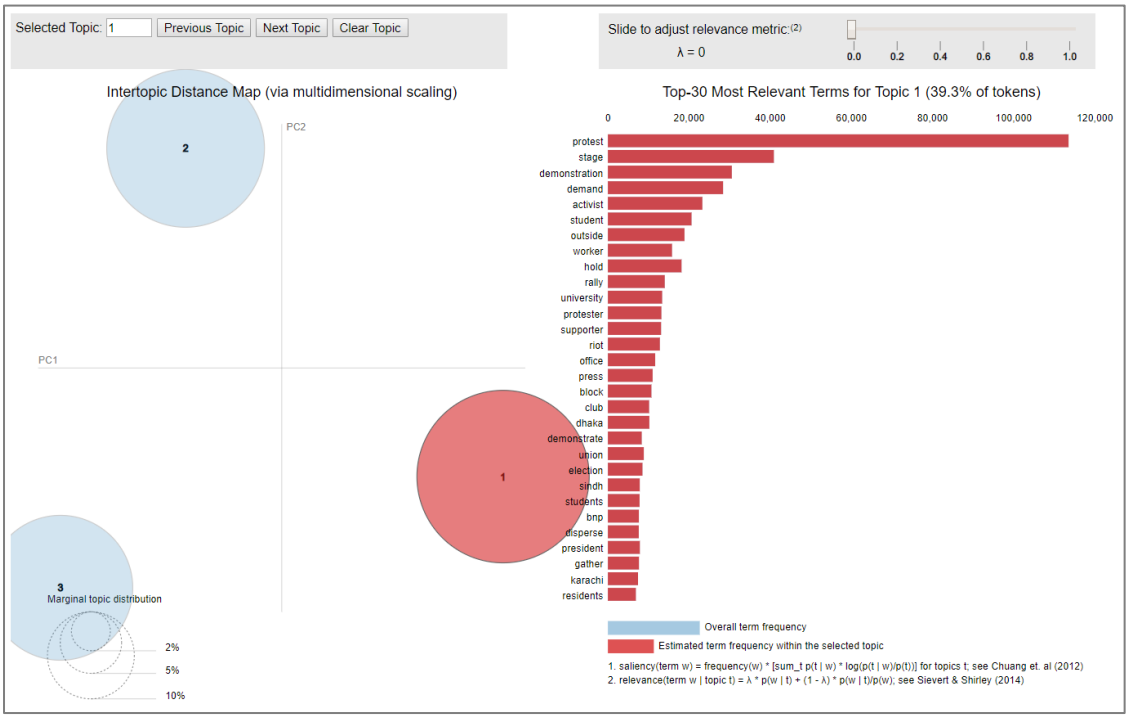
1. **3 Topics | Count vectorizer | 2 max iterations**
   This LDA model is run with 3 topics with a Count Vectorizer (raw count features) with max 2 training epochs. The hyperlink above hosts the interactive visualization hosted on GitHub pages.

   The Topics could potentially be tagged as below:
   a. Civil Unrest (Topic #1)
   b. Two-sided clash (Topic #2)
   c. One-sided attack (Topic #3)

   By keeping $\lambda = 0$ (inspecting the top-30 most relevant tokens for Topic #1), we can see that the most relevant tokens are 'protest', 'stage', 'demonstration', 'demand', 'activist', etc. This shows us that Topic #1 is successfully identifying all 'Civilian Unrest' conflicts which cause public protests, most of which are peaceful without casualties or severe harm or injuries.

