


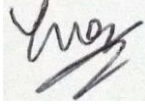




Attn: Dr. Sun Aixin



AI6122 Text Data Management and Processing

Group ID: G 31

We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below. We have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work. We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work.

Name	Signature / Date
YIP CHEN FEI	 31 October 2024
KEE MING YUAN	 31 October 2024
WONG SEIK MAN	 31 October 2024
TIMOTHY YEO XIAO JING	 31 October 2024
TAN SHU TING GERMAINE	 31 October 2024
SHAHRUL AL-NIZAM LING	 31 October 2024

Important note:

Name must **EXACTLY MATCH** the one printed on your Matriculation Card. Any mismatch leads to **THREE (3)** marks deduction.

ABSTRACT

Using the Tucson reviews, a search engine was built based on 2 main indexes – the review index and the business index. Preprocessing like casefolding, stemming and removing of stop words were used. The search engine executes keyword search of businesses (restaurants), reviews, and geospatial search. It also produces users' review summary. An additional application was created with sentiment analysis to help users and business owners better understand the experience patrons are having at the business, for better food decisions and business improvements respectively.

1 Introduction

Yelp, a popular online platform for local business reviews, provides a dataset rich in insights into consumer behaviour, business trends, and sentiment analysis. As it is free to use, Yelp is widely adopted in the US, making its dataset suitable for natural language processing (NLP) and information retrieval (IR) research. This study is divided into four key segments.

Dataset Analysis Tucson is chosen as the focus area, with tokenization and stemming performed, and results before and after stemming discussed. Writing styles on various platforms are also reviewed and compared.

Search Engine Development A search engine is built to support keyword searches for businesses, reviews, and geospatial data. The structure and indexing process are described, along with query processing times.

Review Summary Users' reviews are also summarized to show the number of reviews contributed, the bounding box, the most frequent words used and the most representative sentences.

Application A sentiment analysis application was developed to classify reviews as positive, negative, or neutral, providing valuable insights into customer satisfaction and business reputation.

2 Dataset Used

Two datasets were obtained from Yelp's Open Dataset: the business dataset and the review dataset. Since there were 11 metropolitan cities, we focused on 1. We built heat maps of the 11 cities representing the reviews, taking into consideration the standardized values of business variety, business review count and location. Figure 1 shows the 3 cities that we shortlisted. The basis of a good study is a good dataset, so our goal is to choose a dataset that is fair and well-distributed.

Looking at the heatmaps, it would seem initially that Indianapolis is the best choice as the reviews seem to be distributed radially outwards with a higher concentration in the middle. The search engine has a geospatial search as well, so this may be more suitable. Philadelphia seems to be geospatially skewed towards one part of the city and very densely packed. This suggests the restaurants might all be located in one area only. Comparing Tucson and Indianapolis, while

Indianapolis has a very well-spaced heatmap, Tucson's distribution is also fairly ideal. The metrics in Table 1 below are also comparable for the 2 cities. Eventually we favored the city with a higher number of reviews as it provided us with more data points for the machine learning. Tucson (9,250 businesses) has 24% more businesses than Indianapolis (7,540 businesses).

Table 1: Standardised values of business variety, business review count and location for Indianapolis, Tucson and Philadelphia

	Indianapolis	Tucson	Philadelphia
Business variety std.	1.68	1.55	1.38
Business review count std.	124.72	114.49	188.11
Location std.	62.98	71.59	57.56

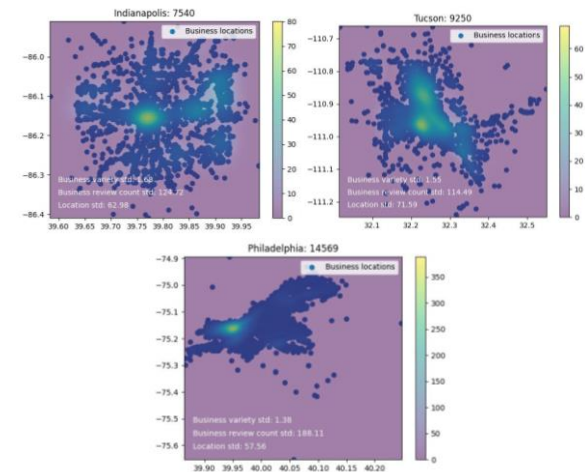


Figure 1: Heat maps of 3 shortlisted cities, representing reviews geolocationally

Focusing on the city of Tucson, we filtered the business dataset to include only businesses located within this area, resulting in 9,250 unique businesses. Correspondingly, the review dataset was filtered to retain only reviews related to these 9,250 businesses, yielding a total of 404,880 reviews. These datasets will be used for subsequent tasks in our research. A detailed specification of the datasets, including attribute names, data types, descriptions, and example values, are provided. (See Appendix)

3 Tokenization and Stemming

In the previous section, we extracted businesses located in Tucson and stored them in a Data Frame. Now, we use the pandas sample function to randomly select two businesses from the Data Frame.

```
# Randomly select two business_ids
random_business_ids =
tucson_df['business_id'].sample(2).tolist()
```

The two businesses randomly selected are B1 'Whole Foods Market' and B2 'Taco Fish'. After extracting the review text for each business, we used the NLTK library

to run an English word tokenizer to obtain tokens, then conducted case-folding to convert these tokens to lowercase. Two experiments were run to observe the effects of stemming. In Experiment 1, we only removed stopwords such that only the meaningful words remained and collected the top 10 most frequent words. In Experiment 2, we removed stopwords and applied stemming to keep the core of each word. The top 10 most frequent words were then collected.

Stopwords often contribute little semantic value to the meaning of text. NLTK's English stopwords list includes 179 words. As observed in Experiment 1, removing these stopwords makes indexing and querying algorithms faster and more efficient by reducing data complexity. Referring to Table 2, removing stopwords significantly reduces storage space, where around 50% of the tokens are removed compared to without. For B1, removing stopwords brought the number of tokens down to 42.86%, and brought down to 51.78% for B2. For Experiment 2, stemming further reduced the token counts in both B1 and B2, leaving approximately 10% of the unique tokens after all preprocessing steps.

Table 2: Review statistics for Experiments 1 and 2 after removing stop words and stemming.

	Review statistics	
	B1, Whole Foods Markets	Tokens in %
Total tokens in review text	21219	100%
Number of tokens after removing stopwords (Experiment 1)	10554	42.86%
Remove duplicate tokens (unique tokens left)	2802	13.20%
Unique tokens after stemming (Experiment 2)	2203	10.38%

	Review statistics	
	B2, Taco Fish	Tokens in %
Total tokens in review text	15372	100%
Number of tokens after removing stopwords (Experiment 1)	7961	51.78%
Remove duplicate tokens (unique tokens left)	1891	12.30%
Unique tokens after stemming (Experiment 2)	1526	9.92%

Referring to Figure 2, The blue bars show the top 10 frequent words without stemming (Experiment 1) and the orange bars show the top 10 frequent words with stemming (Experiment 2). For B1, the changes in the top 10 words after stemming, are a result of how stemming reduces words to their base forms. This process consolidates different word variants (e.g., singular vs. plural) into a single stem. The word "foods" without stemming is considered its own word. After stemming it is reduced to "food", which now includes both singular and plural forms. This causes "food" to appear with a significantly higher frequency post-stemming, as seen in the chart, while "foods" does not exist. This is similar for B2, where "taco" becomes higher frequency after

stemming, and "tacos" is stemmed to become "taco" and does not exist post-stemming.

