IRWA FINAL PROJECT

PART 1

TEXT PROCESSING AND EXPLORATORY DATA ANALYSIS

Jordi Guillén: u198641
Anira Besora: u189647
Ana Cereto: u199767
Marina Castellano: u188311

1. PRE-PROCESS THE DOCUMENTS

We'll keep hashtags as separate words, but remove the "#" symbol itself. By treating hashtags like regular words after taking out the "#", they'll still be easy to search and include in the index, without any special characters causing issues.

To make important hashtags stand out more, especially the ones tied to specific information needs (like topics about the Indian government), we'll use a repetition technique. After the "#" is removed, we'll repeat the hashtag word a couple of times (like twice or three times) in the document index. This way, the word becomes more important and is easier to match with related queries, especially when the hashtag is closely connected to the topic being searched for.

We will ensure that only the tweets that have been mapped to document IDs (using the `tweet_document_ids_map`) are kept for the evaluation stage of the project. Any tweets that don't have a corresponding document ID in the mapping will be deleted from the dataset, as they won't be useful for the evaluation. This ensures that the final dataset only includes relevant tweets linked to the evaluation process.

2. EXPLORATORY DATA ANALYSIS

Word count and top words:

The objective is to print the top 10 most frequent words in a collection of tweets. To achieve this, we do the following process:

1. content_terms = content_words():

There the function *content words* is called. This function iterates through all the tweets in tweets_separated_info, a set that we have filtered before by removing mentions, hashtags, URLs, extra spaces, and emojis. If the tweet has content, we build the terms by tokenizing them, eliminating the stopwords, and performing stemming. We do an update on the Counter *word_counts*, that will add every token to the set if it's not in there and start a new count or add 1 to the count if the token is already there. The function will return that set.

2. top_10_words = content_terms.most_common(10)

This returns the 10 elements from the set with the higher count.

3. print("Top 10 most frequent words:")
   for word, count in top_10_words:
     print(f"{word}: {count}")

Finally, we print it.

Word clouds for the most frequent words:

The objective is to represent in word clouds the most frequent words in the tweets we have done in the word count and store them in the variable content_terms. For that we have created the function plot_top_x_words_wordcloud(words, x) where *words* is a Counter and x is an integer.

The function grabs the most_common x words and then plot the WordCloud. Since we sent the function the content_terms and x = 50, the word cloud will show the most common 50 words of the content-filtered tweets.

Average sentence length:

The objective is to compute the average sentence length of all tweets. For that, we create the function calculate_avg_sentence_length_for_collection(tweets). We will compute the average length of the tweets individually with the function average_sentence_length_for_tweet(content) and add it to the variable total_length. This variable will end up having the total length of the sum of all the tweets. The average lenth will be dividing this with the number of tweets we have stored in the variable total_tweets.

When we call the function we send the variable tweets_separated_info, which are the filtered tweets.

Top hashtags:

The objective is to find the top hashtags. To do so, we have created the function find_top_hashtags(tweet_dict, top_n) where tweet_dic is the dictionary of tweets and top_n is the number that we want to know the top of.

For each tweet in the dictionary, we select only the hashtags. From that list, we create a Counter that will add every hashtag to the set if it's not in there and start a new count or add 1 to the count if the token is already there. From there. we select the most common top_n words.

When we call this function we will send tweets_values_to_keep, which contains the hashtags. Then we create the cloudplot.

Most retweeted/ liked:

We create a function for each request.

To get the most retweeted tweets we create the function get_most_retweeted(tweets, n) where tweets are all the tweets and n is to get the most top n retweeted tweets.

To get the most liked tweets we create the function get_most_liked(tweets, n) where tweets are all the tweets and n is to get the most top n liked tweets.

Both functions sort the tweets based on the value of the ‘retweetCount’ or ‘likeCount’ key in each dictionary, in descending order. Then it selects the first n elements of the list. Then we print them.

Github/ Tag:
https://github.com/Marinagrup2/IRWA.git