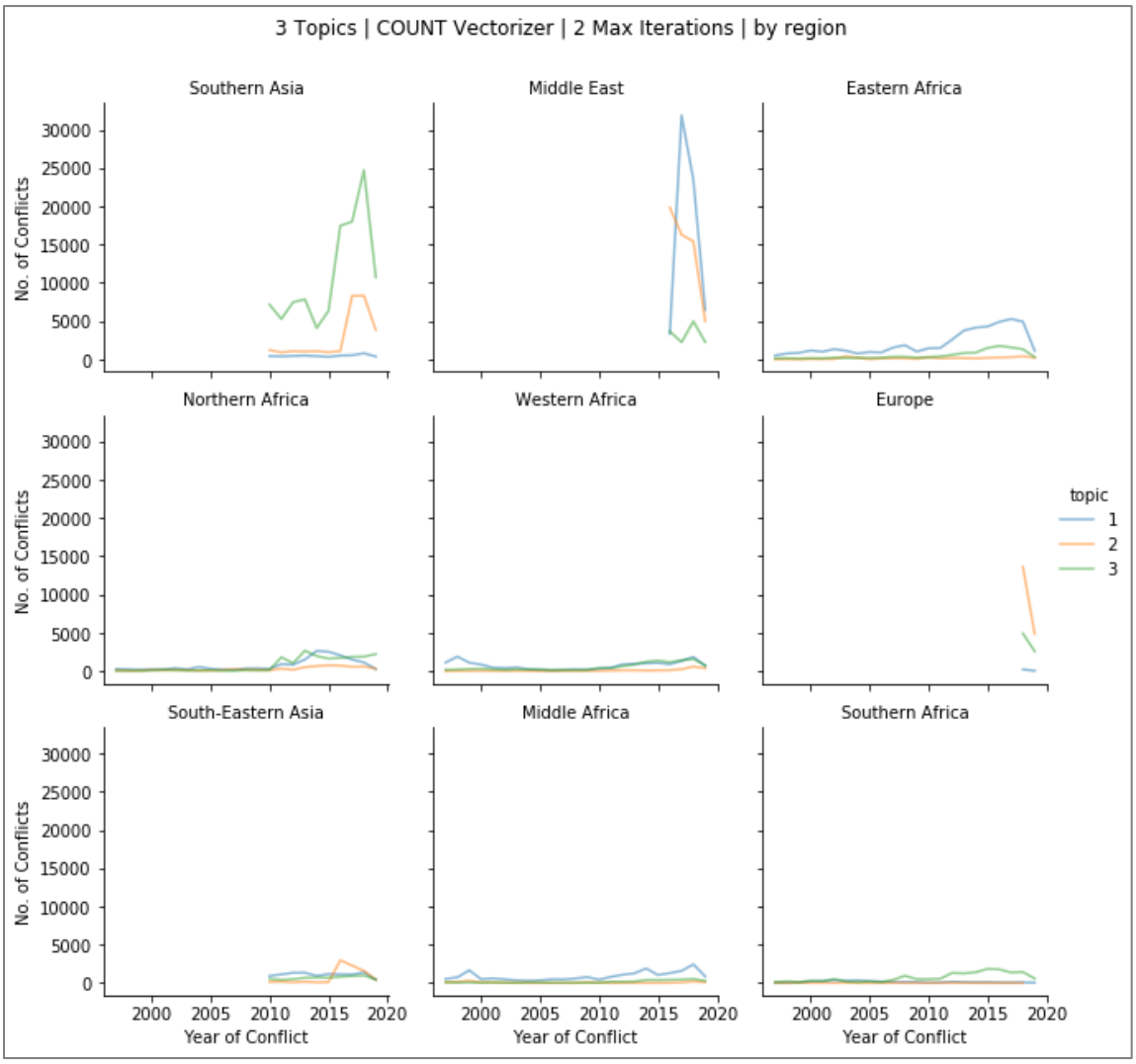
   Similar, inspecting Topic #2 & Topic #3, we can classify them as 'Two-sided clash' and 'One-sided attack' respectively.

   Figure 11: Interactive Visualization for LDA model with 3 Topics | Count Vectorizer
   Top-30 most relevant tokens for Topic #1 can be seen here