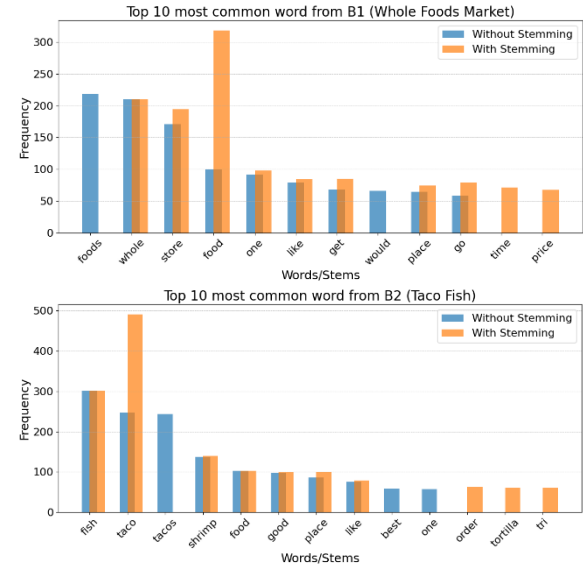


Figure 2: Top 10 most common words for B1 and B2

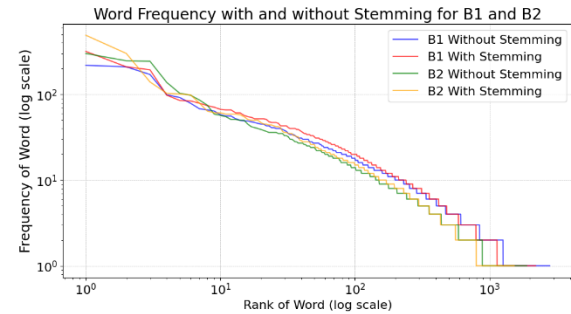


Figure 3: Word frequency for the B1 and B2

In the word frequency log scale plot, we observe the word frequency distribution for both B1 and B2 (Figure 3) follows Zip's law, where a few words occur frequently while most words appear rarely, creating a long tail. Stemming has a notable effect on high to mid-frequency words, consolidating different forms of the same word into a single stem, which increases their overall frequency. This is reflected in the smoother and slightly higher curves for the "with stemming" lines compared to the "without stemming" ones. In the long tail, rare words are less affected by stemming, as they typically have fewer morphological variants. Overall, stemming reduces vocabulary size while increasing the frequency of some words, especially in the higher ranks.

3.1 Writing style

Stack Overflow posts ¹² are characterized by their instructional nature, employing accurate tenses and spelling within shorter sentences. The use of

¹ <https://stackoverflow.com/questions/1919787/how-do-i-fix-this-stack-overflow-error>

² <https://stackoverflow.com/questions/19389490/how-do-pythons-any-and-all-functions-work>

contractions like "I've" and "I'd" contributes to the platform's emphasis on easy comprehension.

Conversely, Reddit posts³⁴ often contain profanities and informal language, including interjections like "aww" and "yeah." Conversational vocabulary, the occasional omission of full stops, and the use of emojis and abbreviations like "TBH" and "IOU" further contribute to the informal and interactive nature of Reddit communication. Some comments may even begin without capitalization.

Patent documents⁵⁶, on the other hand, are marked by their formal and precise language, incorporating numbers, longer sentences, and abbreviations like "SKU," "EoAT," and "AGV." The inclusion of images and references to figures, along with technical terms like "electroencephalography (EEG)" and "electromyography (EMG)," reflects the specialized knowledge conveyed in these documents. Proper grammar and formal phrasing are strictly adhered to.

Straits Times news articles⁷⁸ are written in a journalistic style, featuring dates, quoted material enclosed in inverted commas, and frequent capitalization of proper nouns such as country and organization names. Longer sentences effectively describe and report on situations, maintaining good grammar throughout.

The writing styles across these platforms vary significantly, reflecting their different purposes and audiences. Stack Overflow prioritizes clarity and conciseness, Reddit embraces informality and interactivity, patent documents emphasize precision and formality, and Straits Times news articles maintain a journalistic standard of reporting.

4 Development of Search Engine

Our chosen programming language used is Python, and the chosen IR is a pure Python-based implementation that requires no compilation or Java, Whoosh [1]. Whoosh allows for free-form indexing of our customised dataset information fields and subsequently parsing search queries with customizable scoring functions. The stopword list and stemming is provided by NLTK from the previous section.

The search engine needs to be built to support the following: keyword search of businesses, reviews, and geospatial search. This same search engine should also ideally have the ability for a user to search for a user's review summary. This means that the index needs to be relatively flexible to achieve all of this. Our aim is to strike a balance between query performance and ease of management.

4.1 Index Structure

The most important part of building the search engine is deciding the index structure. This will also affect how we define a "document" and the indexing and querying runtimes. In addition, we also took into consideration if this were a live project, the index needs to be structured in a way where updating and maintenance can be done in an efficient and secure way as well.

The first option was consider having the documents to represent each business. The document would contain the business name and all the business information, and also a concatenation of all the reviews given to the business and each corresponding user (identified by a user_id). This results in a decently sized index with a document count of 9,250 items, 1 for every business. This option will allow a straightforward search for business by keyword and geolocation. However, this also means each document will be extremely chunky and inefficient for review search and requires heavy post-processing to for the user review summary. In addition, the data is not also properly reverse-indexed. Considering ease and efficiency of management, this first option also means that every time there is a new user review, an update must be made to the full document. This results in higher indexing costs and possible index consistency issues if new reviews and changes are added as users are querying the search engine at the same time.

The second option is to consider each review as a document. This means that each review will contain information like the review itself, the user_id of the user that contributed the review and the business reviewed with all the information of the corresponding business. This means that in total we would have 404,800 documents, 1 for every review. This is much more documents than the first option. While this index structure makes review search and user review summary more straightforward and efficient, the same inefficiency and heavy post-processing problems still exist for business keyword search and geolocation. Any business level search would require iteration over an extremely large index and require much post-processing. This is not ideal. Considering ease and efficiency of management, any change to the business information like business name, keyword or geolocation, will require an update to every document with this business information. This corresponds to every review that this business has received. The index is also not elegant as there will be multiple duplication of business data across each document. The entire index will take up a large portion of memory for the sake of search instead of meaningful data.

