EXTRACTING AND ORGANIZING DISASTER-RELATED PHILIPPINE COMMUNITY RESPONSES FOR AIDING NATIONWIDE RISK REDUCTION PLANNING AND RESPONSE

Technical Manual

A Master's Thesis Proposal
Presented to
the Faculty of the College of Computer Studies
Graduate Studies Program
De La Salle University – Manila

In Partial Fulfillment
of the Requirements for the Degree of
Master of Science in Computer Science

by
Nocon, Nicco Louis S.

Ms. Charibeth K. Cheng
Faculty Adviser

May 20, 2020

Contents

1 Overview

This document discusses the contents of a Filipino text analysis tool. It is a Python Application Programming Interface (API) or library with a collection of functions. The list of functions and sample usages are included, located further on the document. Overall, the API's task is to extract valuable information or insights (specifically actions and target subjects) from a group of text, organize that information, and generate a report out of it.

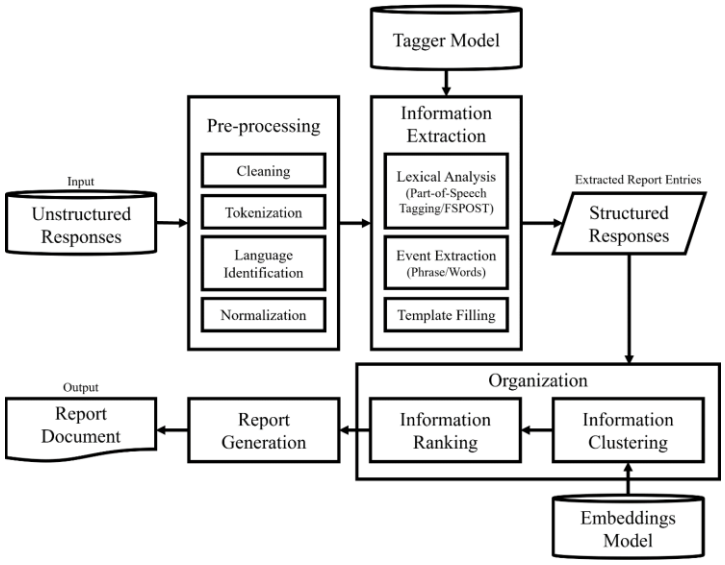

Figure 1.1 Architectural Diagram

It has nine parts that represents the tool's modules, namely Data Utilities, Normalization, Language Identification, Filipino Part-of-Speech Tagger, Information Extraction, Information Organization, Information Clustering, Information Ranking, and Report Generation. Illustrating this, the architectural diagram is shown at Figure 1.1, followed by brief descriptions of every module. As this API has been used primarily on Malasakit Responses dataset, a class object has been created with the following attributes shown on Table 1.1 – this object is used in some of the modules' functions.

Table 1.1 MalasakitResponse Object

Attributes Name	Type	Description
response_id	Integer	A number indicating a response's order in the data (row number).
response	String	A string containing a response.
tag	String	A string indicating a response's category.
fspost_output	Tuple	Filipino Stanford Part-of-Speech Tagger (word, tag) tuple output.
fspost_stanford_format	String	Filipino Stanford Part-of-Speech Tagger word|tag Stanford notation.
pos	String	Filipino Stanford Part-of-Speech Tagger 'tags only' string.
insights_phrase	List	List of insights extracted from a response.
insights_words	List	List of lists of words (action, target, ...) insights extracted from a response.
location	String	String holder for a response's location (can be added by users).
language	String	Language identifier of the response (e.g., tl = Tagalog, en = English).

Data Utilities module (see Table 2.1) contains 10 functions. These functions enable the user to process files outside the program. Main functionalities permit the user to read and write on text or excel files.

Normalization module (see Table 2.2) contains 10 functions, that could be used to utilize or customize the normalizer. There are two normalizers that can be used in this API, namely Nocon, Kho, and Arroyo's (2018) normalizer and another that joins prefixes with root words using Oco and Borra's (2011) resource. As default, both are used in normalizing. There are two ways that normalizations could be done, one is per string or sentence, and another through a list or by batch. The rest of the functions act as means to modify the configuration, that is by setting the file path.

Language Identification module (see Table 2.3) contains 4 functions. These can be used to indicate the language a particular string is under. There are two ways it could be used, one is using a string as input (per sentence evaluation), and another using string or object lists (by batch evaluation). Either way, its output provides a tuple of language and confidence value. Additional functionality enables the user to modify the coverage of languages for identification.

The Filipino Part-of-Speech Tagger module (see Table 2.4) contains 6 functions, that makes use of FSPOST (Go & Nocon, 2017). Three kinds of functions can be seen in the list, which are about modifying the file path (to call the tagger), tagging options (per string, by object or string list), and formatting options (Part-of-Speech only or Stanford word|tag Format). By default, tagging format displays a word and Part-of-Speech tuple.

Information Extraction module (see Table 2.5) contains only 2 functions. These functions enable the user to extract insights in two formats. One is insight phrases which extracts a string starting from a Verb up to a Noun. Another does the same process but formats it with only the Verb and Nouns inside a sub-list or tuple. It is safe to note that on this module, the extractions were made specific to Malasakit responses (dataset used in research). Extractions are performed in an object with the following attributes: Response ID, the Response itself, its Response Category, Language identifier, FSPOST tags, container for the extracted insights, and location (a field that can be added by users).

Information Organization module (see Table 2.6) contains 3 functions that sets the formatting style of the clusters and report. Main functionalities enable the user to organize their extractions or results all in all or uses a per category style of formatting.

Information Clustering module (see Table 2.7) contains 11 functions. Three parts can be taken from this list. First is the main function that invokes the clustering algorithm (i.e., Dice's Coefficient, Word2Vec, or FastText). Then, supporting functions that can retrieve insights from the Malasakit object, remove duplicates in clusters, flatten insights in the cluster, and cluster/lexicalize target/noun words. Last is a list of functions that computes for distance or similarity values between two strings using the selected approach.

Information Ranking module (see Table 2.8) contains only 2 functions. A function that ranks clusters by frequency and arranges them in descending order (highest count first, lowest comes last); and a function that ranks by cluster categories, arranging the order of categories (category prioritization has been determined beforehand) and per categories ranks them by frequency in descending order.

Report Generation module (see Table 2.9) contains 6 functions. Its main function generates the report in a Microsoft Word document. The other five are functions that was used as support in generating the report. Examples for this includes adding a timestamp, divider, title, and setting the document margins and page columns. The idea for separating or having these functions is for future applications that intends to create their own format in the report.

As a whole, the functions listed are intended to be useful in future researches or application that involves the tasks of data processing, information extraction, language identification, part-of-speech tagging, word clustering, and text ranking.