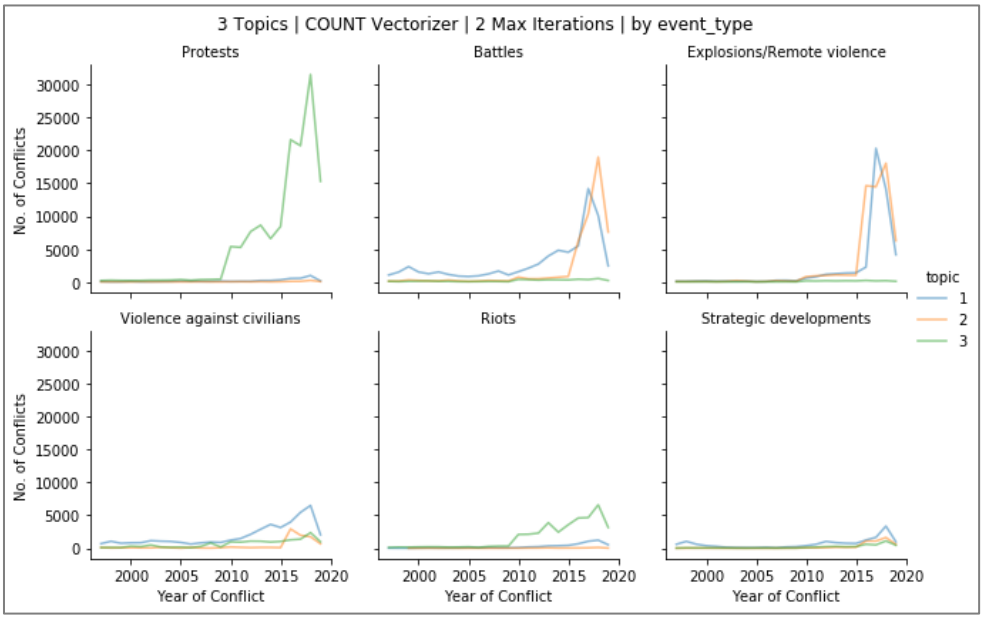


Next, we convert the above LDA model results into 'hard clusters' by choosing the most probable topic for each conflict, i.e. $argmax\{\, p(\, Topic \mid Conflict\,)\,\}$.

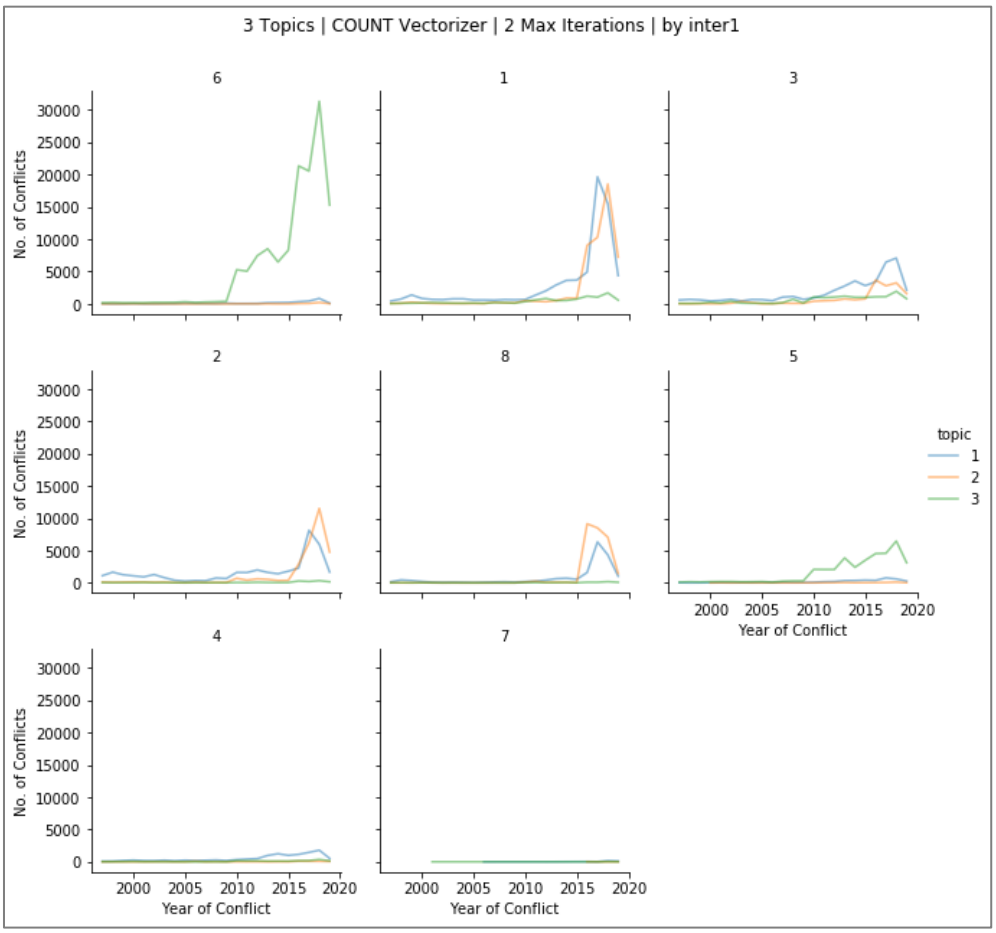
Figure 12: LDA model with 3 Topics | Count Vectorizer | 2 iterations
(a) Topic distribution trends by region (as defined by ACLED)



