

4.3 Conceptual Framework

This study focuses on classifying climate change tweets from the dataset acquired. This study attempts to determine the sentiment of a given tweet, based on the words that the tweet used. The conceptual framework is composed of 3 parts: *Data Preprocessing with LDA*, *Word Embedding Generation*, and *Developing a Text Classification Model*.

Figure 6 illustrates the entire Conceptual Framework of the study.

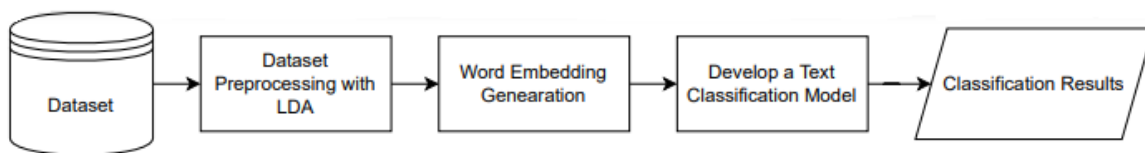


Figure 6: Conceptual Framework

Figure 6 shows the general process on how this study is done. The Dataset is first preprocessed, which also includes LDA. For word embedding generation, the dataset is partitioned into training data and testing data. The training set will be used in creating FastText word embeddings, and are used as an input for the ANN classification model. After training the model with the given training data, the model will then be evaluated with the testing data. Classification results will then be measured through performance metrics such as accuracy, precision, recall, and f1 score. Each part of the Conceptual Framework will be explained further.

Figure 7 illustrates the first part of the conceptual framework, which is *Dataset Preprocessing with LDA*.

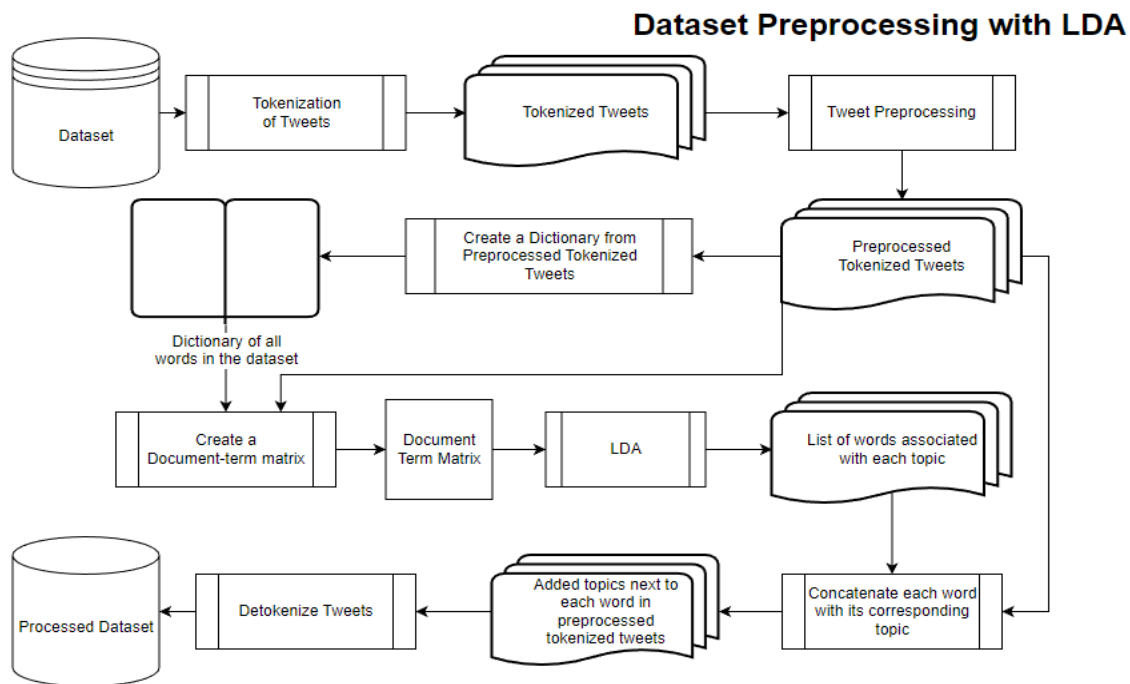


Figure 7: Conceptual Framework of the study : Dataset Preprocessing with LDA

4.3.1 Dataset Preprocessing with LDA

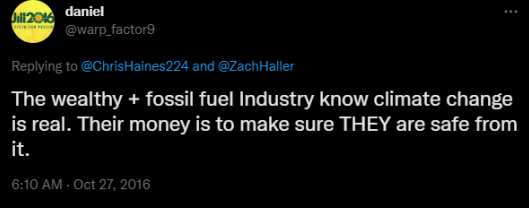
The tweet column will be the feature to be preprocessed in this phase from the dataset being used.

4.3.1.1 Tokenization of Tweets

Tokenization is done to every tweet, wherein each tweet is broken down into a list of words that make up that tweet. This is done to the tweets for easier preprocessing.

