**HUMBER INSTITUTE OF TECHNOLOGY**

**AND ADVANCED LEARNING**

**(HUMBER COLLEGE)**

****

**Case Study #2: Business Intelligence for BIA - BIA-5401-0LA**

**Sentiment Analysis of Hotels in Montreals**

**Date: August 3rd, 2023**

Submitted by Group 4:

Ali Mirza (N01527331)

Andres Ramirez (N01510788)

Ankita Patil (N01510453)

Gizelle Lao (N00685940)

Rafael Muniz (N01510740)

Trung Le (N01455980)

Submitted to: Professor Haytham Qushtom

Table of Contents

[I. Introduction to the Dataset 3](#_heading=h.gjdgxs)

[II. Data Storage and Consolidation 3](#_heading=)

[III. Data Cleaning and Text Preprocessing 6](#_heading=h.1fob9te)

[IV. Feature Extraction (bags-of-words) 12](#_heading=h.3znysh7)

[V. Sentiment Analysis Models 13](#_heading=h.2et92p0)

[TextBlob 14](#_heading=h.tyjcwt)

[VADER 14](#_heading=h.3dy6vkm)

[Hugging Face Transformer 15](#_heading=h.1t3h5sf)

[Results 16](#_heading=h.4d34og8)

[VI. References 19](#_heading=h.2s8eyo1)

## I. Introduction to the Dataset

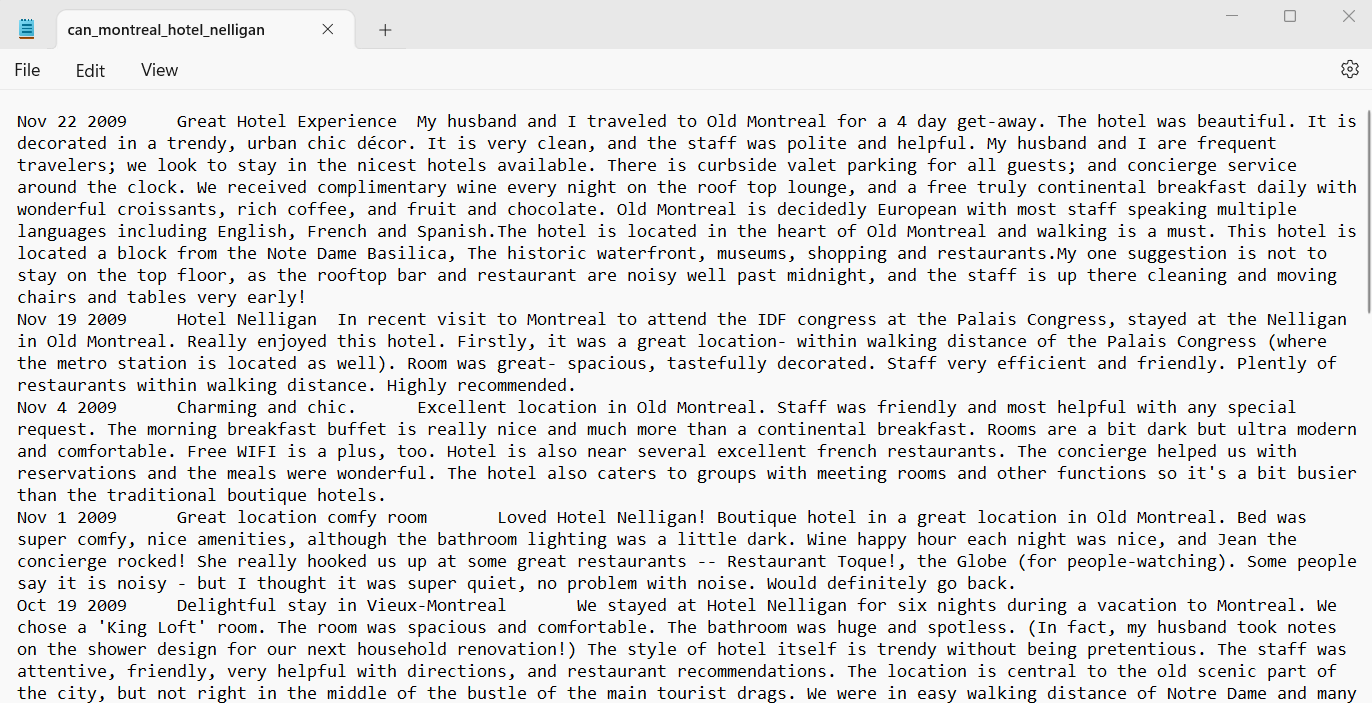
The rapid increase of textual information from resources like social media, consumer review, and feedback has brought both opportunities and challenges in today's data-driven businesses. Market challenges put pressure on businesses to remain flexible and competitive, demanding decisions based on data for expansion and productivity. On-premises systems, however, find it difficult to fully utilize data.