³https://www.reddit.com/r/stupidquestions/comments/1g6ks0h/whats_the_most_likely_outcome_if_you_go_to_a/

⁴https://www.reddit.com/r/aww/comments/1g721x7/adopted_a_pup_and_now_i_ve_got_a_little_shadow_who/

⁵<https://patentimages.storage.googleapis.com/2d/8a/4f/24c634abddb85b/EP3347290B1.pdf>

⁶<https://patentimages.storage.googleapis.com/7a/88/95/164b3203ee6669/US9829971.pdf>

⁷<https://www.straitstimes.com/world/middle-east/eu-chief-diplomat-calls-for-ceasefire-in-middle-east-after-sinwars-death>

⁸<https://www.straitstimes.com/world/gunmen-kill-two-mozambique-opposition-figures-ahead-of-election-protests>

The third option, which is what we chose, is to maintain two indexes – one holding all reviews and the other, businesses. This means that in one index, each review is a document, and in the other, each business is a document. The two indexes are related through the “business_id”. This means that the review index (404,800 documents) will not have the duplication of business data, and with every new review, a new document will just be added to the review index. While this review index will contain many documents and growing, each document is relatively lightweight. This also means that the smaller index of businesses (9,250 documents) can be updated more efficiently. This third option is more flexible and efficient overall. From an operational perspective, it facilitates having different teams manage different indexes, and there is also separation of duties and index segmentation from a security perspective.

We set up the schema for each document as described in Figures 4 and 5 for the review index and document index respectively. For queries that require information from both indexes, the “business_id” will have to be obtained from both indexes for any post-processing steps to combine the returns from each index.

```
# Schema for the Tucson Business Reviews JSON data
review_schema = Schema(
    review_id=ID(stored=True, unique=True), # Unique review ID
    user_id=ID(stored=True), # ID of the user who left the review
    business_id=ID(stored=True), # ID of the business being reviewed
    stars=NUMERIC(stored=True, sortable=True), # Rating stars given in the review
    useful=NUMERIC(stored=True), # Useful votes for the review
    funny=NUMERIC(stored=True), # Funny votes for the review
    cool=NUMERIC(stored=True), # Cool votes for the review
    text=TEXT(stored=True, analyzer=StemmingAnalyzer()), # Text of the review
    date=TEXT(stored=True) # Date the review was posted
)
```

Figure 4: Review level document schema

```
# Schema for the Tucson Business JSON data
business_schema = Schema(
    business_id=ID(stored=True, unique=True), # Unique business ID
    name=TEXT(stored=True), # Business name
    address=TEXT(stored=True), # Address of the business
    city=TEXT(stored=True), # City
    state=TEXT(stored=True), # State
    postal_code=ID(stored=True), # Postal Code (storing as ID)
    latitude=NUMERIC(stored=True), # Latitude
    longitude=NUMERIC(stored=True), # Longitude
    stars=NUMERIC(stored=True, sortable=True), # Rating stars
    review_count=NUMERIC(stored=True), # Review count
    is_open=NUMERIC(stored=True), # Is the business open (1 or 0)
    attributes=TEXT(stored=True), # JSON-style attributes stored as text
    categories=TEXT(stored=True), # Categories of the business
    hours=TEXT(stored=True) # Operating hours in JSON-style text
)
```

Figure 5: Business level document schema

4.2 Preprocessing

Before committing to the index, preprocessing of the data was done to facilitate the querying process. The methodology used is as described by in the Section 3 Tokenization and Stemming above.

Stopwords were removed, to facilitate and efficient search. What is left are the meaningful and informative words in the given review. Case folding was also done to avoid mismatches due to capital letters. Stemming was also done to each word in a text field so that the search is as general as possible. Search inputs pertaining to a business or review must not require very specific grammar to reach.

4.3 Runtime of Indexing

The time taken to index the business documents comes to 0.85 seconds (Table 3) per 10% of the total documents. This is acceptable. The time taken to index the review documents is approximately 69 seconds (Table 3) per 10% of the total documents. While this is still manageable, it is not scalable for a full index at larger scales.

Table 3: Runtime of business and review level document indexing

% of total docs	Indexing time for Business Index (9250 Docs) Unit: Seconds	Indexing time for Review Index (404880 Docs) Unit: Seconds
10%	0.66	63.72
20%	1.33	131.75
30%	2.03	196.38
40%	3.46	262.96
50%	4.17	328.33
60%	4.79	396.25
70%	5.44	465.15
80%	6.08	530.62
90%	6.69	597.83
100%	7.33	670.51

The strategy used in writing to the index is single-pass in-memory indexing (SPIMI) as described in Figures 6 and 7, and then merged. This design makes sense when running a live service search whereby each new review is parsed and index updated immediately upon user submission. One would have to consider the bandwidth available in this case, to have the application be as responsive as possible. Other options could be considered if larger scale back-end hardware is available. Distributed indexing in parallel and further index compression should be considered in such larger scale operations.

```
for i, business in enumerate(businesses, 1):
    writer.add_document(
        business_id=business['business_id'],
        name=business['name'],
        address=business['address'],
        city=business['city'],
        state=business['state'],
        postal_code=business['postal_code'],
        latitude=float(business['latitude']),
        longitude=float(business['longitude']),
        stars=float(business['stars']),
        review_count=int(business['review_count']),
        is_open=int(business['is_open']),
        attributes=str(business['attributes']),
        categories=str(business['categories']),
        hours=str(business['hours'])
    )
```

Figure 6: Code snippet of business indexing

```
for i, review in enumerate(reviews, 1):
    writer.add_document(
        review_id=review['review_id'],
        user_id=review['user_id'],
        business_id=review['business_id'],
        stars=int(review['stars']),
        useful=int(review['useful']),
        funny=int(review['funny']),
        cool=int(review['cool']),
        text=review['text'],
        date=review['date']
    )
```

Figure 7: Code snippet of review indexing

4.4 Querying

```
def search_reviews_by_keyword(keyword, top_n=10):  
  
    # Create a query parser for the 'text' field  
    query_parser = QueryParser("text", review_idx.schema)  
  
    with review_idx.searcher(weighting=TF_IDF()) as searcher:  
        # Parse the query with the stemmed keyword  
        query = query_parser.parse(keyword)  
  
        # Perform the search and limit the number of results to 'top_n'  
        results = searcher.search(query, limit=top_n, terms=True)
```

Figure 8: Code snippet of keyword query searching

```
def search_businesses_by_postal_code_sorted(postal_code, top_n=10):  
    # Use an exact match query for the postal code  
    query = Term("postal_code", postal_code)  
  
    with business_idx.searcher() as searcher:  
        # Perform the search and sort by 'review_count' in descending order  
        sorted_results = searcher.search(query,  
            limit=top_n, sortby=FieldFacet("review_count", reverse=True))  
  
        # Output the business details, sorted by review count  
        for result in sorted_results:  
            print(f"Business Name: {result['name']})")  
            print(f"Address: {result['address']}, City: {result['city']},  
                Postal code: {result['postal_code']})")  
            print(f"Stars: {result['stars']}, Review count: {result['review_count']})")
```

Figure 9: Code snippet of geospatial query searching

For the keyword queries, the search is designed to be a ranked retrieval with ranking of the outputs using a TF-IDF weightage, and to return a configurable input of top N results. TF-IDF is used to evaluate the importance of a word in a document relative to a collection of documents. The higher the TF-IDF score, the more important the word is. Equation 1 below shows how TF-IDF is calculated for each word in a document.

$$\text{TF-IDF}(t,d) = \left(\frac{\text{Number of times } t \text{ appears in } d}{\text{Total number of terms in } d} \right) \times \log \left(\frac{N}{\text{DF}(t)} \right) \quad (1)$$

The TF component increases the weight for terms that appear frequently in a document, signaling their importance. However, if a term appears in many documents, the IDF component reduces its weight to avoid overemphasising common words (such as "the" or "and"). This makes TF-IDF effective for information retrieval and search engines, as it helps to prioritise documents that contain significant, less frequent terms, boosting their relevance in ranked search results.