2 Functions List and Sample Usages

Table 2.1 Data Utility Module Functions

Function Name	Description	Arguments	Return
refresh_excel	Clears out values in excel, excluding values under a protected_cell variable.	filename (str): The file location of the spreadsheet. protected_cells (int): The protected cells or number of columns (e.g., 2 columns → response & tag).	Void (save changes on Excel file)
read_candidate_excel	Reads the values in the candidate's excel file and stores the value in a list	filename (str): The file location of the spreadsheet.	phrase_list: a list of strings containing the system extracted (phrases) information. word_set_list: a list of strings containing the system extracted (word sets) information. total_words: a value indicating the total number of words in the input sentence / whole corpus.
read_gold_standard_excel	Reads the values in the gold standard's excel file and stores the value in a list.	filename (str): The file location of the spreadsheet. sheet_num (int): The sheet number to be read (Sheet 2 is designated for insight phrases and Sheet 3 is for word sets).	goldstandard_phrases_list: a list of strings containing the gold standard (phrases) information. goldstandard_word_sets_list: a list of strings containing the gold standard (word sets) information.
read_excel	Reads the values in an excel file and stores the first two columns (response and tag) into the MalasakitResponse object.	filename (str): The file location of the spreadsheet.	malasakit_response_list: a list of strings containing the Malasakit responses and their respective tags.
write_excel	Writes the system output in an excel file.	filename (str): The file location of the spreadsheet. malasakit_response_list (list): The list containing MalasakitResponse objects.	Void (save changes on Excel file)

Function Name	Description	Arguments	Return
		clusters_list (list): The list containing clustered information that was extracted from Malasakit. ranked_clusters_list (list): The ranked version of the list containing clustered information.	
text_to_list	Transforms a given text file into a list of list [s1 [w1, w2, ..., wN], ..., sN [w1, w2, ..., wN]].	filename (str): The file location of the text file.	sentence_list: a list of strings containing sentences found on the file.
text_to_list_without_stopwords	Transforms a given text file into a list of lists removing Tagalog/English stopwords in the process.	filename (str): The file location of the text file.	sentence_list: a list of strings containing sentences found on the file (without stopwords).
write_text_file	Writes the strings in a list to a text file.	filename (str): The file location of the text file. string_list: The list to be written in the text file.	Void (save changes on text file)
read_text_file	Reads the strings in a file and transfer them into a list	filename (str): The file location of the text file.	string_list: a list containing the contents of the text file.
get_stopwords_from_file	Transforms stop words in a given file into a list.	filename (str): The file location of the text file.	stopwords_list: a list containing the stopwords found on the text file.

Sample Code

```
# Task: Show changes from original input text to its 'no stopwords' counterpart.
import data_utils
input_file = 'test/in'  # File with input sentences
string_list_1 = data_utils.text_to_list(input_file)  # Get tokenized sentences from input
string_list_2 = data_utils.text_to_list_without_stopwords(input_file)
flattened_string_list_1 = [" ".join(sublist) for sublist in string_list_1]  # Join tokens into regular sentences
flattened_string_list_2 = [" ".join(sublist) for sublist in string_list_2]
combined_list = []  # Join the lists
for fsl1, fsl2 in zip(flattened_string_list_1, flattened_string_list_2):
    combined_list.append(fsl1 + ' --> ' + fsl2)
data_utils.write_text_file('test/sample_code_result.txt', combined_list)  # Display using write feature
```

Table 2.2 Normalization Module Functions

Function Name	Description	Arguments	Return Type
normalize_object	Normalizes the MalasakitResponse object and updates the object after.	malasakit_response_list (list): The list containing MalasakitResponse objects with responses to be normalized.	Void (updates MalasakitResponse Object)
normalize_list	Normalizes a given list of strings and returns the normalized list.	string_list (list): The list of strings to be normalized.	normalized_string_list: normalized version of the input string list.
normalize_string	Normalizes a given string and returns the normalized string.	string (string): The string to be normalized.	normalized_string: normalized version of the input string.
join_prefix_word	Joins the prefixes that are separated with a word by whitespace. More prefixes can be added in the tl_prefixes.txt file.	string_list (list): The list of string to be processed.	joined_prefix_word_string_list: a list of string with joined prefix-word modifications.
translate_filipino_colloquialism	Runs the Filipino Colloquialism Translator or Normalizer through command prompt. The file path parameters can be changed by the user.	moses_file (str): The file location of Moses executable file. model_file (str): The file location of the normalizer model. input_file (str): The file location of the input text (source - to be normalized). output_file (str): The file location of the output text (target - normalized version of the input).	Void (updates out file)
set_moses_file_path	Sets Moses' executable file path with the one provided by the user	filename (str): The file location of the Moses executable file.	Void (updates file location)
set_model_file_path	Sets Moses' model configuration file path with the one provided by the user	filename (str): The file location of the normalizer model.	Void (updates file location)
set_input_file_path	Sets Moses' input text file path with the one provided by the user	filename (str): The file location of the input text.	Void (updates file location)
set_output_file_path	Sets Moses' output text file path with the one provided by the user	filename (str): The file location of the output text.	Void (updates file location)
set_prefix_file_path	Sets a user-provided prefix list text file.	filename (str): The file location of the prefix list.	Void (updates file location)

Sample Code

```
# Task: Normalize a list of irregular strings.
import normalize
string_list = ['cge n nga', 'sn n kyo?', 'bro, g k b mmya mag laro?', 'edewups na mudrakels ko sa trip']
print(normalize.normalize_list(string_list))

Output:
['sige na nga', 'saan na kyo?', 'bro, game ka ba mamaya maglaro?', 'puwede na nanay ko sa trip']
```

Table 2.3 Language Identification Module Functions

Function Name	Description	Arguments	Return Type
set_language	Changes the scope of languages.	code_list (list): List of languages to be considered. Must follow ISO 639-1 code.	Void (updates language scope)
identify_language_string	Identifies language (ISO 639-1 code) of a given string. Returns a (language, confidence) tuple.	sentence (str): The sentence to be language identified.	language: a tuple consisting of (language, confidence) fields.
identify_language_object_list	Identifies language ('tl' = Tagalog or 'en' = English) of MalasakitResponse object's responses and updates the language field in the object.	malasakit_response_list (list): The list containing MalasakitResponse objects.	Void (updates MalasakitResponse Object)
identify_language_string_list	Identifies language of sentences in a list. Returns a list of languages respective to the sentences.	sentence_list (list): The list of strings to be language identified.	language_identified_list: a list containing the language result.

Sample Code

```
# Task: Identify languages of strings in a list, set language code limits, and identify again.
import lang_id
string_list = ["I love you", "我爱你", "愛してる", "사랑해", "Σ΄αγαπώ", "Ich liebe Dich"]
print(lang_id.identify_language_string_list(string_list))
lang_id.set_language(['en', 'zh', 'ja'])
print(lang_id.identify_language_string_list(string_list))

Output:
[('en', 9.061840057373047), ('zh', -40.14874768257141), ('ja', -105.30295419692993), ('ko', -54.02418255805969),
('el', -85.62207746505737), ('de', -24.075527667999268)]
[('en', 9.061840057373047), ('zh', -40.14874768257141), ('ja', -105.30295419692993), ('ja', -98.81557035446167),
('zh', -155.25258898735046), ('en', -28.10593557357788)]
```

Table 2.4 FSPOST Module Functions

Function Name	Description	Arguments	Return Type
set_java_path	Sets the java path to make Stanford POS Tagger work.	file_path (str): The java file path / location.	Void (updates file location)
tag_string	Tags a sentence/string. Returns a (word, pos) tuple.	sentence (str): The string to be tagged.	tagged_string: a list of string tokens containing POS labeled (word, pos) tuples.
tag_object_list	Tagging a list of MalasakitResponse object's sentence. This updates the MalasakitResponse object.	malasakit_response_list (list): The list containing the MalasakitResponse objects.	Void (updates MalasakitResponse Object)
tag_string_list	Tags a list of sentences. Returns a list of (word, pos) tuple.	sentence_list (list): The list of strings to be tagged.	tagged_list: a list of strings containing POS labelled (word, pos) tuples.
format_pos	Formats a tuple into a POS-only string.	input_tuple (tuple): The tuple to be formatted.	tagged_string: a string containing POS labels.
format_stanford	Formats a tuple into Stanford word|tag string.	input_tuple (tuple): The tuple to be formatted.	tagged_string: a string containing POS labels in Stanford's word|tag notation.