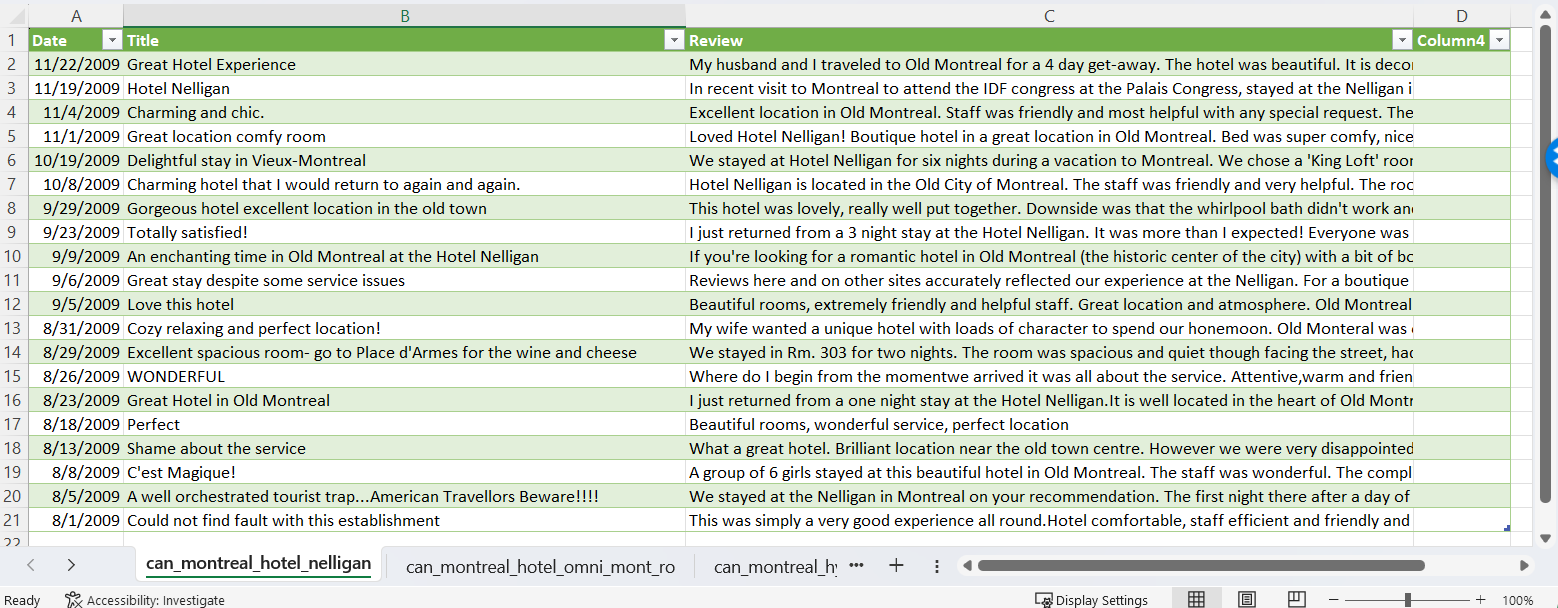
In this report we assist hotels in Montreal to develop a text-mining strategy to analyze customer feedback from consolidating their review collected from online platforms including booking websites, social media, and specialized review sites into a non-structural database (No-SQL). Then we performed data cleaning, text preprocessing and finally conducted three sentiment analysis models to evaluate customer’s experience (positive or negative) upon their travel to Montreal. Data from 10 hotels in Montreal with 20 reviews each, sentiment analysis enables hotel businesses to evaluate customer negative feedback and identify areas for improvement.