Table 3

Tokenized Tweet

Tweet	Tokenized Tweet
	["@ChrisHaines224", "@ZachHaller", "The", "wealthy", "+", "fossil", "fuel", "Industry", "know", "climate", "change", "is", "real.", "Their", "money", "is", "to", "make", "sure", "THEY", "are", "safe", "from", "it."]

4.3.1.2 Tweet Preprocessing

In Tweet Preprocessing, this research makes use of 2 phases, first being the removal of the entire word, and second being removing parts of a word. For phase 1, this includes removing stop words, which majority of these kinds of words are prepositions. After, hashtags are removed, URLs, and lastly mentions. For phase 2, special characters are removed, numbers, and all characters for every word are converted to lowercase. In the figure below, it's the example from the previous figure, which was the tokenized tweet. Words highlighted red during every phase are the words or characters to be removed for that phase. Preprocessing can be done with the help of Python libraries such as *RE*, *SpaCy*, and *NLTK*.

Table 4

Tweet Preprocessing

Tokenized Tweet
["@ChrisHaines224", "@ZachHaller", "The", "wealthy", "+", "fossil", "fuel", "Industry", "know", "climate", "change", "is", "real.", "Their", "money", "is", "to", "make", "sure", "THEY", "are", "safe", "from", "it."]
Phase 1
["@ChrisHaines224", "@ZachHaller", "The", "wealthy", "+", "fossil", "fuel", "Industry", "know", "climate", "change", "is", "real.", "Their", "money", "is", "to", "make", "sure", "THEY", "are", "safe", "from", "it."]

Phase 2
["wealthy", "+", "fossil", "fuel", "Industry", "know", "climate", "change", "real", "money", "make", "sure", "safe", "from"]
Preprocessed Tokenized Tweet
["wealthy", "fossil", "fuel", "industry", "know", "climate", "change", "real", "money", "make", "sure", "safe", "from"]

4.3.1.3 Create a Dictionary

A Dictionary is then created from all the preprocessed tokenized tweets, wherein each word is assigned with a unique ID, and all words in the dictionary are unique. This is done in order to create a Document-term Matrix, where the rows of matrix are the preprocessed tokenized tweets, and the columns are the words of the Dictionary. For example, from the preprocessed tokenized tweet, each word is paired with a key, and all other words in every tweet in the dataset are mapped to the dictionary. In trying to add a new word, the Dictionary first checks if that particular word exists or not. If it does exist, then the word will not be added.

Table 5

Adding words from the preprocessed tokenized tweet to the Dictionary

Preprocessed Tokenized Tweet	Dictionary
["wealthy", "fossil", "fuel", "industry", "know", "climate", "change", "real", "money", "make", "sure", "safe", "from"]	Dictionary = {1: "wealthy", 2: "fossil", 3: "fuel", 4: "industry", 5: "know", 6: "climate", 7: "change", 8: "real", 9: "money", 10: "make", 11: "sure", 12: "safe", 13: "from", 14: ..., nth key: nth word}

4.3.1.4 Create a Document-term Matrix

The Document-term Matrix is created in order to convert text format into numeric data. It is the input for performing LDA.

Table 6

Document-term Matrix

Tweet	change	climate	energy	evolution	fossil	from	fuel	industry	know	make	money	necessary	ready	real	reduce	revolutionary	safe	steps	sure	take	through	wealthy	nth word
["wealthy", "fossil", "fuel", "industry", "know", "climate", "change", "real", "money", "make", "sure", "safe", "from"]	1	1	0	0	1	1	1	1	1	1	1	0	0	1	0	0	1	0	1	0	0	1	...
[ready, take, necessary, steps, reduce, climate, change, through, revolutionary, energy, evolution]	1	1	1	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	1	1	0	...
nth tweet

4.3.1.5 LDA

LDA requires some hyperparameters, which are the number of topics, and the number of iterations. The number of topics is an estimate and an empirical choice for the researchers, also with the number of iterations. The output of the LDA Model is a list of words associated for each topic with its given probabilities. The process of LDA can be done with the help of the Python library *gensim*.

Table 7

Input, Process, and Output of LDA

Input: Document-Term Matrix, Hyperparameters: # of topics, # of iterations
Randomly Assigned topics to each word for every tweet
[wealthy = Topic1, fossil = Topic3, fuel = Topic1, industry = Topic2, know = Topic1, climate = Topic3, change = Topic2, real = Topic1, money = Topic2, make = Topic2, sure = Topic1, safe = Topic3, from = Topic2]
Reassign topics to each word
[wealthy Topic2, fossil Topic2, fuel Topic2, industry Topic2, know Topic1, climate Topic1, change Topic1, real Topic1, money Topic2, make Topic3, sure Topic3, safe Topic3, from Topic3]
Repeat based on the # of iterations
Output: Word Distribution for each Topic <ul style="list-style-type: none"> • Topic #1 - 40% climate, 40% change, nth word • Topic n - ...