Sample Code

```
# Task: Tag a string and change its formatting.
import fspost

fspost.set_java_path("")  # Empty string for default tagger model path
tagged_string = fspost.tag_string('ako nalang ba dito o sasamahan ako ni Nicco ?')

print("Tuple: ", tagged_string)
print("Stanford: ", fspost.format_stanford(tagged_string))
print("POS: ", fspost.format_pos(tagged_string))

Output:
Tuple: [('ako', 'PRS'), ('nalang', 'VBTS_CCP'), ('ba', 'RBI'), ('dito', 'PRL'), ('o', 'CCT'), ('sasamahan', 'VBOF'),
('ako', 'PRS'), ('ni', 'DTP'), ('Nicco', 'NNP'), ('?', 'PMQ')]
Stanford: ako|PRS nalang|VBTS_CCP ba|RBI dito|PRL o|CCT sasamahan|VBOF ako|PRS ni|DTP Nicco|NNP ?|PMQ
POS: PRS VBTS_CCP RBI PRL CCT VBOF PRS DTP NNP PMQ
```

Table 2.5 Information Extraction Module Functions

Function Name	Description	Arguments	Return Type
extract_insights_phrases	Extracts phrase insights (action word to target/s). The MalasakitResponse object is updated after.	malasakit_response_list (list): The list containing MalasakitResponse objects with responses to be extracted.	Void (updates MalasakitResponse Object)
extract_insights_words	Extracts word insights or word sets (action word and target/s). The MalasakitResponse object is updated after.	malasakit_response_list (list): The list containing MalasakitResponse objects with responses to be extracted.	Void (updates MalasakitResponse Object)

Sample Code

```
# Task: Build a MalasakitResponse object.
import malasakit_response
import fspost
import lang_id
import extract
fspost.set_java_path("")

response = 'Maglinis ng mga kanal at kalye o itapon ang mga basura'
response_object = malasakit_response.MalasakitResponse(1, response, 'Sanitation')
response_object.fspost_output = fspost.tag_string(response_object.response)
response_object.fspost_stanford_format = fspost.format_stanford(response_object.fspost_output)
response_object.pos = fspost.format_pos(response_object.fspost_output)
response_object.location = 'Manila, Philippines'
response_object.language = lang_id.identify_language_string(response_object.response)[0]

extract.extract_insights_phrases([response_object])  # Extraction
extract.extract_insights_words([response_object])

print('Response ID: ', response_object.response_id)
print('Response: ', response_object.response)
print('Category: ', response_object.tag)
print('FSPOST Tuple: ', response_object.fspost_output)
print('FSPOST Stanford: ', response_object.fspost_stanford_format)
print('FSPOST POS only: ', response_object.pos)
print('Insight Phrases: ', response_object.insights_phrase)
```

```
print('Insight Word Sets: ', response_object.insights_words)
print('Location: ', response_object.location)
print('Language: ', response_object.language)

Output:
Response ID: 1
Response: Maglinis ng mga kanal at kalye o itapon ang mga basura
Category: Sanitation
FSPOST Tuple: [('Maglinis', 'VBW'), ('ng', 'CCB'), ('mga', 'DTCP'), ('kanal', 'NNC'), ('at', 'CCA'), ('kalye',
'NNC'), ('o', 'CCT'), ('itapon', 'VBTF'), ('ang', 'DTC'), ('mga', 'DTCP'), ('basura', 'NNC')]
FSPOST Stanford: Maglinis|VBW ng|CCB mga|DTCP kanal|NNC at|CCA kalye|NNC o|CCT itapon|VBTF ang|DTC mga|DTCP
basura|NNC
FSPOST POS only: VBW CCB DTCP NNC CCA NNC CCT VBTF DTC DTCP NNC
Insight Phrases: [1, 'Maglinis ng mga kanal at kalye', 'itapon ang mga basura']
Insight Word Sets: [[1, 'Maglinis', 'kanal', 'kalye'], [1, 'itapon', 'basura']]
Location: Manila, Philippines
Language: tl
```

Table 2.6 Information Organization Module Functions

Function Name	Description	Arguments	Return Type
organize_sublist	Organizes a given sublist (a single category or the current category).	sub_malasakit_response_list_copy (list): The list containing a subset of MalasakitResponse objects. clusters_list (list): The list of clustered responses. ranked_clusters_list (list): The list of clustered and ranked responses. clustering_technique (str): Select a clustering technique from the following: 'dice', 'word2vec', or 'fasttext'. current_category (str): The label of the current category being processed. clustering_time (float): The variable for tracking the execution time of the clustering process. ranking_time (float): The variable for tracking the execution time of the ranking process.	clusters_list: a list containing clustered responses. ranked_clusters_list: a list containing clustered and ranked responses. clustering_time: a float variable for tracking the execution time of the clustering process. ranking_time: a float variable for tracking the execution time of the ranking process.
organize_by_response_categories	Organizes the information based on (or per) response categories.	malasakit_response_list_copy (list): The list containing MalasakitResponse objects. clustering_technique (str): Select a clustering technique from the following: 'dice', 'word2vec', or 'fasttext'. priority_categories (list): [] if prioritization of categories will use the default, otherwise pass a list.	clusters_list: a list containing clustered responses. ranked_clusters_list: a list containing clustered and ranked responses.
organize_all_entries	Organizes the entire information (per entry).	malasakit_response_list_copy (list): The list containing MalasakitResponse objects. clustering_technique (str): Select a clustering technique from the following: 'dice', 'word2vec', or 'fasttext'.	clusters_list: a list containing clustered responses. ranked_clusters_list: a list containing clustered and ranked responses.