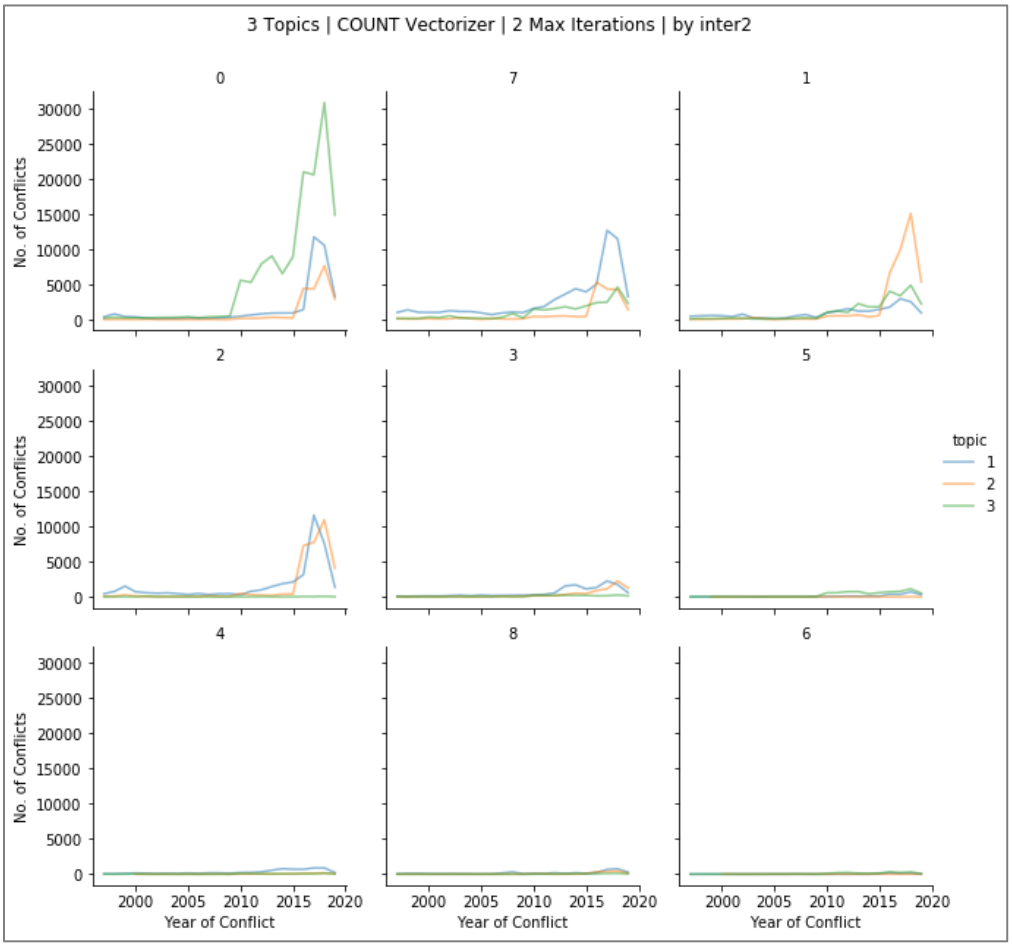
(b) Topic distribution trends by event type (as defined by ACLED)



(c) Topic distribution trends by Actor1 category ('inter1' as defined by ACLED)



(d) Topic distribution trends by Actor2 category ('inter2' as defined by ACLED)



2. **[3 Topics | Count vectorizer | 5 max iterations]**
   We run the same LDA model but now with 5 max iterations (training epochs). The pyLDAvis visualizations are hosted on GitHub pages hyperlinked above.

   We can easily see that the Topics extracted, the most relevant tokens for each topic, as well as other characteristics of the Topics do not change much with the 3 additional epochs. Thus, we can say that increasing the 'max iterations' hyperparameter does not significantly impact the topic modeling results. We therefore run all our subsequent models with $max\_iterations = 2$

3. **[3 Topics | TFIDF vectorizer | 2 max iterations]**
   When we use a TFIDF (term-frequency inverse-document frequency) vectorizer to accurately incorporate distinctiveness of tokens along with the raw count, the LDA model results are considerably different!