4.3.1.6 Add topics next to each word

For the preprocessed tokenized tweet, each word is concatenated with its given topic to the right, which overall adds a lot of features to each given tweet. These topics are then renamed, and how these topics are named is an empirical choice for the researchers.

4.3.1.7 Detokenize Tweets

This is done so that all tweets are no longer a list of words, and are back to being an entire string of text.

Table 8

Adding topics next to each word and Detokenize tweets process

Topics next to each word in the preprocessed tokenized tweet
[wealthy Topic2, fossil Topic2, fuel Topic2, industry Topic2, know Topic1, climate Topic1, change Topic1, real Topic1, money Topic2, make Topic3, sure Topic3, safe Topic3, from Topic3]
Renaming each topic: Topic1 = environment, Topic2 = energy, Topic3 = safety
[wealthy energy, fossil energy, fuel energy, industry energy, know environment, climate environment, change environment, real environment, money energy, make safety, sure safety, safe safety, from safety]
Detokenize Tweet
[wealthy energy fossil energy fuel energy industry energy know environment climate environment change environment real environment money energy make safety sure safety safe safety from safety]

Figure 8 illustrates the second and third part of the Conceptual Framework, which is *Word Embedding Generation* and *Developing a Text Classification Model* respectively.

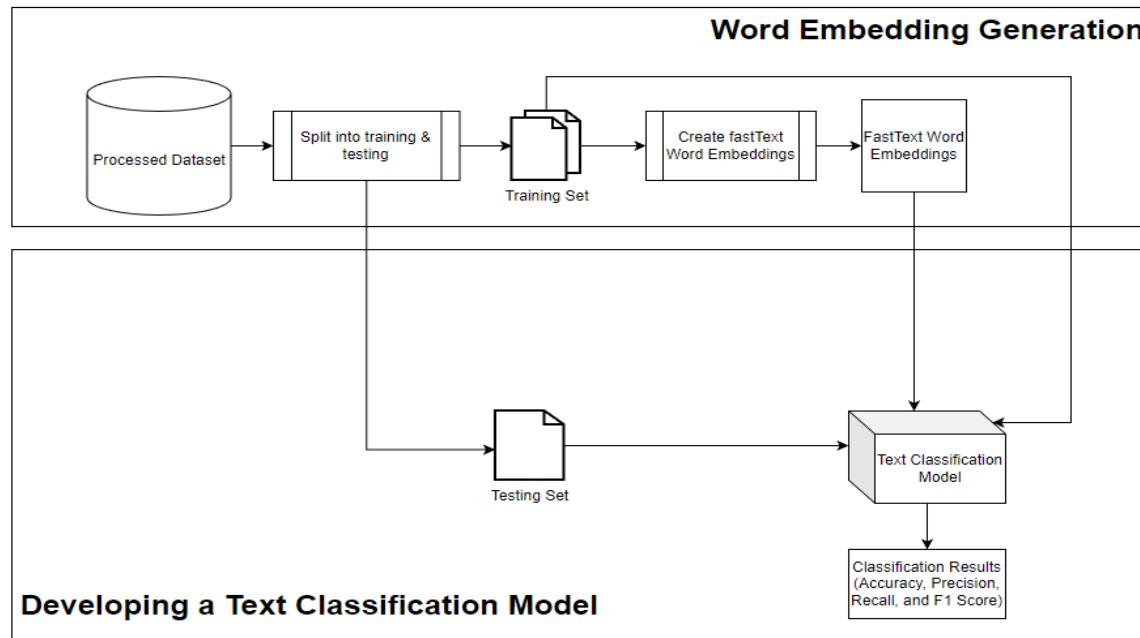


Figure 8: Conceptual Framework of the study : Word Embedding Generation and Developing a Text Classification Model

4.3.2 Word Embedding Generation

With the preprocessed dataset from the Dataset Preprocessing with LDA step, this will be used to create word embeddings.

4.3.2.1 Split into training & testing

The dataset is then split into Training and Testing. The Training Set is used in order to generate FastText Word Embeddings.

4.3.2.2 Create FastText Word Embeddings

FastText word embedding generation is done with the FastText python library. A FastText model provides a vector representation of all words in the training set, and a vector representation estimate of out of vocabulary words that will later be created when using the model to look up words from the testing set.

4.3.3 Developing a Text Classification Model

With FastText word embeddings created, these are then used to convert words in a training tweet into numeric form, and then fed into an ANN for the text classification task.

4.3.3.1 Text Classification Model

Training data, wherein each word in a training tweet is now in vector form, is used as an input to train the created text classifier. The text classifier is an ANN, created using the Keras library. Once training is finished, the text classification model has been created, and its performance will be evaluated using the testing set.

4.3.3.2 Classification Results

After testing, performance metrics such as accuracy, precision, recall, and f1 score will be measured and used to see the overall performance of the model. K-fold Cross-Validation will also be performed along with the performance metrics mentioned to validate how well the model performs with the dataset overall and to check whether or not the model performs consistently.