Sample Code

```
# Task: Organize and cluster a set of insights.
import malasakit_response
import organize
response_object_1 = malasakit_response.MalasakitResponse(1, 'response1', 'tag1')
response_object_1.insights_words = [[1, 'maglinis', 'kanal', 'kalye'], [1, 'itapon', 'basura']]
response_object_2 = malasakit_response.MalasakitResponse(2, 'response2', 'tag2')
response_object_2.insights_words = [[2, 'magkaisa', 'tao']]
response_object_3 = malasakit_response.MalasakitResponse(3, 'response3', 'tag3')
response_object_3.insights_words = [[3, 'magkaroon', 'komunikasyon']]
response_object_4 = malasakit_response.MalasakitResponse(4, 'response4', 'tag4')
response_object_4.insights_words = [[4, 'wastong', 'pagtatapon'], [4, 'bantayan', 'gamit']]
response_object_5 = malasakit_response.MalasakitResponse(5, 'response5', 'tag5')
response_object_5.insights_words = [[5, 'dumating', 'pagkain'], [5, 'magkaroon', 'equipment'], [5, 'maglinis',
'kalsada']]
response_list = [response_object_1, response_object_2, response_object_3, response_object_4, response_object_5]
cluster_list, ranked_cluster_list = organize.organize_all_entries(response_list, 'dice')
print(cluster_list)
print(ranked_cluster_list)

Output:
[['1|5', 2, 'maglinis', 'kanal, kalye, kalsada'], ['1', 1, 'itapon', 'basura'], ['2', 1, 'magkaisa', 'tao'],
['3|5', 2, 'magkaroon', 'komunikasyon, equipment'], ['4', 1, 'wastong', 'pagtatapon'], ['4', 1, 'bantayan',
'gamit'], ['5', 1, 'dumating', 'pagkain']]
[['1|5', 2, 'maglinis', 'kanal, kalye, kalsada'], ['3|5', 2, 'magkaroon', 'komunikasyon, equipment'], ['1', 1,
'itapon', 'basura'], ['2', 1, 'magkaisa', 'tao'], ['4', 1, 'wastong', 'pagtatapon'], ['4', 1, 'bantayan', 'gamit'],
['5', 1, 'dumating', 'pagkain']]
```

Table 2.7 Information Clustering Module Functions

Function Name	Description	Arguments	Return Type
string_similarity_fasttext	Computes FastText's vector similarity (how close) between two strings.	string1 (str): The string to be compared to. string2 (str): The string to be compared to.	similarity: resulting score of pairs based on how close they are from each other (higher value is better).
string_distance_fasttext	Computes FastText's vector distance (how far) between two strings.	string1 (str): The string to be compared to. string2 (str): The string to be compared to.	distance: resulting score of pairs based on how far they are from each other (lower value is better).
string_similarity_word2vec	Computes Word2Vec's vector similarity (how close) between two strings.	string1 (str): The string to be compared to. string2 (str): The string to be compared to.	similarity: resulting score of pairs based on how close they are from each other (higher value is better).
string_distance_word2vec	Computes Word2Vec's vector distance (how far) between two strings.	string1 (str): The string to be compared to. string2 (str): The string to be compared to.	distance: resulting score of pairs based on how far they are from each other (lower value is better).
string_similarity_dice	Computes Dice's Coefficient similarity (how close) between two strings.	string1 (str): The string to be compared to. string2 (str): The string to be compared to.	similarity: resulting score of pairs based on how close they are from each other (higher value is better).
string_distance_dice	Computes Dice's Coefficient distance (how far) between two strings.	string1 (str): The string to be compared to. string2 (str): The string to be compared to.	distance: resulting score of pairs based on how far they are from each other (lower value is better).
collect_all_insights_from_object	Retrieves all insights in the MalasakitResponse object and stores them in one list.	malasakit_response_list (list): The list containing MalasakitResponse objects. insights_type (str): A character/string indicating the type of insights to be collected. 'p' for phrases and 'w' for word sets.	insights_list: a list containing all insights taken from the object list.

Function Name	Description	Arguments	Return Type
merge_cluster_insights	Merges the insights in one cluster into a single line.	cluster (list): The list containing the current cluster. clustering_technique (str): Select a clustering technique from the following: 'dice', 'word2vec', or 'fasttext'.	cluster: a list containing modified (merged) words in the cluster's insights.
remove_duplicate	Removes duplicate strings in the list (cluster).	cluster_zero (list): The list containing the current cluster.	filtered_cluster_zero: a list containing the modified (filtered-off duplicates) words in the cluster.
cluster_words	Clusters target/noun words. Given a list it will join similar words using the 'word1 (word2, ..., wordN)' notation.	target_word_list (list): The list containing the words to be clustered. clustering_technique (str): Select a clustering technique from the following: 'dice', 'word2vec', or 'fasttext'.	new_target_word_list: a list containing the clustered and formatted words.
cluster_information	Clusters text using either Sørensen-Dice Coefficient (String Clustering), Word2Vec, or FastText Word Embeddings (Semantic Clustering) and returns a list of clusters.	malasakit_response_list (list): The list containing the MalasakitResponse objects. clustering_technique (str): Select a clustering technique from the following: 'dice', 'word2vec', or 'fasttext'.	clusters_list: a list containing the clustered insights.

Sample Code

```
# Task: Compare two words using the three clustering approaches.
import cluster
string1 = 'malinis'
string2 = 'kalinisan'
print('Dice:', cluster.string_similarity_dice(string1, string2))
try:
    similarity = cluster.string_similarity_word2vec(string1, string2)
except KeyError:
```

```
    similarity = 0.0  # No operation done if not on the model, so set similarity to 0
print('Word2Vec:', similarity)
try:
    similarity = cluster.string_similarity_fasttext(string1, string2)
except KeyError:
    similarity = 0.0  # No operation done if not on the model, so set similarity to 0
print('FastText:', similarity)

Output:
Dice: 0.6666666666666666
Word2Vec: 0.7630582
FastText: 0.623511
```

Table 2.8 Information Ranking Module Functions

Function Name	Description	Arguments	Return Type
rank_clusters_by_frequency	Ranks the clusters based on their frequency counts (descending order: highest count first).	clusters_list (list): The list containing the clustered insights.	updated_clusters_list: a list containing the ranked clusters.
rank_by_response_categories	Ranks the clusters (rearranges the groups) based on their response category prioritization order. Categories follow the Malasakit Codebook 4.7.	clusters_list (list): The list containing the clustered insights. priority_categories (list): [] if prioritization of categories will use the default, otherwise pass a list.	updated_clusters_list: a list containing the ranked clusters.

Sample Code

```
# Task: Rank a given list of clusters.
import rank
cluster_list = [['1|5', 2, 'maglinis', 'kanal, kalye, kalsada'], ['1', 1, 'itapon', 'basura'],
                ['2', 1, 'magkaisa', 'tao'], ['3|5', 2, 'magkaroon', 'komunikasyon, equipment'],
                ['4', 1, 'wastong', 'pagtatapon'], ['4', 1, 'bantayan', 'gamit'], ['5', 1, 'dumating', 'pagkain']]
print(rank.rank_clusters_by_frequency(cluster_list))

Output:
[['1|5', 2, 'maglinis', 'kanal, kalye, kalsada'], ['3|5', 2, 'magkaroon', 'komunikasyon, equipment'], ['1', 1,
'itapon', 'basura'], ['2', 1, 'magkaisa', 'tao'], ['4', 1, 'wastong', 'pagtatapon'], ['4', 1, 'bantayan', 'gamit'],
['5', 1, 'dumating', 'pagkain']]
```