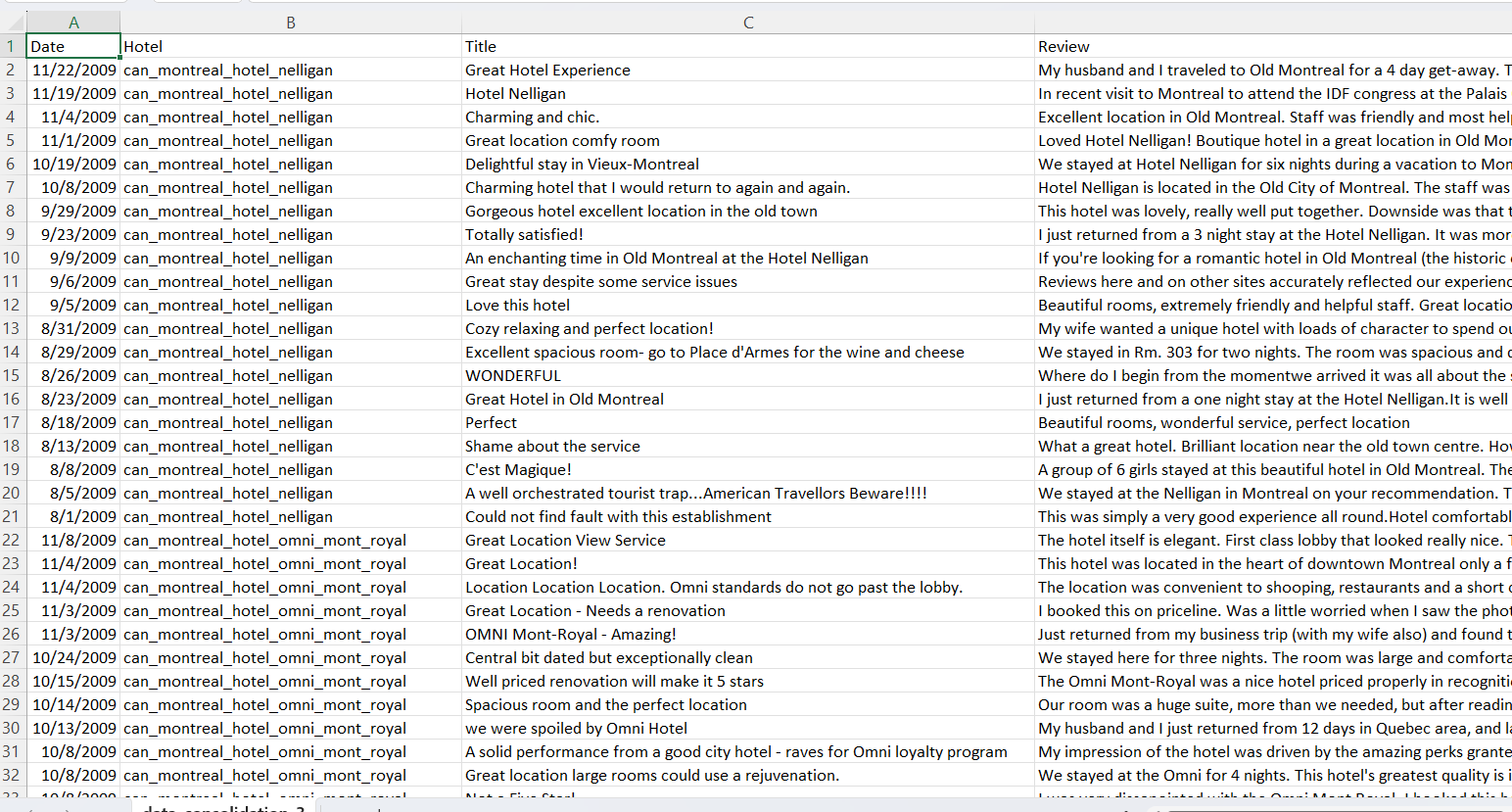
## II. Data Storage and Consolidation

To be able to perform the text-mining analysis in this project, first some data processing was conducted. The data was collected from the OpinRank Review Dataset (Ganesan & Zhai, 2011), in which 20 reviews from 10 different hotels, in Montreal, were selected.

After selecting those mentioned reviews, the data was transformed from a TAB separated text file (Figure 1) into a CSV file (Figure 2), as shown below:

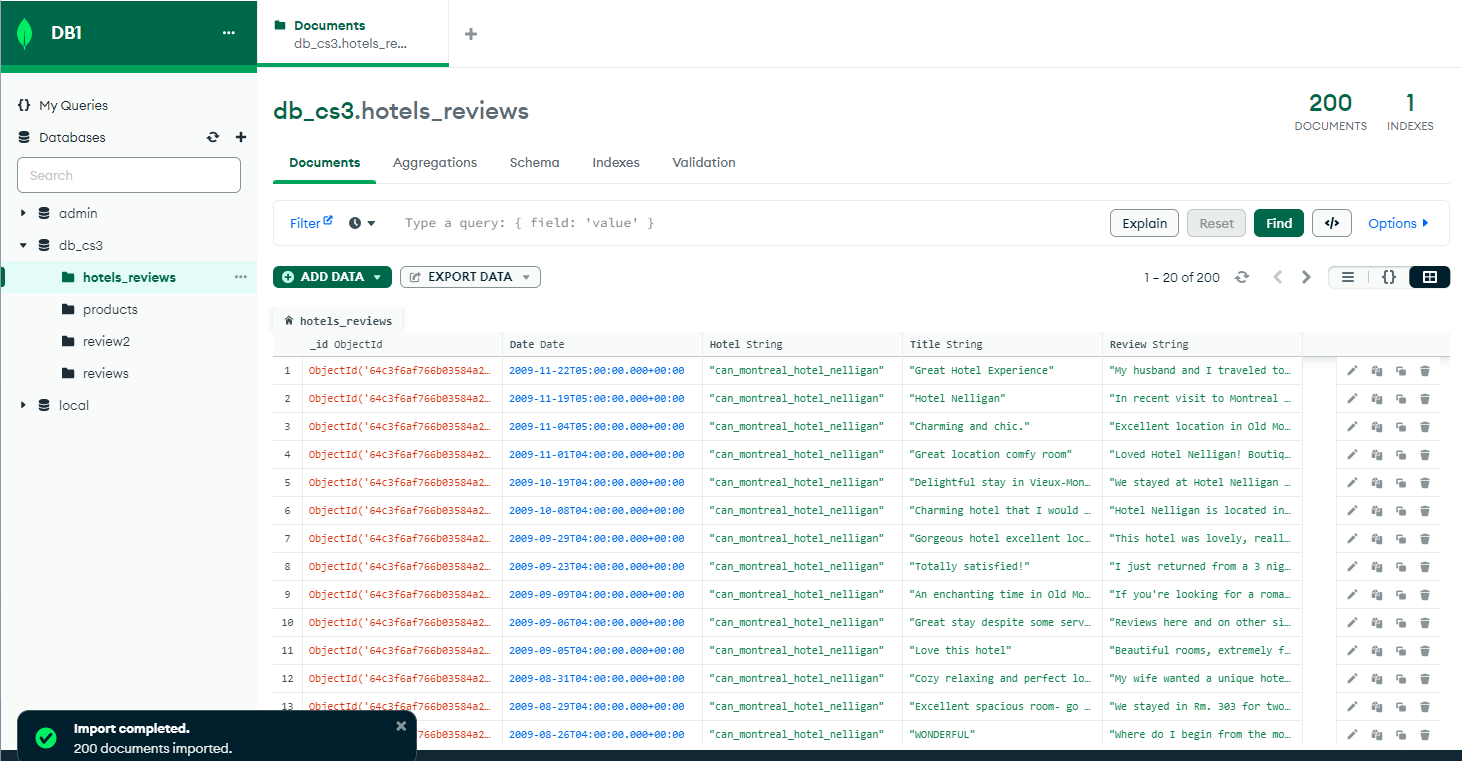
**Figure 1** – TAB separated text file with an hotel’s reviews

