Lexicon, Forward and Inverted Index Submission

Name	CMS
Ali Aqdas	453852
AbdurRehman Shafique	453956
Muhammad Raza	456326

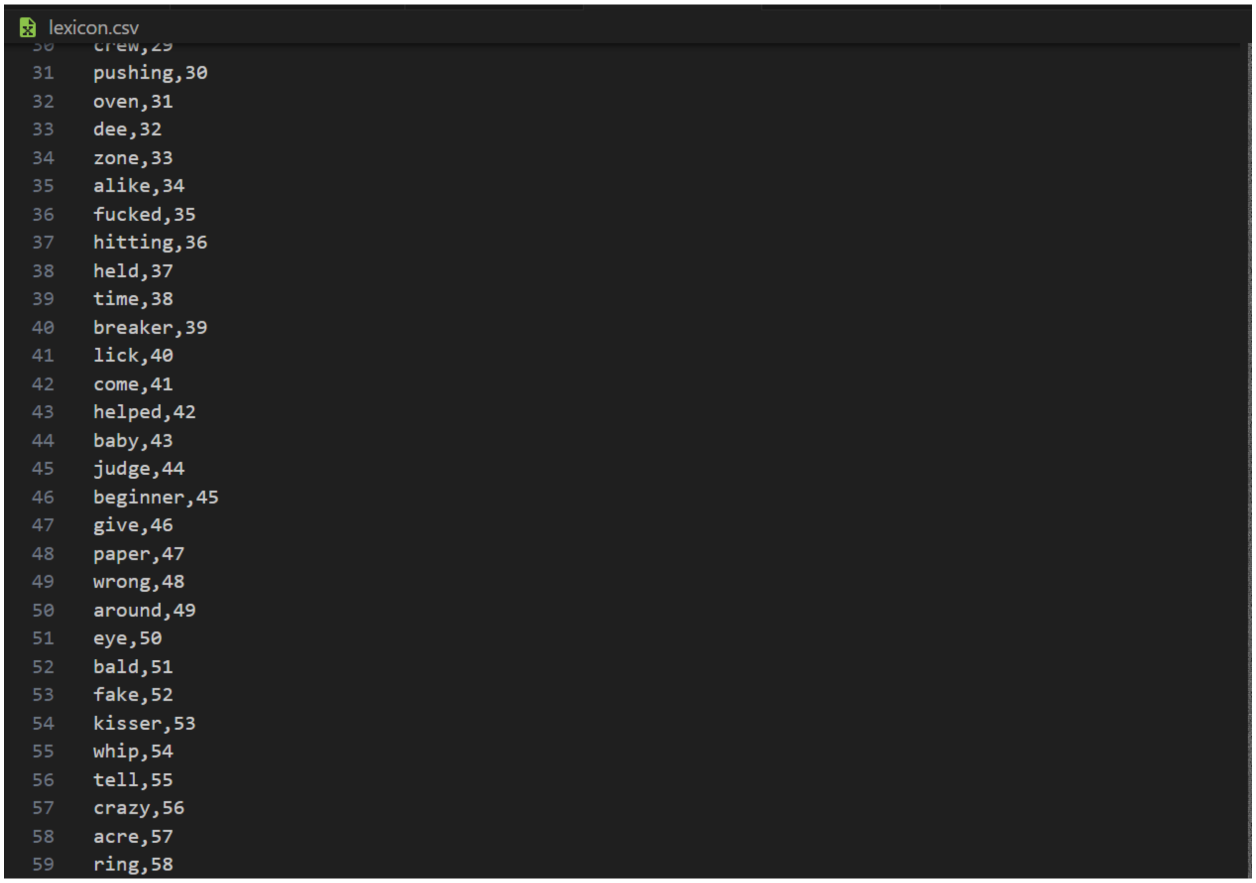
Github Repository URL: https://github.com/Aqdasali09/DSA-Project

Lexicon

Code:

```
import re
import nltk
from collections import defaultdict
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

# Download required NLTK data if not already downloaded(uncomment it)
# nltk.download('punkt')
# nltk.download('stopwords')
# nltk.download('wordnet')
# nltk.download('punkt_tab')

def preprocess_text(text):
    """
    Preprocess text: lowercase, remove punctuation, tokenize, remove stopwords,
and lemmatize.
    """
    if not isinstance(text, str):
        return []
    text = text.lower()
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
    tokens = word_tokenize(text)  # Tokenize
    tokens = [word for word in tokens if word not in stopwords.words('english')]
# Remove stopwords

    # Initialize lemmatizer
    lemmatizer = WordNetLemmatizer()

    # Lemmatize each token
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    return tokens

def create_lexicon(data, searchable_fields):
    """
    Create a lexicon with unique IDs for each word.
    :param data: List of dictionaries containing dataset rows.
    :param searchable_fields: List of fields to include in the lexicon.
    :return: Lexicon dictionary with terms as keys and unique IDs as values.
    """
    lexicon = {}  # Store words and their unique IDs
    id_counter = 1  # Start IDs from 1

    for row in data:
        for field in searchable_fields:
            tokens = preprocess_text(row.get(field, ""))
            for token in set(tokens):  # Avoid processing duplicate words in the
same row
                if token not in lexicon:
                    lexicon[token] = id_counter  # Assign a unique ID to the
token
                    id_counter += 1  # Increment the ID counter

    return lexicon
```