Table 2.9 Report Generation Module Functions

Function Name	Description	Arguments	Return Type
add_timestamp	Adds a timestamp in the document. Follows Month-Date-Year Hours-Minutes-Seconds format (e.g., Oct-02-2019 18:51:50).	document (object): The instance of the document to be edited.	Void (updates the file)
add_divider	Adds a divider in the document that is made from a 1x1 table object.	document (object): The instance of the document to be edited.	Void (updates the file)
add_title	Adds the title in the document and formats it for display.	document (object): The instance of the document to be edited. title_text (str): The string input for the title label.	Void (updates the file)
set_document_margin	Sets the document's margin (in inches).	document (object): The instance of the document to be edited. section_number (int): The section to be changed. -1 is for the first section. top (int): The number to be set on the top margin. bottom (int): The number to be set on the bottom margin. left (int): The number to be set on the left margin. right (int): The number to be set on the right margin.	Void (updates the file)
set_number_of_page_columns	Sets the number of columns in a section through xpath.	section (object): The section to be modified. columns (int): The number of columns to be applied.	Void (updates the file)
write_report	Generates the report in word document (.docx) format. Default filename: Report.docx	filename (str): The file path of the word document. malasakit_response_list (list): The list containing the MalasakitResponse objects. ranked_clusters_list: The list containing the clustered and ranked insights.	

Sample Code

```
# Task: Generate a customized report.
import generate
from docx import Document
document = Document()  # Create an instance of the document
# Use API functions
generate.set_document_margin(document, -1, 2, 2, 2, 2)
generate.add_divider(document)  # Divider
generate.add_title(document, 'THIS IS MY FIRST CUSTOMIZED REPORT')  # Title text
generate.add_divider(document)  # Divider
generate.add_timestamp(document)  # Timestamp
# Carry on with basic document function/s
p = document.add_paragraph("This is a paragraph.")
p.add_run(' In ')
p.add_run('BOLD').bold = True
p.add_run(' and ')
p.add_run('italic.').italic = True
table = document.add_table(rows=1, cols=3)
hdr_cells = table.rows[0].cells
hdr_cells[0].text = 'Item'
hdr_cells[1].text = 'Quantity'
hdr_cells[2].text = 'Description'
table_elements = (
    ('Food', '350,000', 'Contains canned goods, vegetables, eggs, and meat.'),
    ('Medicine', '93,500', 'Includes vaccines, herbs, and prescription drugs.'),
    ('Clothes', '51,926', 'Donations consisting of shirt, pants, and underwear.')
)
for item, quantity, description in table_elements:
    row_cells = table.add_row().cells
    row_cells[0].text = str(item)
    row_cells[1].text = quantity
    row_cells[2].text = description
p = document.add_paragraph("Additional document functions at: \n")
p.add_run("https://python-docx.readthedocs.io/en/latest/").bold = True
document.save('test/My First Report.docx')
```

THIS IS MY FIRST CUSTOMIZED REPORT

May-20-2020 21:42:23

This is a paragraph. In **BOLD** and *italic*.

Item	Quantity	Description
Food	350,000	Contains canned goods, vegetables, eggs, and meat.
Medicine	93,500	Includes vaccines, herbs, and prescription drugs.
Clothes	51,926	Donations consisting of shirt, pants, and underwear.

Additional document functions at:
https://python-docx.readthedocs.io/en/latest/

Sample Code Output

3 Full Run and Evaluation

Included in the API are modules for running the entire tasks as indicated in the Architectural Diagram, and quantitatively evaluating two files (candidate and gold standard). There are two main parts in running the analysis tool, namely configurations and software modules. In configurations, there are a set of variables that could be modified by the user (see Table 3.1 for the list).

Table 3.1 Main Module Functions

Variable	Default Value	Description
java_path	''	Path for Java compiled Stanford POS Tagger (empty string used to indicate default value).
malasakit_filename	'test/MalasakitResponses.xlsx'	File path of input dataset.
report_filename	'test/Report.docx'	File path and filename of generated report.
protected_cells	2	Untouched cells in input dataset.
organization_type	''	Organization style for clustering, empty string for organize all entries and 'categories' for organize by response categories.
clustering_technique	'dice'	Technique used for clustering (keywords: 'dice', 'word2vec', 'fasttext')
priority_categories	[]	Order of priority in response categories. Users can modify the order and title of categories.

In software modules part, this follows the main tasks involved in analyzing the texts. The tasks are written in order with their corresponding function calls intact. In each task, there are system prints that indicate its beginning and end. After each task and at the end of the program, its runtime and total execution time are displayed, respectively. A sample progress display of this run is shown at Figure 3.1 and sample outputs are shown at Figure 3.2 to 3.8.

Figure 3.1 Progress Display

FILIPINO TEXT ANALYSIS TOOL REPORT

Mar-04-2020 19:40:07

The information below were extracted and organized automatically.

Entry 1

ID/s:
26|28|31|39|58|69|73|90|148|151|167|203|204|208|209|211|306|339|341|342|366|436|440|448|455|495|506|507|510|529|554|576|580|612|629|643|649|673|732|743|757|762|766|767|777|791|793|811|812|813|821|836|850|851|852|853|854|885|888|890|891|892|894|896|899|900|901|902|922|923

Frequency: 80

Proposed action: dapat, sapat

Target: barangay (baragay, baranggay), active, pakiki-isa, komunidad, sakuna, eg, komunikasyon, kita, kalinisan, mamamayan, disaster, balita, evacuation, center, aware, pamamagitan, seminar, oras, anunsiyo, prepared, updated (update), kanal, pangongolekta, lugar, alert, tao, araw, kalamidad, alirto, radio, abiso, saknila, posters, paalala, official, ugnayan, membro, i, lage, panahon, drill, like, for, example, about, paraan, truck, encourage, pagbabahahindi, dn, before, during, and, after, of, the, calamity, my, weekly, paligid, mg, roong, mgkaisa, ulat, darating, pra, programa (program), pagkain, pangangailangan

Entry 2

ID/s:
6|7|13|16|19|41|46|122|128|208|236|243|307|311|341|365|419|439|443|447|457|471|502|513|557|598|603|611|616|618|621|626|631|651|660|690|714|721|728|731|747|748|753|758|760|766|783|785|786|806|815|817|820|823|824|830|833|854|889|903|925

Frequency: 63

Proposed action: maging, palaging, messaging, laging, pagiging, magiging, nagiging

Target: tao, kapitbahay, bagay, bagyobaha (bagyo), darating (daratinga), disaster, mamamayan, aware, oras, sakuna, paglaki, pra, kalagayan, alertoat, barangay (kabarangay), tagapag, handa, pamaraan, kalamidad (kalamid), agrisibo, kabaro, atentibo, balita, cla, brgys, pkiking, beforeafter, and, during, kalikasan, kagamatin, opisyal, sanhi, kasanyan, xa, trahedya, media, etc, gawim, paalala, prepared, dahilan

Entry 3

ID/s:
80|112|121|123|125|132|137|139|140|145|146|149|153|154|156|165|169|175|192|195|198|199|215|222|233|234|235|248|285|292|308|309|318|340|371|392|575|585|619|654|755|794|795|816|831|859|861|863|868|909|928|929|932|934

Frequency: 56

Proposed action: be

Target: advantage, training, possibilities, news, plenty, calamities, community, beforehand, incoming, disaster (disasters), officials, time (times), dont, publicsoundnotifsystem, happenings, typhoon, aware, garbage, evacuation, plan, drills, things, preparedness (prepare), programs, constituents, flood, devices, instructions, government, technology, need, orientation, effects, management, response, unit, barangay, events, posters, case, duty, info, society

Entry 4

ID/s:
3|6|7|8|11|14|20|27|33|35|40|44|50|66|68|209|289|307|310|311|321|339|341|368|467|478|488|550|555|696|716|752|762|779|787|800|847|885|926|930

Frequency: 42

Extracting and Organizing Disaster-related Philippine Community Responses for Aiding Nationwide Risk Reduction Planning and Response (N. Nocon, 2019)

Figure 3.2 Report Organize All Entries

FILIPINO TEXT ANALYSIS TOOL REPORT

Mar-24-2020 20:24:54

The information below were extracted and organized automatically.