   Though Topic #1 & Topic #2 remain relatively unchanged, Topic #3 now corresponds to some other theme / topic, and not necessarily the Topic #3 identified with a Count vectorizer.

Given the above findings, we lay more emphasis on LDA models run with a TFIDF vectorizer going forward, though corresponding models with a Count vectorizer were also run for a comparative analysis.

4. **5 Topics | TFIDF vectorizer | 2 max iterations**
   Corresponding LDA model with a Count vectorizer was also run, and the visualization for the same can be seen here.

5. **7 Topics | TFIDF vectorizer | 2 max iterations**
   Corresponding LDA model with a Count vectorizer was also run, and the visualization for the same can be seen here.

6. **10 Topics | TFIDF vectorizer | 2 max iterations**
   Corresponding LDA model with a Count vectorizer was also run, and the visualization for the same can be seen here.

Soft Clustering results for all models were converted to 'hard clusters' using $argmax\{\, p(\, Topic \mid Conflict\,)\,\}$. The corresponding topic distribution trends by region, event type, and interactor categories, can be found on the project GitHub repository here.

Considerations / Limitations

Since the Topic Modeling exercise is exploratory, certain assumptions were made and liberties were taken to expedite the process and focus on analyzing initial results. However, the process should be repeated iteratively till a coherent set of Topics is extracted, and which can be used further by researchers and policy makers.

Key considerations to be made during the same are as follows:

1. LDA provides soft clusters with conditional probabilities $p(Topic \mid Conflict\ Note)$. There is no absolute objective way to convert these into 'hard clusters' to assign one topic for each conflict. Certain conflicts may be mis-categorized if we use a strictly argmax criteria.

2. LDA models run so far were based on unigram BoW features, and hence any information or insights contained in the sequence or combination of the words / tokens were not extracted efficiently. A continuous BoW approach is therefore highly recommended!

3. Using the pre-trained Word2Vec word embeddings may introduce certain biases, especially since these embeddings were trained on a wider, more generalized corpus from Wikipedia and not on conflict-related news articles.

   For example, Topics extracted when model is run with 7 or 10 topics club countries or sub-regions together (perhaps since Word2Vec embeddings are closer for words denoting geography). However, the conflicts in these areas may not necessarily be connected or similar. Hence, using word embeddings obtained from a conflict-related corpus should be used!

IV. Final considerations

Our initial approach was to work together on all the goals. We all wanted to understand how different NLP methods work, and what steps are relevant for each different kinds of objectives. In the end, we did work together on discussing methodology for each of the goals and coming up with the implementation that we were going to do for each section, although we divided up the implementation of each goal. Gonzalo focused on goal 1, Cristina focused on goal 2, and Darshan focused on Goal 3.

For goal 1 and goal 2 a big part of the implementation was learning how to take the base code we had built for our homework 1 and adapt it for more complicated architectures by specifying hidden networks.

Something relevant for goal 1 was identifying the relevant input for the model. As mentioned, the original plan was to use Stanford NER, but during initial testing this didn't work well. After researching, we decided pre-trained SpaCy embeddings were better suited for this objective, and so we had to learn how to use them for the model. Some additional creative work had to be done to analyze our output with more detail, so we could understand what our results meant and how the accuracy of the network varied across different labels.

The biggest challenge for goal 2 was learning how to take this base code and adapt it for a recurrent neural network, in particular an LSTM. We had not previously done embedding layers nor processing sequences, so it was a valuable learning experience. Looking at different sources and understanding the different approaches was useful but challenging, because in many times they did not have the same objective as us (multi classification), so we did have to work a lot on the code and the shape of the inputs and outputs to make the pipeline work.

For goal 3, the biggest challenge was to identify the most appropriate methodology to follow for the Topic Modeling. The recent research published by Altuncu et al. from Imperial College London was theoretically the most appropriate. However, implementation of the same in pyTorch or other Python libraries proved to be a steep uphill task. Switching back to using LDAs forced us to make certain assumptions and take certain liberties in order to be able to thoroughly analyze the results. This was a critical learning in terms of output-oriented project & task management, be always prioritizing the actionable insights we can provide to researchers and policymakers. Building a semi-automated end-to-end pipeline for topic modeling complemented by interactive & static visualizations was also a very enriching & fulfilling learning experience.