**Figure 2** – Reviews transformed into a CSV file

Once the reviews from each hotel were transformed into a CSV file, they were consolidated into one CSV file (Figure 3): 

**Figure 3** – Consolidated CSV file

After the step above, the data were uploaded into a Nosql database, called MongoDB, as shown in Figure 4:



**Figure 4** – MongoDB database with the hotels’ reviews

The next steps were to create the connection between this MongoDB database with the Jupyter Notebook (Figure 5), and to select the specific Database and Collection to work on (Figure 6):



**Figure 5** - Connecting MongoDB to the Jupyter Notebook



**Figure 6** - Selecting the specific Database (inside MongoDB) and the Collection

The Collection mentioned above (inside the MongoDB database) was used to do the data preprocessing and the text-mining analysis through a machine learning model that will be explained further in this report.

## III. Data Cleaning and Text Preprocessing

It is important to conduct thorough data cleaning and preprocessing when working while handling large datasets to ensure high-quality input data, address missing values and data consistency, resulting in optimal model performance (Qushtom, 2023). Following the process required to connect to the MongoDB database, data cleaning and preprocessing was conducted in Jupyter Notebook. There were no missing values in the data set, as the initial data was consolidated and placed in a master .csv file, then uploaded into MongoDB. Next, URLs were removed from the data set to remove noise and allow for consistent data. The below figure 7 reflects the code used to remove the URLs:

A screenshot of a computer

Description automatically generated**Figure 7:** Removing URLs

To enhance the model’s reliance on unique text and optimize the final output, irrelevant characters and punctuations were removed. Removing the non-alpha-numeric characters such as commas, periods and hyphens was achieved in Figure 8:

**Figure 8:** Removed irrelevant characters and punctuations.A screenshot of a computer

Description automatically generated

Also, it was integral to avoid duplications for an accurate word count. Through the process of converting all characters in reviews to lowercase, it prevents the model from considering words such as Great, great and GREAT as distinct, then treats them as the same word despite the lower case and upper case differences. It is evident in the final random sample in Figure 9, the characters in the review column were converted into lower case:

A screen shot of a computer code

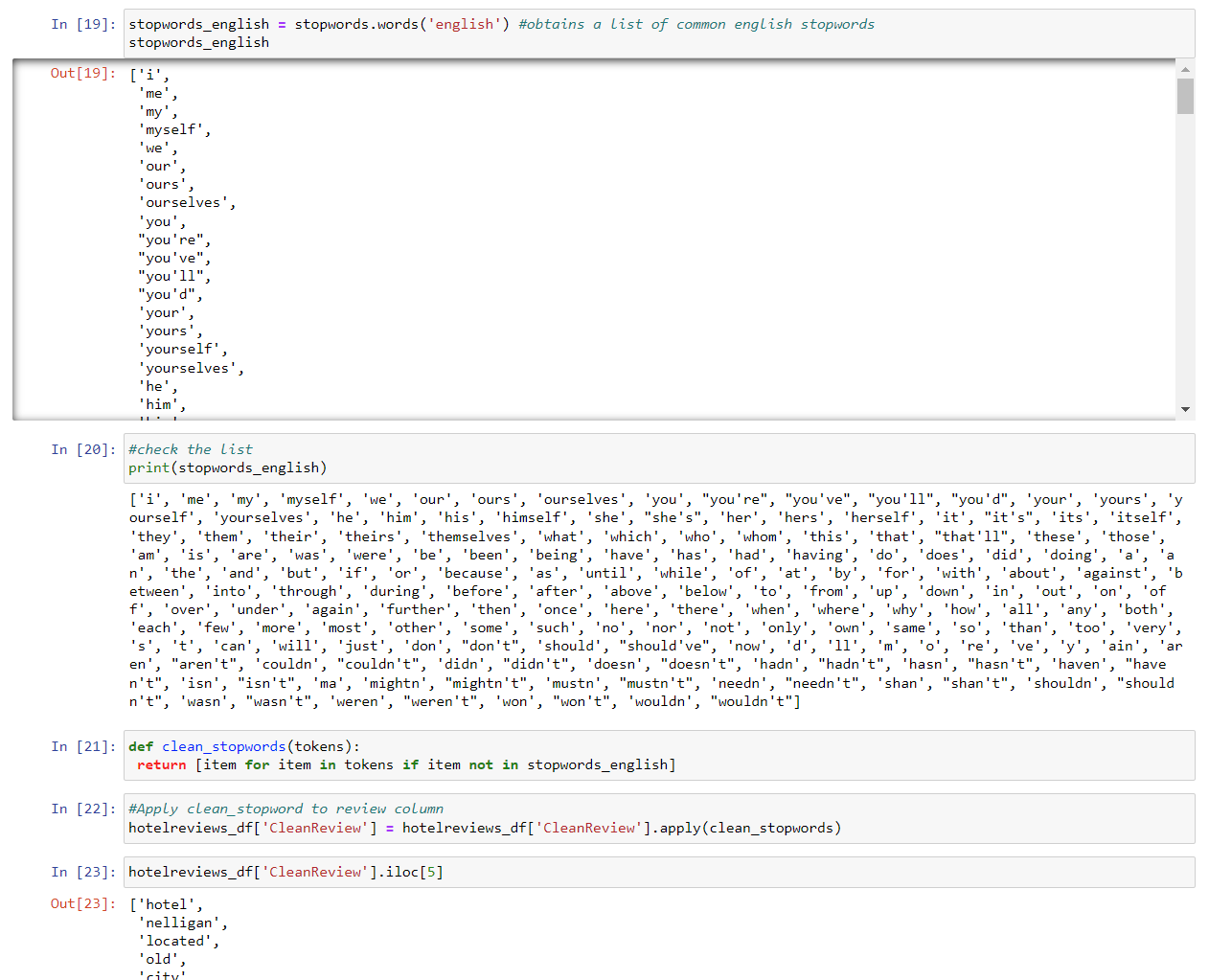
Description automatically generated  
**Figure 9**: All characters were transformed into lower case

Tokenization was conducted as it involves splitting the text into smaller units known as “tokens”. This process is essential for feature extraction, as it represents the values as individual words. This was accomplished by importing the Natural Language Toolkit (NLTK) shown in Figure 10:



**Figure 10:** Tokenization of the text for all reviews.

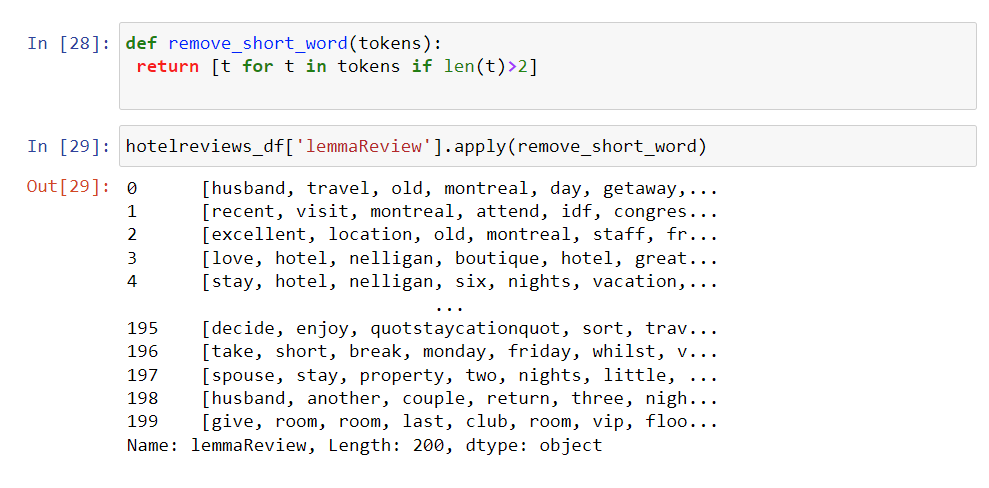
In addition to removing characters and punctuations as mentioned above, removal of stop words was needed. Stop words are those that are most common and not part of the NLTK package. As shown in Figure 11, stop words in English were removed:



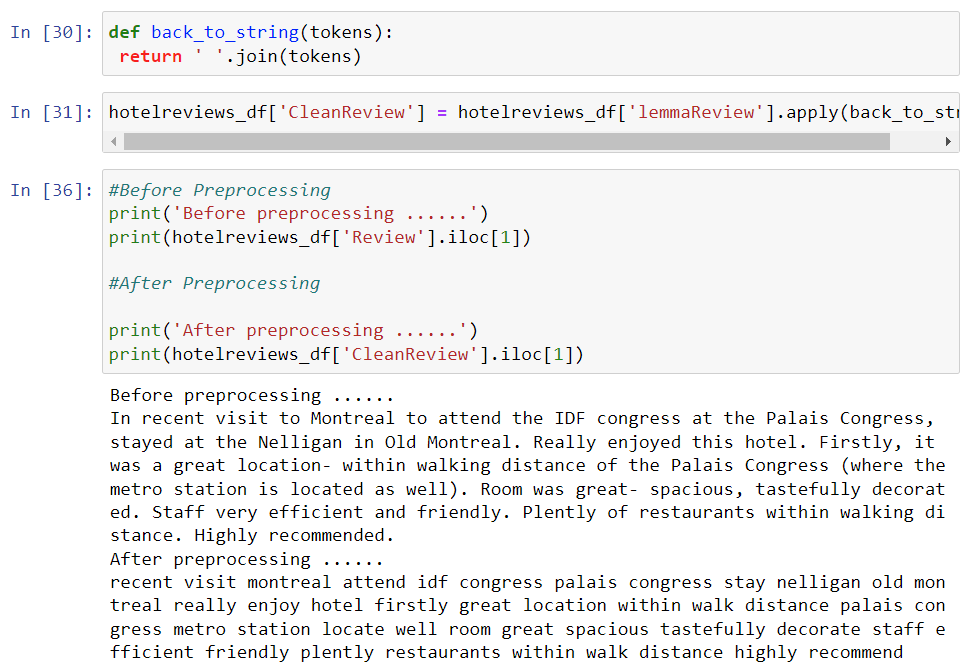
**Figure 11:** Removing stop words in English

Next step, stemming and lemmatization were performed to convert words back to their base form in order to reduce dimensionality of the data. For example, after the lemmatization process, the word “visited” was transformed back to the original form “visit”. Since we have a small dataset, only lemmatization will be used for further processing in developing sentiment models (Figure 12).

**Figure 12:** Stemming and Lemmatization of Review

Following the Steaming and Lemmatization process, we remove the short word with length less than 2 to reduce noise in our corpus (Figure 13):

**Figure 13**. Remove short word having length less than 2

Finally, we revert the list of tokens back to string format and compare with the original review (Figure 14):

**Figure 14**: Before and after preprocessing

## IV. Feature Extraction (bags-of-words)

Bag-of-words was created by extracting features (words) from the review text, which was contained in the ‘CleanReview’ column. Figure 15 shows the process of creating the bag-of-words. The text in the reviews was vectorized using the CountVectorizer() function imported from scikit-learn. Then, the text was tokenized, words were extracted from the CountVectorizer and stored in a dataframe. In the dataframe, each word appears as a column while each row corresponds to a hotel review in which that word either appeared or not.

A screenshot of a computer

Description automatically generated

**Figure 15**. Creating Bag-of-Words

The most frequent 20 words, across all reviews, are shown in Figure 16. Unsurprisingly, the top 4 words are the hotel, room, stay, and the city name, Montreal. These words are used by both positive and negative reviewers to establish context. In fact, most of the top 20 words are context words. The second most popular category of words are words such as ‘great’, ‘nice’, ‘good’ etc., which may indicate that most reviews are positive. There are no noticeable words that clearly indicate negative sentiment.

A graph of words in a bag of words

Description automatically generated

**Figure 16**: Top 20 Words in Hotel Reviews

## V. Sentiment Analysis Models

To know the sentiment of each review the following 3 models were used. Using an odd number of models helped with the mitigation of the bias of each model and at each we chose the mode of each label.

TextBlob: The textblob\_sen function utilizes the TextBlob library for sentiment analysis. TextBlob is a simple natural language processing library that provides an API for common NLP tasks, including sentiment analysis (TextBlob, n.d.).

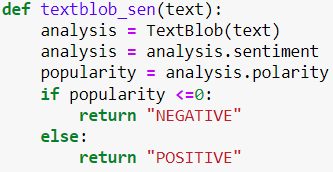
VADER: Valence Aware Dictionary for Sentiment Reasoning is a model used for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion. It is available in the NLTK package and can be applied directly to unlabeled text data (Beri, 2020).

Hugging Face Transformers: The pipeline uses a pre-trained model for sentiment analysis, specifically a DistilBERT model for text classification to predict sentiment (Hugging Face, n.d.).

### TextBlob

The code in Figure 17 defines a function called textblob\_sen that performs sentiment analysis on a given piece of text. Sentiment analysis is a natural language processing technique that aims to understand the emotions and attitudes conveyed in a text (AWS, n.d.).