For the geospatial query, we design it based on a user's perspective of inputting a postal code. Given the input, we return the business results ranked by the number of reviews given to each business.

4.5 Results

```
1 search_businesses_by_keyword("Hot Dogs",5)  
  
Business: La Reyna Hot Dogs, Address: 712 E Prince Rd, Stars: 4.5  
Business: El Kora Hot Dogs, Address: 4721 E American Beauty Dr, Stars: 4.5  
Business: Sammy El Sinaloense Hot Dogs, Address: 2327 North Country Club Rd, Stars: 5.0  
Business: Monster Sonoran Hot Dogs, Address: 1439 S 4th Ave, Stars: 3.5  
Business: El Rey Hot Dogs, Address: 1579 W Grant Rd, Stars: 3.0
```

Figure 10: Keyword search of businesses result

For the business search based on keywords (Figure 10), we see that the matched keywords are always present in the results and expected. For example, for the keyword search "Hot Dogs", it returned businesses with "Hot Dogs" in its name, together with information that is relevant for the user, like its address and stars.

```
1 search_reviews_by_keyword("visited multiple times", top_n=5)  
  
User ID: WY8GwECUE61hwdDNYmOdZa, Stars: 4, TF-IDF Score: 57.10225089742659  
Highlighted Excerpt: anesthesia this TIME, since she needed MULTIPL procedures, so...very  
-----  
User ID: WY8GwECUE61hwdDNYmOdZa, Stars: 4, TF-IDF Score: 57.10225089742659  
Highlighted Excerpt: anesthesia this TIME, since she needed MULTIPL procedures, so...did I  
-----  
User ID: J_-IdcTHSosoIyCST3vutg, Stars: 2, TF-IDF Score: 38.19845608157734  
Highlighted Excerpt: happy to come back MULTIPL TIME!  
Back to the service...VISIT repeatedly. I'm talking atleast 10 TIME. They were seated aft  
-----  
User ID: AFLID_qyrrbEeephgVvBsg, Stars: 3, TF-IDF Score: 36.70022357887707  
Highlighted Excerpt: I have already VISIT the gardens MULTIPL TIME.  
-----  
This is the first month...I always schedule TIME to VISIT gardens. Compared...other garden  
-----  
User ID: bdyDFXPPm3-IYs4XPjNjw, Stars: 2, TF-IDF Score: 35.592857526055624  
Highlighted Excerpt: The first TIME we went (around the TIME it first opened), the...flow.
```

Figure 11: Keyword search of reviews result

For the more complex task of review search based on keywords (Figure 11), we search for reviews that indicate multiple visits to said business. The general sentiment of having visited the business multiple times is there and the matched terms to the input keywords are not obfuscated due to the preprocessing steps taken. The TF-IDF score also reflects the number of occurrences of the input keywords and it shows that the weightage of the keyword "time" is higher than that of "visit".

Lastly, the geospatial business search (Figure 12) is a boolean retrieval for the postal code which is not error prone and is as expected, with the postprocessing of ranks by their retrieved review counts. This was conducted considering a user when defining a bounding box will want to look for a place closest to a particular area. The input is thus the postal code. The ranked review count is displayed as well as the number of stars so that the user can make a judgement based on number of reviews and number of stars.

```
1 search_businesses_by_postal_code_sorted("85641", top_n=5)  
  
Business Name: Budget Car Rental  
Address: 1 Terminal Dr, City: Tucson, Postal Code: 85641  
Stars: 2.0, Review Count: 46  
-----  
Business Name: Molecular Munchies  
Address: , City: Tucson, Postal Code: 85641  
Stars: 5.0, Review Count: 21  
-----  
Business Name: Substance Diner  
Address: , City: Tucson, Postal Code: 85641  
Stars: 4.5, Review Count: 17  
-----  
Business Name: Southern Arizona Plumbing  
Address: , City: Tucson, Postal Code: 85641  
Stars: 4.5, Review Count: 16  
-----  
Business Name: 2 Sisters Photography  
Address: , City: Tucson, Postal Code: 85641  
Stars: 5.0, Review Count: 15  
-----
```

Figure 12: Geospatial search result

5 Review Summary

A review summary of a queried reviewer (user_id) contains the number of reviews the user has given, the type of businesses the reviewer has reviewed, the top 10 most frequently used words and phrases the reviewer used in the review(s) and the 3 most representative sentences selected from the review(s). User_id is the input, and the number of reviews submitted by the user is counted. In Figure 13, we plot the number of reviewers against the number of reviews contributed per users and we observe that most users contribute a small number of reviews, while a small group of enthusiastic reviewers contribute many reviews.

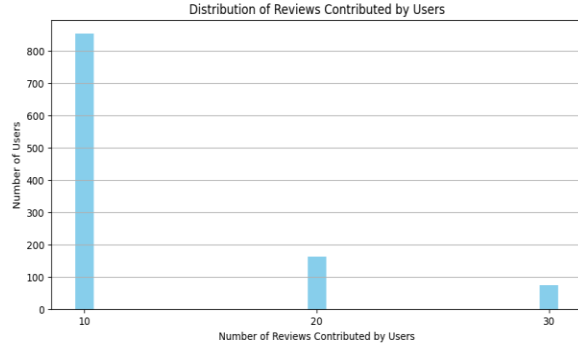


Figure 13: Distribution of the number of reviews contributed by the reviewers.

5.1 Bounding box of businesses reviewed

For each business that was reviewed, the latitude and longitude is collected from the business.json and the maximum and minimum values of both the longitude and latitude are returned. This gives the bounding area of the location of businesses the user has reviewed in Tucson.