None of us had used any of the tools we used for the project before, except for the pyTorch code from the first assignment of this course.

Bibliography

1. Identifying civilians killed by police with distantly supervised entity-event extraction. Katherine A. Keith, Abram Handler, Michael Pinkham, Cara Magliozzi, Joshua McDuffie, and Brendan O'Connor. Proceedings of EMNLP 2017.

2. Learning to Extract International Relations from Political Context. Brendan O'Connor, Brandon M. Stewart, and Noah A. Smith. Proceedings of ACL 2013.

3. Content-driven, unsupervised clustering of news articles through multiscale graph partitioning. M. Tarik Altuncu, Sophia N. Yaliraki, Mauricio and Barahona. Data Science, Journalism & Media workshop 2018.

Appendix 1

10- SOLE MILITARY ACTION	26- REBELS VERSUS PROTESTERS	47- COMMUNAL MILITIA VERSUS CIVILIANS
11- MILITARY VERSUS MILITARY	27- REBELS VERSUS CIVILIANS	48- COMMUNAL MILITIA VERSUS OTHER
12- MILITARY VERSUS REBELS	28- REBELS VERSUS OTHERS	50- SOLE RIOTER ACTION
13- MILITARY VERSUS POLITICAL MILITIA	30- SOLE POLITICAL MILITIA ACTION	55- RIOTERS VERSUS RIOTERS
14- MILITARY VERSUS COMMUNAL MILITIA	33- POLITICAL MILITIA VERSUS POLITICAL MILITIA	56- RIOTERS VERSUS PROTESTERS
15- MILITARY VERSUS RIOTERS	34- POLITICAL MILITIA VERSUS COMMUNAL MILITIA	57- RIOTERS VERSUS CIVILIANS
16- MILITARY VERSUS PROTESTERS	35- POLITICAL MILITIA VERSUS RIOTERS	58- RIOTERS VERSUS OTHERS
17- MILITARY VERSUS CIVILIANS	36- POLITICAL MILITIA VERSUS PROTESTERS	60- SOLE PROTESTER ACTION
18- MILITARY VERSUS OTHER	37- POLITICAL MILITIA VERSUS CIVILIANS	66- PROTESTERS VERSUS PROTESTERS
20- SOLE REBEL ACTION	38- POLITICAL MILITIA VERSUS OTHERS	67- PROTESTERS VERSUS CIVILIANS
22- REBELS VERSUS REBELS	40- SOLE COMMUNAL MILITIA ACTION	68- PROTESTERS VERSUS OTHER
23- REBELS VERSUS POLITICAL MILIITA	44- COMMUNAL MILITIA VERSUS COMMUNAL MILITIA	78- OTHER ACTOR VERSUS CIVILIANS
24- REBELS VERSUS COMMUNAL MILITIA	45- COMMUNAL MILITIA VERSUS RIOTERS	80- SOLE OTHER ACTION
25- REBELS VERSUS RIOTERS	46- COMMUNAL MILITIA VERSUS PROTESTERS	

Appendix 2

60- SOLE PROTESTER ACTION

“A protest was staged in Colombo on 26 October 2017 demanding the release of IUSF activists and other students in remand custody.”

12- MILITARY VERSUS REBELS

“MoD reports Afghan army conducted military operations across 17 provinces; killing 36 suspected Taliban militants. Fatalities split across 15 provinces, with some specific events listed in article. 21 fatalities coded in this series while 15 previously coded in other events. 15 Events in total as operations, with Nangarhar and Kunduz were already coded separately.”

10- SOLE MILITARY ACTION

“Military forces established an operational base in Kazimiya, in order to better control this area on Lake Tanganyika.”

80- SOLE OTHER ACTION

“The Saudi-led coalition carried out three air raids on the Atias mountain and two air raids on the Nashr area in the Sirwah district, Marib governorate. No casualties were reported but private property was damaged.”

18- MILITARY VERSUS OTHER

“Pro-Houthi forces claim to have shelled Saudi soldiers in Raqabat-sudais and Makhroq, Najran. No injuries reported.”