In this function, we use a library called TextBlob, which is a tool for processing textual data. When we pass a text (like a review in this case) to the textblob\_sen function, it creates a TextBlob object to analyze the sentiment. The sentiment analysis process involves assigning a numerical score to the text, known as "polarity." A positive polarity (greater than zero) means the text has a positive sentiment, while a negative polarity (less or equal to zero) indicates a negative sentiment (TextBlob, n.d.), so the function returns "NEGATIVE" to indicate that. Otherwise, the function returns "POSITIVE."



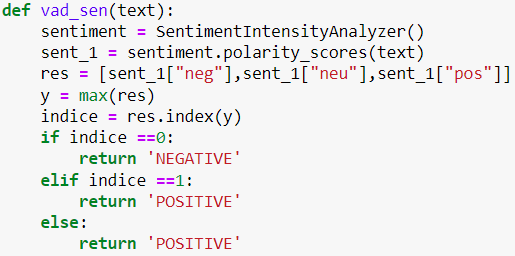
**Figure 17.** Define TextBlob function

### VADER

The function in Figure 18 creates an instance of the SentimentIntensityAnalyzer class, a crucial component of VADER that helps in understanding sentiment. Once the analyzer is ready, the function proceeds to calculate sentiment scores for the input text. These scores represent the degree of positivity and negativity conveyed in the text. The polarity\_scores method provides a dictionary containing these scores.

The function then extracts the specific scores for negative and positive sentiments from the dictionary and stores them in a list. Afterward, it identifies the highest score among the three, which indicates the dominant sentiment expressed in the text.

Based on the index of the highest score, the function assigns a sentiment label to the text. If the highest score corresponds to the "neg" (negative) aspect, the function labels the text as "NEGATIVE." If it relates to the "pos" (positive) aspect, the text is labeled as "POSITIVE." However, if there's a tie between the "neu" (neutral) and "pos" scores, the function considers it as "POSITIVE" as well.



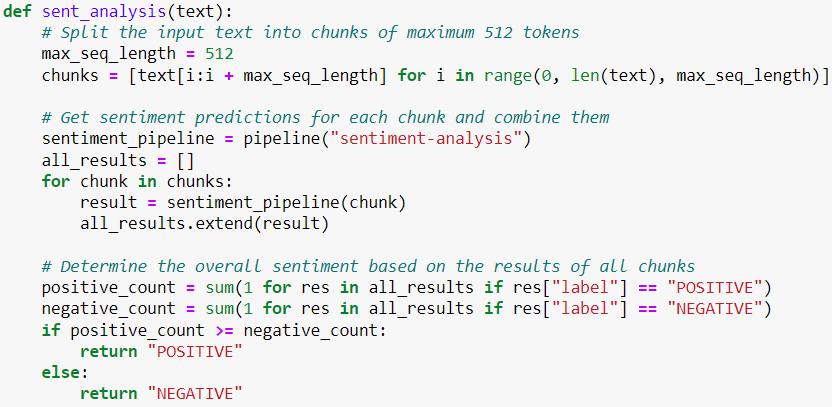
**Figure 18.** Define VADER function

### Hugging Face Transformer

The function sent\_analysis defined in Figure 19 is designed to analyze the sentiment of a given text using a pre-trained language model specialized in understanding the emotions conveyed in natural language. It starts by dividing the input text into smaller segments, ensuring that each segment contains a maximum of 512 words. This step is necessary as the DistilBERT model has limitations on the length of the input it can process effectively.

Next, the function utilizes a pre-trained sentiment analysis model, known as a "pipeline," to evaluate the sentiment of each segmented part of the text individually. Each segment is processed through the pipeline, resulting in sentiment predictions for each segment. Further, the sentiment predictions from all segments are then combined into a list called all\_results, which holds the sentiment labels (either "POSITIVE" or "NEGATIVE") for each corresponding segment.

To determine the overall sentiment of the entire input text, the function calculates the total count of "POSITIVE" and "NEGATIVE" sentiment labels in the all\_results list. If the number of "POSITIVE" sentiment labels is greater than or equal to the number of "NEGATIVE" sentiment labels, the function concludes that the overall sentiment of the text is "POSITIVE." Otherwise, it determines the overall sentiment to be "NEGATIVE."



**Figure 19.** Define Hugging Face Transformer function (DistilBERT)

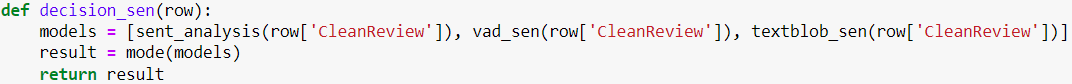
### Results

The decision\_sen function in Figure 20 takes a single row of a data frame as input, which represents a specific hotel review. It aims to determine the overall sentiment of the review by utilizing three different sentiment analysis models: sent\_analysis, vad\_sen, and textblob\_sen.