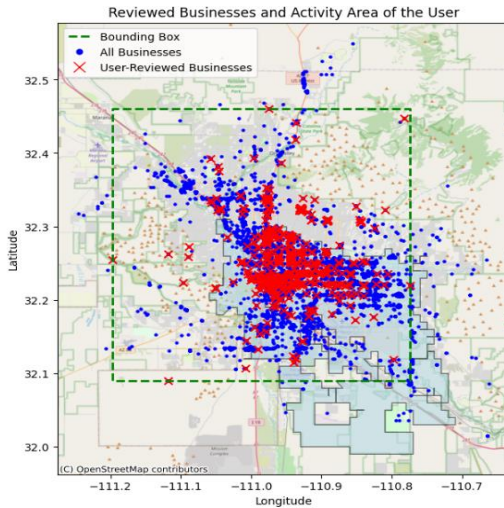


Figure 14: Bounding Box of businesses reviewed

In Figure 14, we visualize the location of all businesses in Tucson as well as the location of businesses reviewed by the user. A base map of Tucson is generated in the background to visualize the geographical shape of Tucson by utilizing OpenStreetMap with the support of the geo data file retrieved from the United States Census Bureau [2]. The bounding box is represented by a green dash line that defines the geographic boundaries that highlights the user's activity area. The businesses in Tucson are represented by blue dots while the businesses reviewed by the user are represented by red cross. For this user, the reviews are mostly concentrated at the centre of Tucson, with a minority of the reviews near to the outskirts of Tucson.

5.2 Top 10 frequently used words

We chose 3 users with different numbers of reviews – 781 reviews (the highest number of reviews), 5 reviews

and 1 review, to have a good representation of user review summary. Table 4 below shows the top 10 most frequent words of 3 users with varying review counts.

Table 4: Top 10 most frequent words from 3 users with varying review counts, excluding stop words.

User ID	Review Count	Word: Frequency	
6ObFF8-uKnOA1XuSH4TlyQ	781	food: 411	good:352
		place: 400	menu: 349
		one: 376	also: 333
		really: 371	time: 322
		like: 360	super: 322
zzeRzizkihWHz9bVAvbcVw	5	service: 5	also: 2
		place: 4	good: 2
		customer: 3	great: 2
		back: 3	atmosphere: 2
		delicious: 2	asked: 2
BUPAhzsbbKbTqyBjT8YnNQ	1	chicken: 2	flavor: 1
		good: 2	generous: 1
		called: 1	amount: 1
		mediterranean: 1	pita: 1
		shawarma: 1	food: 1

5.3 Top 10 frequently used phrases

The top 10 most frequently used phrases were identified without removing stopwords from the tokenized text and different pretrained models were used. A total of 4 pretrained models were used for this experiment. They are Gensim phrases model, RAKE_NLTK phrases model, Textrank phrases model, and the KeyBERT phrases model.

5.3.1 Gensim phrases model

The pretrained Gensim's Phrases Model is from the *gensim.models.phrases* library. For this model, there is a score tied to each of the phrases identified. The higher the score, the more likely the words in the phrase are associated and would likely appear together. This score corresponds to the hyperparameter 'threshold' where the user can specify the phrases produced by the model after scanning all reviews need to be at least of the threshold score. If the threshold value is too low, the phrases identified by the model could have no meaning such as "it is". If the threshold value is too high, real phrases with meanings would be excluded. The range of the hyperparameter needs to be at least 1 and it does not have a maximum value.

The downside to this model is that it is unable to understand punctuation marks or the grammar of English. For example, the word "I'm" will be considered as a phrase as the model sees 'I' and 'm' as separate words. Hence, all phrases containing punctuation marks except for the hyphen (-) are dropped. This is because hyphens can be added to words to make the word an adjective such as "Mother-in-law".

The tuning of the threshold is done manually following the Occam's Razor's approach [3]. This means that the smallest threshold number which can give the top 10 phrases which do not contain any word combinations without meanings will be used. For example, if the threshold from the range of 40 to 100 gives all 10

meaningful phrases which are the same, then the threshold value of 40 will be chosen.

Nevertheless, the team found that the threshold value cannot be generalized for users who have contributed very different numbers of reviews. Referring to the below table, the user 6ObFF8-uKnOAlXuSH4TlyQ has provided 781 reviews which is the greatest number of reviews among other users of Tucson businesses. When tuning the threshold based on that user, the threshold value found suitable for the top 10 meaningful phrases returned is 50. When the threshold value of 50 is applied to other users such as zzeRzizkihWHz9bVAvbcVw who only contributed 5 reviews, two meaningful phrases were returned. This meant that given the small number of reviews by that reviewer, the model cannot identify 10 phrases from it. Nevertheless, when tuning the threshold value for zzeRzizkihWHz9bVAvbcVw specifically, the optimal threshold value is found to be 17, where 4 meaningful phrases were returned. Hence, the genism model does not perform well for our dataset where there are reviewers who gave vastly different numbers of reviews.

Table 5: Comparing performance of Gensim model to users with different number of reviews

Number of Reviews: 781			
Threshold: 50			
Phrase	Count		
my favorite	77		
pick up	59		
super friendly	45		
locally owned	42		
come here	40		
ice cream	40		
my boyfriend	37		
make sure	34		
happy hour	33		
food truck	32		
Number of Reviews: 5			
Threshold: 50			
Threshold: 17			
Phrase	Count	Phrase	Count
be back	3	customer service	3
waiting for 3	2	be back	3
		very good	2
		waiting for 3	2

The difference in optimal threshold value for each user is due to how the scores of the phrases are calculated in the gensim model. The scores of the phrases are calculated by the Equations 2 and 3 below [4]:

$$\ln\left(\frac{\text{prob}(\text{word}_a, \text{word}_b)}{\text{prob}(\text{word}_a) \times \text{prob}(\text{word}_b)}\right) - \ln(\text{prob}(\text{word}_a, \text{word}_b)) \quad (2)$$

$$\text{where } \text{prob}(\text{word}) = \frac{\text{word_count}}{\text{corpus_word_count}} \quad (3)$$

If a phrase such as “has been” keeps appearing in the reviews of that user due to its use as the sentence’s grammar, “has been” would be captured as a phrase with a high score by the gensim model even if “has been” does not have any meaning. This would pose a problem to

users who contributed little reviews as they do not have many meaningful phrases repeating in their reviews.

For the user with 5 reviews, there are other meaningful phrases used in his/her review and are uncaptured in the top 10 phrases as those phrases appeared once, causing the probability of the word appearing with other words to be small. Hence, these meaningful phrases were uncaptured as a phrase with a threshold value of 17. Nevertheless, if a smaller threshold of 16 is used, it will lead to meaningless phrases such as “I will” to be captured. The problem is inflated as most reviewers provide little reviews as shown in Figure 15 where 89.3% of the reviewers contributed at most 5 reviews.

Percentage of Reviewers by Number of Reviews given

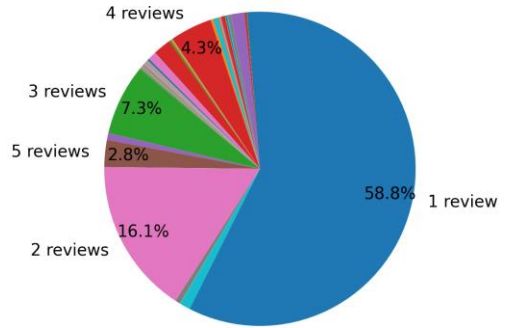


Figure 15: Distribution of reviews by reviewers.