INFORMATION CAMPAIGN AND CAPACITY BUILDING

Entry 1

ID/s:
26|208|209|339|507|576|580|649|762|766|811|813|836|852|854|885

Frequency: 20

Proposed action: dapat

Target: barangay, seminar, oras, prepared, kalamidad, sakuna, posters, paalala, drill, like, for, example, about, paraan, encourage, before, during, and, after, of, the, calamity, my, weekly, mg, roong

Entry 2

ID/s:
13|46|208|236|243|307|311|598|603|618|631|651|758|760|766|854|903|925

Frequency: 19

Proposed action: maging, palaging, laging

Target: kapitbahay, tao, disaster, mamamayan, aware, kabarangay, kabaro, kalamidad, darating, beforeafter, and, during, pra, bagyo, trahedya, gawim, paalala

Entry 3

ID/s:
33|68|209|289|307|310|311|339|696|716|762|787|800|847|885|926|930

Frequency: 19

Proposed action: magkaroon, magkaron, magkakaroon, karoon

Target: seminars (seminar, seminarsdrill), about, kaalaman (kaalam), disaster, drill, sos, program, regarding, and, iinvite, emergency, kits, training, progma, pra, disipline, mamamayan, weekly, meeting, organisasyon, orientation, barangay

Entry 4

ID/s:
80|121|132|149|156|169|215|222|234|340|585|654|794|831|929

Frequency: 15

Proposed action: be

Target: advantage, possibilities, calamities, dont, times (time), drills, typhoon, disaster, preparedness (prepare), programs, officials, instructions, orientation, effects, duty

Entry 5

ID/s:
130|215|245|249|259|266|340|390|794|838

Frequency: 12

Proposed action: have

Target: assembly, disaster, drill, place, representative, check, emergency, needs, training (taraining), barangay, officials, seminar, meeting

Entry 6

ID/s:
169|222|303|331|343|358|385|614|627|654|661

Frequency: 11

Proposed action: help

Target: prepare, families, disasters (disaster), outcomes, community, idea, seminars, people, signage, calamity

Entry 7

ID/s:
101|104|169|287|333|343|347|396|583|644

Frequency: 10

Proposed action: preparing, prepare

Target: typhoon, disaster, seminar, times

Entry 8

Extracting and Organizing Disaster-related Philippine Community Responses for Aiding Nationwide Risk Reduction Planning and Response (N. Nocon, 2019)

Figure 3.3 Report Organize by Category

	A	B	C	D
1	magkaisa dapat ang mga tao	Filipino values	tl	magkaisa dapat ang mga tao
2	mag karoon ng pagkakaiisa upang sa mga (	Filipino values	tl	magkaroon ng pagkakaiisa upang sa mga darating na mga sakuna ay malalagpasan
3	magkaroon ng komunikasyon kung saan m	early warning system	tl	magkaroon ng komunikasyon kung saan magkikita sa panahon ng kalamaidad
4	paglilinis ng kanal wastong pagtatapon ng l	infrastructure maintenance and management	tl	paglilinis ng kanal wastong pagtatapon ng basura at kailangan magikot ikot ang mga ta
5	malawakang information drive	information campaign and capacity building	tl	malawakang information drive
6	bago dumating ang bagyo magkaroon ng e	early warning system	tl	bago dumating ang bagyo magkaroon ng early warning system para mas maging hanc
7	magkaroon ng early warning upang magin	early warning system	tl	magkaroon ng early warning upang maging handa ang mga tao sa darating na bagyo
8	higit na pagtibayin ang early warning syste	early warning system	tl	higit na pagtibayin ang early warning system device magkaroon ng maintenance quart
9	lalo pang lumawat at lumago ang pagmam	Filipino values	tl	lalo pang lumawat at lumago ang pagmamalasakit sa aming kabarangay
10	pagsunod sa sinasabi sa kung ano ang dap	preparedness for emergency	tl	pagsunod sa sinasabi sa kung ano ang dapat gawin paghandaan ang lahat ng bibitbitin
11	pagkakaroon ng early warning device	early warning system	tl	pagkakaroon ng early warning device
12	pagbibigay ng humanitarian assistance goo	disaster relief	tl	pagbibigay ng humanitarian assistance goods sa panahon ng kalamidad
13	nais ko po sana magkaroon pa po ngmga ib	information campaign and capacity building	tl	nais ko po sana magkaroon pa po ngmga ibat ibang paraan upang lalo pang maging ha
14	siguruhing magkaroon ng mga basurahan s	infrastructure maintenance and management	tl	siguruhing magkaroon ng mga basurahan sa buong barangay dahil ito ang pangunahir

Figure 3.4 Excel Output (Responses, Categories, Language, and Normalized)

	E
1	magkaisa|VBW dapat|VBS ang|DTC mga|DTCP tao|NNC
2	magkaroon|VBAF ng|CCB pagkakaiisa|NNC upang|CCB sa|CCT mga|DTCP darating|NNC na|CCP mga|DTCP sakuna|NNC ay|LM malalagpasan|VBTF
3	magkaroon|VBAF ng|CCB komunikasyon|NNC kung|RBK saan|RBQ magkikita|VBTF sa|CCT panahon|NNC ng|CCB kalamaidad|NNC
4	paglilinis|NNC ng|CCB kanal|NNC wastong|JJD_CCP pagtatapon|NNC ng|CCB basura|NNC at|CCA kailangan|VBS magikot|VBW ikot|NNC ang|DTC m
5	malawakang|JJD_CCP information|FW drive|FW
6	bago|RBW dumating|VBAF ang|DTC bagyo|NNC magkaroon|VBW ng|CCB early|FW warning|FW system|FW para|CCT mas|JJCC maging|VBW hand
7	magkaroon|VBAF ng|CCB early|FW warning|FW upang|CCB maging|VBW handa|JJD ang|DTC mga|DTCP tao|NNC sa|CCT darating|NNC na|CCP bag
8	higit|JJCC na|CCP pagtibayin|VBOF ang|DTC early|FW warning|FW system|FW device|FW magkaroon|VBAF ng|CCB maintenance|FW quarterly|FW
9	lalo|JJCC pang|RBI_CCP lumawat|VBAF at|CCA lumago|VBAF ang|DTC pagmamalasakit|NNC sa|CCT aming|PRSP_CCP kabarangay|NNC
10	pagsunod|NNC sa|CCT sinasabi|VBTR sa|CCT kung|RBK ano|RBQ ang|DTC dapat|VBS gawin|VBOF paghandaan|VBOF ang|DTC lahat|PRI ng|CCB bib
11	pagkakaroon|VBW ng|CCB early|FW warning|FW device|FW
12	pagbibigay|VBW ng|CCB humanitarian|VBTR assistance|FW goods|FW sa|CCT panahon|NNC ng|CCB kalamidad|NNC
13	nais|VBS ko|PRS po|RBS sana|VBS magkaroon|VBW pa|RBI po|RBS ngmga|NNC ibat|JJD ibang|PRI_CCP paraan|NNC upang|CCB lalo|JJCC pang|RBI
14	siguruhing|RBD_CCP magkaroon|VBW ng|CCB mga|DTCP basurahan|NNC sa|CCT buong|PRI_CCP barangay|NNC dahil|CCR ito|PRO ang|DTC pangur