First, the function calls each of the three-sentiment analysis models and passes the cleaned review text, which is obtained from the 'CleanReview' column of the input row. These models individually analyze the sentiment of the review and provide their respective predictions. The predictions are then stored in the model’s list.

Next, the function uses a statistical concept called "mode" to find the sentiment label that appears most frequently among the predictions in the model’s list. In other words, it identifies the sentiment label that has the highest count among the three models.

Finally, the function returns the sentiment label with the highest count as the overall sentiment for the review. This approach of aggregating the predictions from multiple sentiment analysis models helps to achieve a more robust and reliable sentiment classification for each hotel review, providing a more comprehensive understanding of the sentiments expressed by the reviewers.



**Figure 20.** Overall sentiment function definition

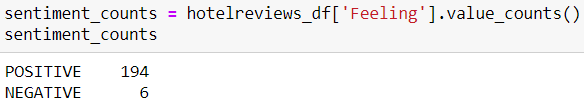
The objective of the code in Figure 21 is to randomly select 10 rows from this data frame for further analysis. To achieve this, the sample() method is used, specifying n=10 to select 10 random rows. Additionally, random\_state=42 is set to ensure reproducibility, meaning that if the code is executed again with the same random state, it will yield the same random rows (Pramoditha, 2022).

After obtaining the 10 randomly selected rows, we want to focus on specific columns for analysis. Therefore, we create a new data frame called random\_rows, which contains only the columns we are interested in \_id, Date, Hotel, CleanReview, and Feeling. By using the syntax random\_rows[selected\_columns], we filter the random\_rows DataFrame to show only the desired columns. The output displays the information from these selected columns for the 10 randomly sampled hotel reviews, providing a more concise and relevant view of the data for our analysis. This allows us to easily examine the key information related to each review, such as its unique identifier (\_id), date of submission (Date), hotel name (Hotel), the cleaned review text (CleanReview), and the sentiment label (Feeling).



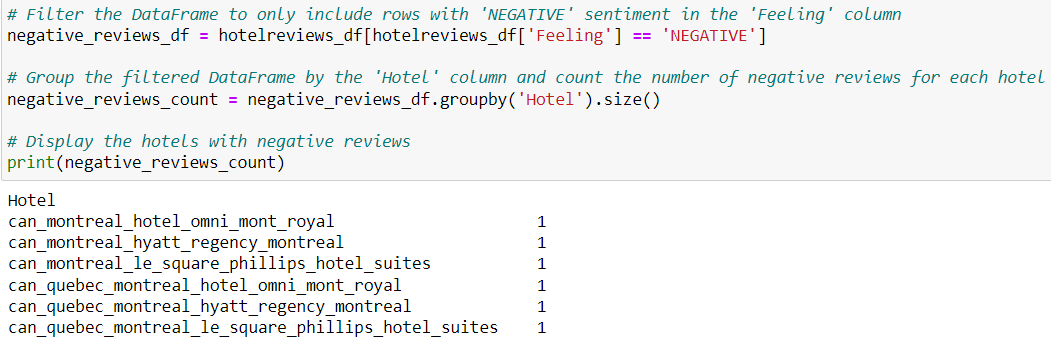
**Figure 21.** Display 10 random reviews and sentiment

Results in Figure 22 shows that from the 200 reviews across 10 hotels in Montreal, 194 are POSITIVE and only 6 are NEGATIVE.



**Figure 22.** Sentiment count

Finally, Figure 23 shows the name of the hotels that have negative reviews.



**Figure 23.** Hotels with negatives reviews

## VI. References

AWS. (n.d.). *What Is Sentiment Analysis?* Retrieved July 29, 2023, from AWS: https://aws.amazon.com/what-is/sentiment-analysis/

Beri, A. (2020, May 27). *Sentiment Analysis Using VADER*. Retrieved from Towards Data Science: https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664

Ganesan, K., & Zhai, C. (2011). OpinRank Review Dataset. UCI Machine Learning Repository. Retrieved July 29, 2023, from https://doi.org/10.24432/C5QW4W

Hugging Face. (n.d.). *DistilBERT*. Retrieved July 29, 2023, from Hugging Face: https://huggingface.co/docs/transformers/model\_doc/distilbert

Pramoditha, R. (2022, April 30). *Why do we set a random state in machine learning models?* Retrieved from Towards Data Science: https://towardsdatascience.com/why-do-we-set-a-random-state-in-machine-learning-models-bb2dc68d8431#:~:text=With%20random\_state%3D42%20%2C%20we%20get,affect%20the%20model's%20performance%20score.

TextBlob. (n.d.). *TextBlob: Simplified Text Processing*. Retrieved July 29, 2023, from TextBlob: <https://textblob.readthedocs.io/en/dev/#:~:text=TextBlob%20is%20a%20Python%20(2,classification%2C%20translation%2C%20and%20more>.

Qushtom, H (n.d.). *Data\_preprocess [PDF document]*. Retrieved July 29, 2023, from Blackboard: https://learn.humber.ca/ultra/courses/\_226294\_1/outline