Since genism model do not perform well to our dataset, the team will be evaluating other phrase models. The RAKE, Textrank, and KeyBert phrase models will be assessed next. Although these models can identify both keywords and phrases, they have been configured to return only phrases with more than one word as the top 10 most frequent phrases. Stop words are configured not to be removed for all three phrase models.

5.3.2 RAKE_NLTK phrases model

RAKE (Rapid Automatic Keyword Extraction) from the *rake_nltk* library is an algorithm created to automatically capture the most important phrases from a text. The sentences are separated with punctuation marks as the delimiters. Hence, sentences such as “After I ran 5km, I rested for a good 30 minutes.” would be broken down to “After I ran 5km”, “I rested” and “good 30 minutes”. Next, a co-occurrence matrix is built where the matrix record shows how frequently each word appears together in the same phrase [5]. The score of each word is calculated by considering the number of times the word appeared in the reviews and how frequently the word appears beside other words. With these scores, the words are grouped together where key phrases are identified.

5.3.3 Textrank phrases model

The Textrank phrase model from the *spacy* library using the *pytextrank* extension preprocesses texts by converting letters to lowercase and removing punctuations before assigning POS tags to each word to understand the dependency parsing of the sentences.

Important words with POS tags such as ‘ADJ’ (adjective) or ‘NOUN’ are kept while other words are dropped. A graph is created where words are nodes and words co-occurring with other words are connected by edges. The more frequently the words appear together, the bigger the weight of the edges. The score of the relevance of each word is calculated as per the Equation 4 below. [6]

$$S(V_i) = (1 - d) + d \times \sum_{j \in \ln(V_i)} \frac{1}{|\text{out}(V_j)|} S(V_j) \quad (4)$$

Where $S(V_i)$ is the score of the node (word), d is the damping factor meaning the probability of the nodes teleporting from one node to the other (the default 0.85 was used), $\ln(V_i)$ is the set of nodes connected and $L(V_i)$ is the number of edges on the node. The calculated method resembles that of PageRank where the scores are calculated every iteration till convergence. Key phrases are formed by linking words with high scores together.

5.3.4 KeyBert phrases model

The KeyBert phrases model uses the *KeyBert* library which uses the BERT (Bidirectional Encoder Representations from Transformers) to capture key phrases from the reviews [7]. The BERT model is pre-trained to be able to provide context embeddings to each word by using the word appearing at its left and right side for context. Vectors are generated to represent each word. Potential phrases are generated using n-gram methods. The range of words to be considered as phrases can be configured and is defaulted to 2 to 3 words. Cosine similarity is calculated for each potential phrase to generate which phrases are more relevant.

5.3.5 Phrases model conclusion

Since more than half of the reviewers only provide 1 review as per figure 15 above, it is important to evaluate if the three phrase models perform well to reviewers with 1 review.

Table 6: Comparing top 10 phrases by RAKE_NLTK, Textrank and KeyBert phrase models based on the user BUPAhzsbbKbTqyBjT8YnNQ with only 1 review.

RAKE_NLTK Model			
Index	Phrases	Count	Score
1	mediterranean chicken shawarma	1	8
2	would revisit	1	4
3	next time	1	4
4	good service	1	4
5	good flavor	1	4
Textrank Model			
1	good flavor	1	0.1659
2	good service	1	0.1659
3	a mediterranean chicken shawarma	1	0.1117
4	the area	1	0.0544
5	the food	1	0.0544
6	the amount	1	0.0513
7	the pita	1	0.0513

KeyBert Model			
1	mediterranean chicken shawarma	1	0.7267
2	chicken shawarma	1	0.7002
3	in mediterranean chicken	1	0.6860
4	mediterranean chicken	1	0.6816
5	chicken shawarma good	1	0.6699
6	shawarma good flavor	1	0.5733
7	pita the food	1	0.5381
8	chicken in the	1	0.5313
9	called in mediterranean	1	0.5068
10	chicken in	1	0.5015

The table above shows the top phrases produced by different models sorted by frequency then by its scores for the reviewer BUPAhzsbbKbTqyBjT8YnNQ who has only 1 review. For the RATE_NLTK and Textrank models, we observe that less than 10 phrases are listed. For the RAKE_NLTK model, all phrases are meaningful. For the Textrank model, although all phrases are meaningful, phrases such as “the area”, “the food” and “the amount” provided little information. For the KeyBert model, although it gave 10 phrases, most phrases contain overlapping words such as “in mediterranean chicken” and “mediterranean chicken”. This is expected as the model identifies the phrases using an n-gram method. Moreover, not all phrases are meaningful such as “called in mediterranean”, which can explain why it is scored the lowest in the KeyBert model.

To analyze if the models can generalize to reviewers with many reviews, the user with the greatest number of reviews (781 reviews) is chosen and the top 10 phrases produced by the three models is displayed in Table 7. It is observed that both the RAKE_NLTK and Textrank model identified 10 phrases which are meaningful while the KeyBert model identified some meaningless phrases such as “local beer loving”.

Table 7: Comparing top 10 phrases by RAKE_NLTK, Textrank and KeyBert phrase models based on the user 60bFF8-uKnOAlXuSH4TlyQ with 781 reviews.

RAKE_NLTK Model			
Index	Phrases	Count	Score
1	super friendly	39	4.169
2	customer service	33	4.528
3	make sure	30	3.284
4	live music	24	3.660
5	last visit	23	3.941
6	go wrong	22	3.449
7	go back	18	3.387
8	locally owned	17	5.792
9	ice cream	17	5.553
10	old pueblo	17	5.542
Textrank Model			
1	nice food food menu	1	0.046
2	good food	1	0.045
3	good service	1	0.043
4	crazy good food	1	0.042
5	more menu items	1	0.041
6	fresh menu items	1	0.041
7	fresh home made cornbread	1	0.041
8	more places	1	0.040
9	great place	1	0.040
10	more time	1	0.040

KeyBert Model			
1	brewery for visit	1	0.587
2	local arizona beer	1	0.582
3	beer options	1	0.579
4	local beers and	1	0.578
5	the beer options	1	0.572
6	beer selection seems	1	0.563
7	local beer loving	1	0.563
8	beers and locally	1	0.561
9	brewery pub vibe	1	0.554
10	arizona beer	1	0.551

Comparing the RAKE_NLTK model to other models, RAKE_NLTK identified phrases which appeared many times while the other two models only identified phrases appearing once. Hence, it can be concluded that for the criterion of “most frequent phrases”, RAKE_NLTK is more suitable. It is noted that the KeyBert model took a long time to run (12 minutes) compared to the other two models which took less than 1 minute. Although this is expected as more word combinations are considered as the word length from the total number of reviews increase, the KeyBert phrase model is not wise to be used for users with many reviews.

Upon further testing by the team on other users, we conclude that for the reviews of Tucson given by Yelp, the RAKE_NLTK Model is most suitable phrase model as it can identify the most frequently used meaningful phrases with useful information and can generalize well to users with both large and small numbers of reviews.