Figure 3.5 Excel Output (Part-of-Speech Tagged)

	F	G	H	I
1	1	magkaisa dapat ang mga tao		
2	2	magkaroon ng pagkakaiisa		
3	3	magkaroon ng komunikasyon	magkikita sa panahon	
4	4	wastong pagtatapon	kailangan magikot ikot	bantayan mga gamit
5	5	malawakang information drive		
6	6	dumating ang bagyo	magkaroon ng early warning system	maging handa ang mga tao
7	7	magkaroon ng early warning	maging handa ang mga tao	
8	8	pagtibayin ang early warning system device	magkaroon ng maintenance quarterly	masigurong maayos ito bago dumating an
9	9	lumawat at lumago ang pagmamalasakit		
10	10	sinasabi sa kung ano ang dapat gawin paghandaan ang lahat ng bibitbitin sa tuwing may sakuna		
11	11	pagkakaroon ng early warning device		
12	12	pagbibigay ng humanitarian assistance goods		
13	13	nais ko po sana magkaroon pa po ngmga	ibat ibang paraan	maging handa ang aming mga kapitbahay
14	14	magkaroon ng mga basurahan	pangunahing sanhi	

Figure 3.6 Excel Output (Insight Phrases)

	A	B	C	D	E	F
1	1	magkaisa	tao			
2	2	magkaroon	pagkakaiisa			
3	3	magkaroon	komunikasyon			
4	3	magkikita	panahon			
5	4	wastong	pagtatapon			
6	4	kailangan	ikot			
7	4	bantayan	gamit			
8	5	malawakang	information	drive		
9	6	dumating	bagyo			
10	6	magkaroon	early	warning	system	
11	6	maging	tao			
12	7	magkaroon	early	warning		
13	7	maging	tao			
14	8	pagtibayin	early	warning	system	device
15	8	magkaroon	maintenance	quarterly		
16	8	masigurong	kalamidad			
17	9	lumawat	pagmamalasakit			
18	10	sinasabi	sakuna			
19	11	pagkakaroon	early	warning	device	
20	12	pagbibigay	assistance	goods		
21	13	nais	ngmga			
22	13	ibat	paraan			

Figure 3.7 Excel Output (Insight Word Sets / Verb-Noun Tuples)

	A	B	C	D	E	F
1	INFORMATION CAMPAIGN AND CAPACITY BUILDING					
2	Cluster 1					
3	13|46|208|236|243|307|311|598|603|6	19	maging, palaging, laging	kapitbahay, tao, disaster, mamamayan, awa		
4	Cluster 2					
5	26|208|209|339|507|576|649|762|766	18	dapat	barangay, seminar, oras, prepared, kalamida		
6	Cluster 3					
7	33|68|209|307|310|311|339|696|716|7	16	magkaroon, magkakaroon	seminars (seminar, seminarsdrill), about, kaa		
8	Cluster 4					
9	80|121|132|149|156|169|215|222|234	15	be	advantage, possibilities, calamities, barangay		
10	Cluster 5					
11	130|215|245|249|259|266|340|390|794	12	have	assembly, disaster, drill, place, representativ		
12	Cluster 6					
13	207|238|269|289|332|343|344|382|396	12	conducting, magconduct, conduct, pag	seminars (seminar), drills, community, assen		
14	Cluster 7					
15	101|104|169|287|333|343|347|396|583	10	preparing, prepare	typhoon, disaster, seminar, times		
16	Cluster 8					
17	169|222|303|331|343|358|614|627|654	10	help	prepare, families, disasters (disaster), outcor		
18	Cluster 9					
19	26|362|764|813|852|883|903|926	9	may	pangangailangan, seminar, meetings, kalami		
20	Cluster 10					
21	101|104|179|218|337|347|350|386|405	9	inform, informs, informing	consequence, people, neighbor, subordinate		
22	Cluster 11					
23	152|200|216|303|594|627|798|876|927	9	giving, living	disaster, drill, knowledge, seminars, leaflets,		
24	Cluster 12					
25	344|382|390|398|485|700|842|846|866	9	regarding	prevention, disaster (disasters), awareness, p		
26	Cluster 13					

Figure 3.8 Excel Output (Ranked Clusters)

Evaluating an output, two files must be provided: candidate (file to be compared) and gold standard (file to be based). These can be indicated in system output and gold standard variables. Under this module constitute functions that computes and compares the two files. Shown on Table 3.2 are the list of usable functions with their descriptions. It is important to note that running the module executes the necessary set of codes that outputs a summary of the evaluation results.

Given the two files, several statistics and standard metrics will be computed and displayed as output. The statistics for insight phrases consist of extraction counts, complete matches, over extractions, under extractions, overlapping extractions, and complete mismatches. On the other hand, statistics for word sets (verb-noun tuples) are extraction counts, exact matches, partial matches, action matches, target matches, crossover matches, and no matches. For both, standard metrics consists of Precision, Recall, Accuracy and F-Measure, computed through True Positive, False Positive, False Negative, and True Negative values. Adding to the display, members of some of these values are displayed on a list, where it can be reviewed.

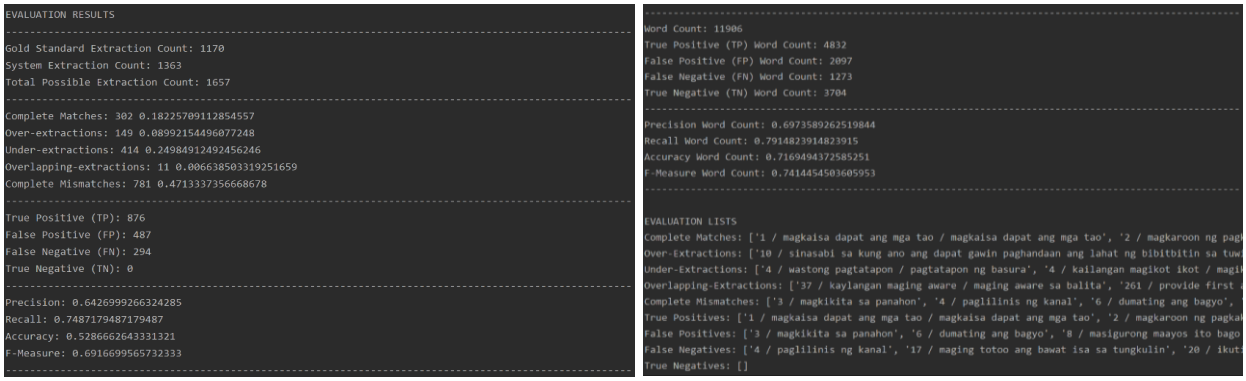

Figure 3.9 Evaluation Output (Insight Phrases)