Appendix 3

		R-squared:	0.424
		Adj. R-squared:	0.423
		F-statistic:	934.3
		Prob (F-statistic):	0.00

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.9505	0.007	141.122	0.000	0.937	0.964
C(interaction_x)[T.11]	-0.2698	0.011	-24.932	0.000	-0.291	-0.249
C(interaction_x)[T.12]	0.0538	0.007	7.582	0.000	0.040	0.068
C(interaction_x)[T.13]	-0.5507	0.010	-55.623	0.000	-0.570	-0.531
C(interaction_x)[T.14]	-0.8556	0.028	-30.209	0.000	-0.911	-0.800
C(interaction_x)[T.15]	-0.2332	0.012	-20.215	0.000	-0.256	-0.211
C(interaction_x)[T.16]	-0.2632	0.011	-23.404	0.000	-0.285	-0.241
C(interaction_x)[T.17]	-0.2592	0.009	-27.333	0.000	-0.278	-0.241
C(interaction_x)[T.18]	-0.1626	0.014	-11.827	0.000	-0.190	-0.136

C(interaction_x)[T.20]	-0.7882	0.013	-62.517	0.000	-0.813	-0.764
C(interaction_x)[T.22]	-0.3820	0.015	-26.063	0.000	-0.411	-0.353
C(interaction_x)[T.23]	-0.8225	0.016	-50.463	0.000	-0.854	-0.791
C(interaction_x)[T.24]	-0.8416	0.040	-21.035	0.000	-0.920	-0.763
C(interaction_x)[T.25]	-0.8599	0.095	-9.098	0.000	-1.045	-0.675
C(interaction_x)[T.26]	-0.8385	0.120	-6.963	0.000	-1.074	-0.602
C(interaction_x)[T.27]	-0.3260	0.012	-26.406	0.000	-0.350	-0.302
C(interaction_x)[T.28]	-0.5199	0.014	-37.498	0.000	-0.547	-0.493
C(interaction_x)[T.30]	-0.5780	0.013	-44.781	0.000	-0.603	-0.553
C(interaction_x)[T.33]	-0.7717	0.020	-38.139	0.000	-0.811	-0.732
C(interaction_x)[T.34]	-0.8717	0.048	-18.153	0.000	-0.966	-0.778
C(interaction_x)[T.35]	-0.8570	0.062	-13.730	0.000	-0.979	-0.735

C(interaction_x)[T.36]	-0.8818	0.095	-9.330	0.000	-1.067	-0.697
C(interaction_x)[T.37]	-0.2394	0.009	-26.629	0.000	-0.257	-0.222
C(interaction_x)[T.38]	-0.8612	0.015	-57.975	0.000	-0.890	-0.832
C(interaction_x)[T.40]	-0.8874	0.108	-8.239	0.000	-1.098	-0.676
C(interaction_x)[T.44]	-0.4109	0.021	-19.227	0.000	-0.453	-0.369
C(interaction_x)[T.45]	-0.8567	0.196	-4.362	0.000	-1.242	-0.472
C(interaction_x)[T.47]	-0.7299	0.016	-45.226	0.000	-0.761	-0.698
C(interaction_x)[T.48]	-0.8746	0.120	-7.266	0.000	-1.111	-0.639
C(interaction_x)[T.50]	-0.6646	0.013	-49.734	0.000	-0.691	-0.638
C(interaction_x)[T.55]	-0.6196	0.015	-40.242	0.000	-0.650	-0.589
C(interaction_x)[T.56]	-0.8523	0.060	-14.093	0.000	-0.971	-0.734
C(interaction_x)[T.57]	-0.7128	0.012	-59.717	0.000	-0.736	-0.689

C(interaction_x)[T.58]	-0.3709	0.019	-19.064	0.000	-0.409	-0.333
C(interaction_x)[T.60]	0.0903	0.007	13.278	0.000	0.077	0.104
C(interaction_x)[T.66]	-0.8446	0.083	-10.207	0.000	-1.007	-0.682
C(interaction_x)[T.67]	-0.8641	0.120	-7.179	0.000	-1.100	-0.628
C(interaction_x)[T.68]	-0.8143	0.042	-19.383	0.000	-0.897	-0.732
C(interaction_x)[T.78]	-0.4518	0.016	-28.075	0.000	-0.483	-0.420
C(interaction_x)[T.80]	-0.1601	0.012	-13.582	0.000	-0.183	-0.137
C(year)[T.2019]	-0.0155	0.003	-5.109	0.000	-0.021	-0.010
notes_char	-0.0003	1.8e-05	-16.925	0.000	-0.000	-0.000