5.4 Top 3 most representative sentences

The most representative sentences of a user are defined as the top 3 sentences with the highest TF-IDF score. For the top 3 most representative sentences, the Term Frequency-Inverse Document Frequency (TF-IDF) score is used, where each document is one review by the user and a collection is all the reviews contributed by the user. To determine the 3 most representative sentences, the reviews are tokenized and stop words are removed from each review. Part-of-Speech tagging is added to the words and the words were then lemmatized to reduce each word to their root word. In this case, TF has a slightly different method of calculation. TF of each word is calculated by counting how many times that word appeared in each sentence instead of each review. Document Frequency (DF) for each word is calculated by counting how many reviews contain that word and the IDF and TF-IDF are subsequently calculated.

Tables 8 and 9 show the top 3 sentences from 3 users with varying review counts. Looking at table 8, the first sentence is very long as the algorithm segregates sentences by the full stop. Hence, the length of a sentence depends on the writing style of the reviewer such as how frequently they add full stop to their sentences. One interesting observation in Table 9 for user BUUpAhzsbbKbTqyBjT8YnNQ with 1 review count is that as there is only one review which corresponds to one document, the resulting TF-IDF score will be zero as there are no other documents in the collection to compare against.

Table 8: Top 3 most representative sentences for user with 781 reviews

User_ID	Review Count	Sentence	TF-IDF Score
6ObFF8-uKnOAl XuSH4T lyQ	781	I give this joint a solid 5 stars...and there are so many reasons I'll just list them: -Kitchen is so clean you lick the floor -Fresh Fresh Fresh Carne Asada is always rotated on the grill top -Consistently ripe and delicious Avocados in Stock -BEST QUESADILLA IN TOWN: (besides my own or my moms): These guys use the legit Sonoran Flour Tortilla that gets Crispy and Flaky, Not Chewy and Flavorless like the Gringo types served at Guadalajara Grill.	156.0194
		The calamari was my fave, the chips and guacamole were your standard dish and for dinner I ordered the Southwestern Steak Salad served with Arugula, spinach, poblanos, toasted pepitas queso fresco in a sliced bistro tender steak, shaved red onion, roasted poblanos, baby roma tomatoes, toasted pepitas, queso fresco and cilantro lime vinaigrette.	154.0511
		There are ten different concepts to choose from: Avenue Mexican, Isabella's Ice Cream, Upper Crust Pizza, OPA Greek, AZ Rib House, The Bite (burgers), Cafe con Leche (coffee shop), Dumb Fish (Poke), Dos Amigos meat market (where you can buy tortillas, meat from the butcher and other Mexican goodies!	142.6133

Table 9: Top 3 most representative sentences for 2 users with 5 reviews and 1 review

User_ID	Review Count	Sentence	TF-IDF Score
zzeRzizk ihWHz9 bVAvbc Vw	5	Very dissatisfied at the customer service not to mention the service that never happen after waiting for 3 hours an 8 o'clock appointment that turn out to waiting for 3 hours and many phone calls in between to what was happening many excuse just not acceptable will never use your service ever again very disappointed	41.5861
		We had about 5 different servers and when asked how everything was and I replied "I don't really like my meal" I was told "oh okay."	18.6201
		I tried the famous hot dog and it was delicious, the beans gave it just the right texture.	14.4849
BUUpAhzsbbKbTqyBjT8YnNQ	1	Called in a Mediterranean chicken shawarma.	0
		Good flavor and they were generous with the amount of chicken in the pita.	0
		The food was ready to go when I arrived.	0

6 Application

For some users, the stars may be insufficient information to decide which business to visit. The stars reflect a culmination of multiple aspects of the business - from customer service to item selection. For example, the stars are not enough information for users who want to find a relaxing café as it does not reflect the ambience of the business. For a business owner wanting to make decisions to improve the business, looking through all

the reviews is not the most efficient way to gain actionable insights into what resonates with customers and what might need refinement. The best way to get a sense of this in an efficient way is sentiment analysis.

We have designed a simple command-line interface (CLI) application to help users discover, analyze, and understand customer sentiments for businesses in an efficient, user-friendly way. By combining Whoosh for robust search capabilities and NLTK's sentiment analysis tools, this application enables users to search for businesses by name or category, sort results by either review count or score, and delve deeper into customer feedback. This functionality allows users to quickly assess a business's reputation based on authentic customer reviews, making it an invaluable tool for consumers, business analysts, and marketers alike.

The core of the application's sentiment analysis relies on NLTK's VADER model, which is optimized for social media and customer reviews because of its ability to detect nuanced sentiments like positive, neutral, and negative tones. [8] For each business, the application categorizes customer reviews by sentiment and extracts common 3-gram phrases from each category. This phrase extraction approach provides users with a clear view of recurring themes, such as frequently mentioned praises, complaints, or general observations. By showcasing these key phrases, the application not only conveys overall customer sentiment but also pinpoints specific aspects of the business experience that customers frequently discuss.

One of the application's main impacts is how it simplifies the otherwise time-consuming process of sifting through customer reviews manually. With this tool, users can access summarized, sentiment-categorized data at a glance, allowing them to obtain a well-rounded view of any business's customer experience. By offering search and sorting options, the application further enables users to prioritize businesses based on their needs, whether they are looking for highly rated businesses or the most popular spots by review count. This functionality is especially beneficial for users who want to quickly gather insights without reading through countless reviews.

A practical example of this functionality can be seen from Figure 16 when searching for "ramen" restaurants. By selecting "best review score" as the sorting criterion, the user is presented with a list of top ramen spots sorted by their average ratings. Upon selecting a restaurant, such as "Ikkyu," the application performs sentiment analysis on the reviews, categorizing 510 reviews as positive, 35 as negative, and 9 as neutral. It then displays the top phrases in each category, such as "best ramen tucson" and "super hiro roll" for positive reviews, indicating popular menu items and aspects of the customer experience. This phrase extraction feature allows users to go beyond raw review scores to understand what specifically attracts customers to a business or detracts from their experience.

```
Enter a business name or keyword (e.g., 'sushi'): ramen

How would you like the results sorted?
1. Most Review Count
2. Best Review Score
Enter 1 or 2: 2

Top 10 Results:
1. Raijin Ramen - 2995 E Speedway
   Review Count: 594, Review Score: 4.0
2. Mian Sichuan - 4695 N Oracle Rd, Ste 105
   Review Count: 126, Review Score: 4.5
3. Ikkyu - 2840 W Orange Grove Rd, Ste 180
   Review Count: 511, Review Score: 4.5
4. Jimmy's Pita & Poke - 845 E University Blvd, Ste 175
   Review Count: 183, Review Score: 4.5
5. JA Ramen Curry - 2643 N Campbell Ave
   Review Count: 153, Review Score: 4.0
6. Toss-Fried Chicken and Ramen - 1655 S Alvernon Way
   Review Count: 79, Review Score: 4.5
7. Toss - 1655 S Alvernon Way
   Review Count: 30, Review Score: 4.5
8. Kiyami Ramen - 4610 E Speedway Blvd
   Review Count: 149, Review Score: 4.0
9. K Japanese Restaurant - 2962 N Campbell Ave
   Review Count: 166, Review Score: 4.0
10. Sushi Zona - 5655 E River Rd, Ste 151
   Review Count: 122, Review Score: 4.0

Enter the number of the business you want to analyze (1-10): 3

Analyzing reviews for Ikkyu...

Business Name: Ikkyu
Business Categories: Restaurants, Ramen, Japanese
Business Reviews Score: 4.5

Positive Reviews: 510
Top Positive Phrases: [('super hiro roll', 6), ('spicy tuna top', 6),
('wo n't disappointed', 6), ('best ramen tucson', 6)]

Negative Reviews: 35
Top Negative Phrases: [('restaurant food die', 1), ('cooks looked
crazy', 1), ('looked crazy questioning', 1), ('crazy questioning
food', 1)]

Neutral Reviews: 9
Top Neutral Phrases: [('spicy tuna volcano', 2), ('tuna volcano
style', 2), ('volcano style eat', 1), ('style eat least', 1)]
```