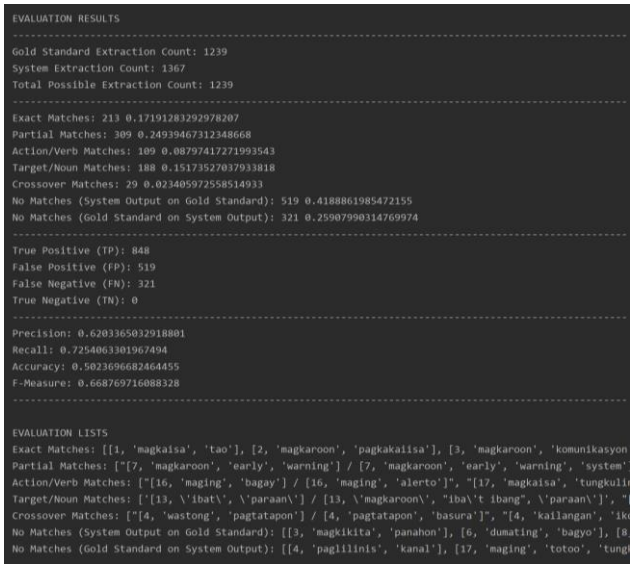

Figure 3.10 Evaluation Output (Insight Word Sets)

Table 3.2 Evaluation Module Functions

Function Name	Description	Arguments	Return Type
precision	Computes Precision	true_positive (int): System EXTRACTED a text that is an insight / in the Gold Standard. false_positive (int): System EXTRACTED a text that is not an insight / not in the Gold Standard.	precision_value: score from TP / (TP + FP).
recall	Computes Recall	true_positive (int): System EXTRACTED a text that is an insight / in the Gold Standard. false_negative (int): System did NOT EXTRACT a text that is an insight / in the Gold Standard.	recall_value: score from TP / (TP + FN).
accuracy	Computes accuracy	true_positive (int): System EXTRACTED a text that is an insight / in the Gold Standard. false_positive (int): System EXTRACTED a text that is not an insight / not in the Gold Standard. false_negative (int): System did NOT EXTRACT a text that is an insight / in the Gold Standard. true_negative: System did NOT EXTRACT a text that is not an insight / not in the Gold Standard.	accuracy_value: score from (TP + TN) / (TP + FP + FN + TN).
f_measure	Computes F-Measure or F1	precision_value (float): System EXTRACTED a text that is an insight / in the Gold Standard. recall_value (float): System EXTRACTED a text that is not an insight / not in the Gold Standard.	f_measure_value: score from (2 * P * R) / (P + R).
retrieve_text	Reads contents in candidate and gold standard excel files and stores their values in lists.	candidate_filename (str): file location of the candidate.	phrase_list: a list of strings containing the system extracted (phrases) information.

Function Name	Description	Arguments	Return Type
		reference_filename (str): file location of the reference / gold standard.	word_set_list: a list of strings containing the system extracted (word sets) information. reference_phrase_list: a list of strings containing the gold standard (phrases) information. reference_word_set_list: a list of strings containing the gold standard (word sets) information. word_counter: a value indicating the total number of words in the input sentence / whole corpus.
compare_ie_phrases	Evaluates insight phrases. Displays statistics and results of computed standard metrics.	system_output_list (list): a list of strings containing the system extracted (phrases) information. gold_standard_list (list): a list of strings containing the gold standard (phrases) information. word_count (int): a value indicating the total number of words in the input sentence / whole corpus.	Void (displays evaluation results)
compare_ie_word_sets	Evaluates insight word set. Displays statistics and results of computed standard metrics.	system_output_list (list): a list of strings containing the system extracted (phrases) information. gold_standard_list (list): a list of strings containing the gold standard (phrases) information.	Void (displays evaluation results)

4 Resources

Incorporated in the API package are the necessary resources needed in order to run the modules. Details of these files are shown at Table 4.1 and its file structure on Figure 4.1.

Table 4.1 Resource Folders and Files

Resource	Location	Description
[Nokhonfusion]-Filipino-Colloquialism-MT/	model/	Folder containing resources used by the Filipino Colloquialism normalizer.
dictionary/	model/	Folder containing modifiable dictionary files such as Tagalog prefix list and stopwords.
tl_fasttext/	model/	Folder containing FastText's pre-trained Tagalog word embeddings model.
tl_word2vec/	model/	Folder containing Word2Vec's pre-trained Tagalog word embeddings model.
filipino-left5words-owlqn2-distsim-pref6-inf2.tagger	model/	Filipino Stanford Part-of-Speech Tagger model file.
in	test/	Input file of Filipino Colloquialism normalizer.
out	test/	Output file of Filipino Colloquialism normalizer.
MalasakitResponses.xlsx	test/	Malasakit dataset. Upon running full run of API. This file will be used as input and output (sheets will be overwritten while retaining protected cells).
Report.docx	test/	Generated report.

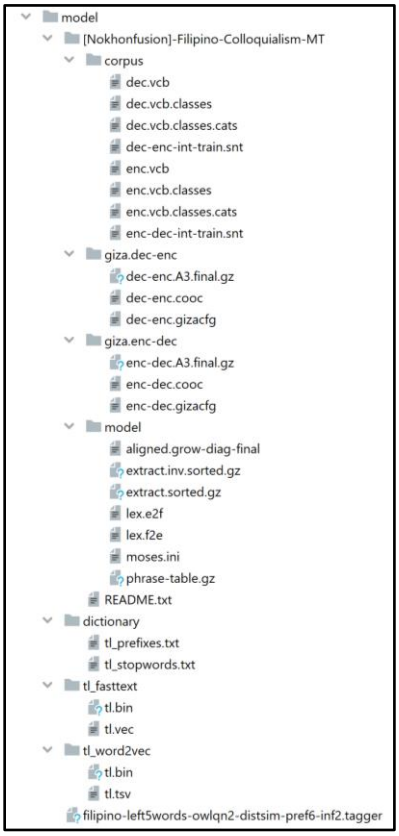

Figure 4.1 Resource File Structure

5 References

Go, M. P., & Nocon, N. (2017). Using stanford part-of-speech tagger for the morphologically-rich filipino language. In *Proceedings of the 31st pacific asia conference on language, information and computation* (pp. 81–88).

Nocon, N., Kho, N. M., & Arroyo, J. (2018). Building a filipino colloquialism translator using sequence-to-sequence model. In *Tencon 2018-2018 ieee region 10 conference* (pp. 2199–2204).

Oco, N., & Borra, A. (2011). A grammar checker for tagalog using languagetool. In *Proceedings of the 9th workshop on asian language resources* (pp. 2–9).