Figure 16: Example result from our CLI application

Looking ahead, we could explore other features to enhance the application's functionality. Adding more granular sentiment categories or integrating a machine learning model to refine sentiment analysis further could make the insights even more accurate and personalized. Additionally, expanding n-gram phrase extraction to include 4-grams would provide more detailed insights into customer feedback. The application could also benefit from visual summaries of sentiment trends over time, allowing users to track whether a business's customer satisfaction is improving or declining. These future enhancements would make the tool even more versatile and valuable to a broader range of users.

7 Conclusion

In this report, the data related to the city of Tucson by Yelp has been analyzed. Data sample, tokenization and stemming of the dataset was first conducted. Next, a search engine allowing users to search by business names, keywords of reviews and geospatially was built. The search engine was then integrated to provide a Review Summary where details such as the number of reviewers provided, activity area, top 10 most frequently used words and phrases, and top three most representative sentences of the user_id being input is returned. Lastly, the team built an application which targets both customers and businesses to allow them to analyze the sentiment reviews of the Tucson businesses.

APPENDIX

Attribute Name	Type	Description	Values
business_id	String	The unique id of the business	Eg. "tUFRWirKiKi_TAnsVWINQQ"
name	String	The business's name	Eg. "Target"
address	String	Full address of the business	Eg. "5255 E Broadway Blvd"
city	String	City the business is located in	Eg. "Tucson"
state	String	State Code of the business	Eg. "AZ"
postal_code	String	Postal Code of the business	Eg. "85711"
latitude	float	Latitude of the business	Eg. "32.223236"
longitude	float	Longitude of the business	Eg. "-110.880452"
business_rating	float	Star rating of the business	Between 1 to 5 Eg. "3.5"
review_count	integer	Number of review for the business	Eg. "22"
Is_open	integer	Whether the business is open or closed	{0 = closed , 1 = open} Eg. "0"
attributes	String	An array of strings that contains the business attributes	Eg. "{ 'BikeParking': 'True', 'BusinessAcceptsCreditCards': 'True', 'RestaurantsPriceRange2': '2', 'CoatCheck': 'False', 'RestaurantsTakeOut': 'False', 'RestaurantsDelivery': 'False', 'Caters': 'False', 'WiFi': 'u'no'', 'BusinessParking': '{ 'garage': False, 'street': False, 'validated': False, 'lot': True, 'valet': False }, 'WheelchairAccessible': 'True', 'HappyHour': 'False', 'OutdoorSeating': 'False', 'HasTV': 'False', 'RestaurantsReservations': 'False', 'DogsAllowed': 'False', 'ByAppointmentOnly': 'False' }"
categories	String	An array of strings that contains the type of business categories the business is in	Eg. "Department Stores, Shopping, Fashion, Home & Garden, Electronics, Furniture Stores"
hours	String	An array of strings that contains the opening hours of the business	Eg. "{ 'Monday': '8:0-22:0', 'Tuesday': '8:0-22:0', 'Wednesday': '8:0-22:0', 'Thursday': '8:0-22:0', 'Friday': '8:0-23:0', 'Saturday': '8:0-23:0', 'Sunday': '8:0-22:0' }"
review_id	String	The unique id of the review	Eg. "IOmiYoBPtQsY_fh5uA4mXg"
user_id	String	The unique id of the user who wrote the review	Eg. "P-NT0AMFVSDFGkhcj4GaIQ"

review_rating	String	The star rating given by the user to the business	Between 1 to 5 Eg. "4.0"
useful	Integer	Number of votes received that the review is "Useful"	Eg. "1"
funny	Integer	Number of votes received that the review is "Funny"	Eg. "1"
cool	Integer	Number of votes received that the review is "Cool"	Eg. "1"
text	String	The review description given by the user	Eg. "We are fans of Target. They seem to have a little better quality over Walmart. This Target is easy to get to from the U of A. When we have errand to run while we are over by the university we tend to stop by and pick up what we need here.\n\nThis Target happens to be one of the oldest and smallest in town. That being said they tend to have a smaller selection. The cafe now seems to be closed every time we go in. Be careful when schools starts this place is packed. On the plus side the restrooms are clean and the staff is friendly."
date	String	The date of the review	Eg. "2017-02-19 15:11:22"

REFERENCES

- [1] Whoosh Documentation, Whoosh 2.7.4, Whoosh.readthedocs.io, 2023. [Online]. Accessed: October 31, 2024 from <https://whoosh.readthedocs.io/en/latest/>
- [2] United States Census Bureau, Maps, Explore Census data. [Online] Accessed: October 31, 2024 from https://data.census.gov/map?q=Tucson%20city,%20Arizona&layer=VT_2020_160_00_PY_D1&loc=32.1467,-110.7604,z10.4217.
- [3] T. F. Smith, "Occam's razor," *Nature*, vol. 285, pp. 620–620, 1980. Accessed October 31, 2024 from <https://www.oxfordreference.com/display/10.1093/oi/authority.20110803100244343>
- [4] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," *Neural Inf Process Syst*, pp. 3111–3119, 2013.
- [5] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," *Text Mining*. Wiley, pp. 1–20, March 26, 2010.
- [6] "TextRank: Bringing Order into Texts Rada Mihalcea and Paul Tarau Department of Computer Science University of North Texas ✉ rada, tarau @cs."
- [7] M. P. Grootendorst, "KeyBERT," *Github.io*. [Online]. Accessed: October 26, 2024 from <https://maartengr.github.io/KeyBERT/>
- [8] C. J. Hutto and E. Gilbert, "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text," in *Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media (ICWSM-14)*, Ann Arbor, MI, USA, June 2014, pp. 216–225. [Online]. Accessed: October 31, 2024, from <http://eegilbert.org/papers/icwsm14.vader.hutto.pdf>