Output:



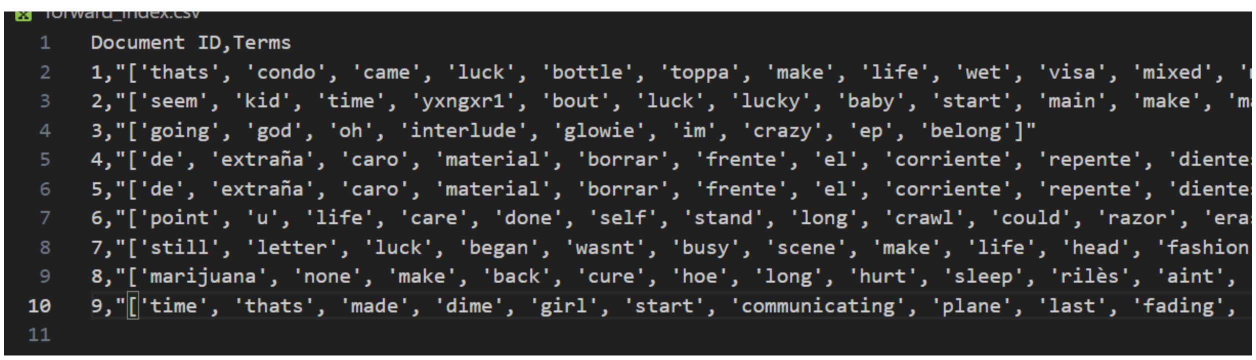
```
crew,29
pushing,30
oven,31
dee,32
zone,33
alike,34
fucked,35
hitting,36
held,37
time,38
breaker,39
lick,40
come,41
helped,42
baby,43
judge,44
beginner,45
give,46
paper,47
wrong,48
around,49
eye,50
bald,51
fake,52
kisser,53
whip,54
tell,55
crazy,56
acre,57
ring,58
```

Forward Index

Code:

```python
from lexicon import preprocess_text

def create_forward_index(data, searchable_fields):
    """
    Create forward index for each document based on searchable fields,
    using integer-based document IDs that increment.

    :param data: List of dictionaries containing dataset rows.
    :param searchable_fields: List of fields to include in the forward index.
    :return: Forward index as a dictionary mapping document IDs (integers) to
tokens.
    """
    forward_index = {}
    doc_id_counter = 1  # Start document ID from 1

    for row in data:
        doc_id = doc_id_counter  # Assign current integer document ID
        all_tokens = []

        # Process all searchable fields for the document
        for field in searchable_fields:
            all_tokens.extend(preprocess_text(row.get(field, "")))

        forward_index[doc_id] = list(set(all_tokens))  # Avoid duplicate tokens
        doc_id_counter += 1  # Increment document ID for the next document

    return forward_index
```

Output:



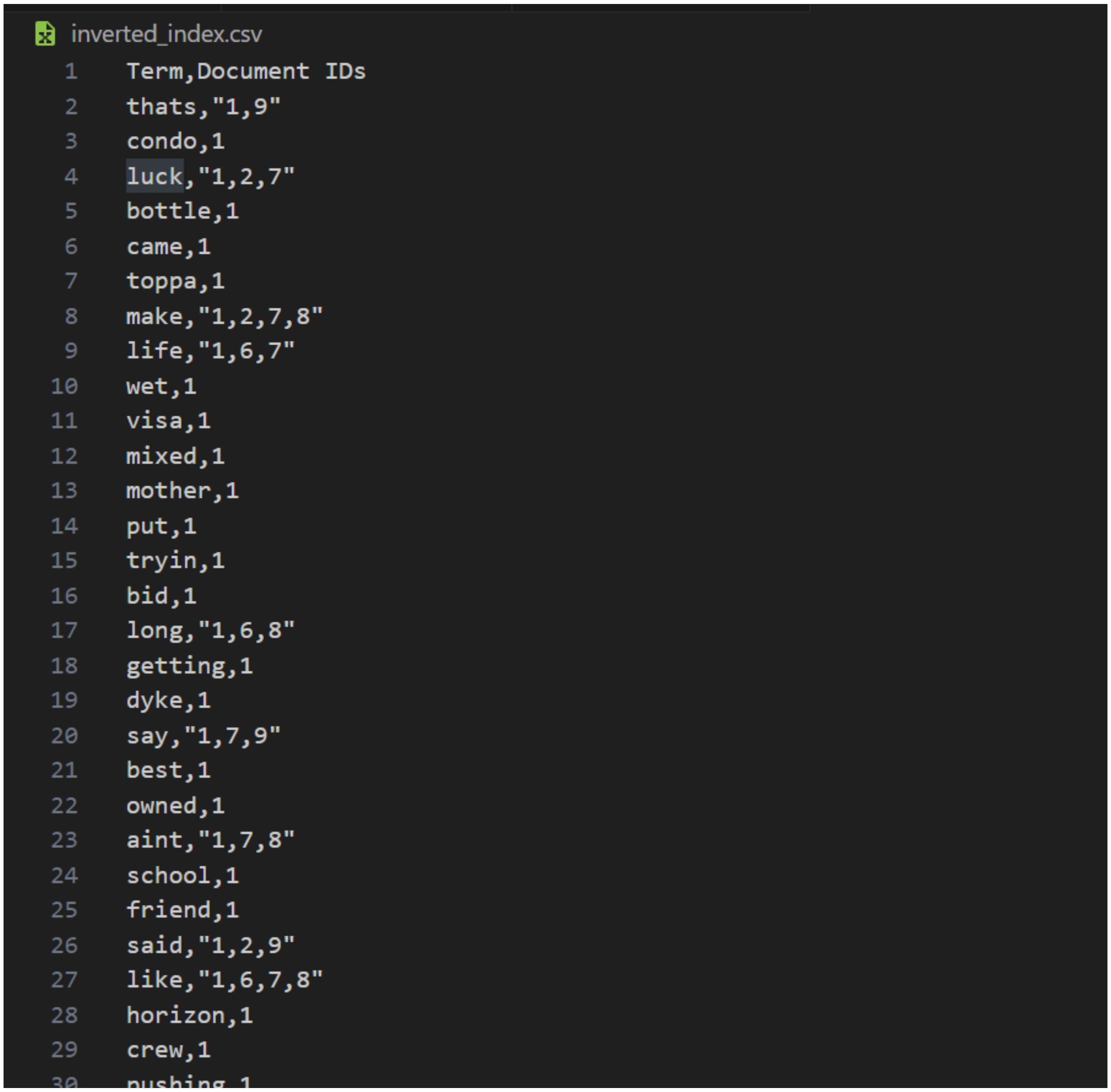
```
forward_index.csv
Document ID,Terms
1,"['thats', 'condo', 'came', 'luck', 'bottle', 'toppa', 'make', 'life', 'wet', 'visa', 'mixed', '
2,"['seem', 'kid', 'time', 'yxngxr1', 'bout', 'luck', 'lucky', 'baby', 'start', 'main', 'make', 'm
3,"['going', 'god', 'oh', 'interlude', 'glowie', 'im', 'crazy', 'ep', 'belong']"
4,"['de', 'extraña', 'caro', 'material', 'borrar', 'frente', 'el', 'corriente', 'repente', 'diente
5,"['de', 'extraña', 'caro', 'material', 'borrar', 'frente', 'el', 'corriente', 'repente', 'diente
6,"['point', 'u', 'life', 'care', 'done', 'self', 'stand', 'long', 'crawl', 'could', 'razor', 'era
7,"['still', 'letter', 'luck', 'began', 'wasnt', 'busy', 'scene', 'make', 'life', 'head', 'fashion
8,"['marijuana', 'none', 'make', 'back', 'cure', 'hoe', 'long', 'hurt', 'sleep', 'rilès', 'aint',
9,"['time', 'thats', 'made', 'dime', 'girl', 'start', 'communicating', 'plane', 'last', 'fading',
```

Inverted Index

Code:

```
import csv
from collections import defaultdict

class InvertedIndex:
    def __init__(self):
        # Dictionary to store the inverted index. Default is an empty list.
        self.index = defaultdict(list)

    def build(self, forward_index):
        """
        Build the inverted index from the forward index.
        :param forward_index: Dictionary mapping document IDs to lists of terms.
        """
        # Iterate over all documents and their terms to build the inverted index
        for doc_id, terms in forward_index.items():
            # Use set to avoid processing duplicate terms in the same document
            for term in set(terms):
                self.index[term].append(doc_id)

    def save_to_csv(self, file_path):
        """
        Save the inverted index to a CSV file.
        :param file_path: Path to the CSV file.
        """
        # Prepare rows for CSV file: each row contains a term and the
corresponding document IDs as a comma-separated string.
        with open(file_path, mode='w', newline='', encoding='utf-8') as file:
            writer = csv.writer(file)
            writer.writerow(["Term", "Document IDs"])  # Writing header

            for term, doc_ids in self.index.items():
                writer.writerow([term, ",".join(map(str, doc_ids))])  # Writing
term and doc_ids

    def search(self, term):
        """
        Search for a term in the inverted index.
        :param term: The term to search for.
        :return: List of document IDs containing the term.
        """
        # Return the list of document IDs that contain the term or an empty list
if not found
        return self.index.get(term, [])
```

Output:



inverted_index.csv

```
Term,Document IDs
thats,"1,9"
condo,1
luck,"1,2,7"
bottle,1
came,1
toppa,1
make,"1,2,7,8"
life,"1,6,7"
wet,1
visa,1
mixed,1
mother,1
put,1
tryin,1
bid,1
long,"1,6,8"
getting,1
dyke,1
say,"1,7,9"
best,1
owned,1
aint,"1,7,8"
school,1
friend,1
said,"1,2,9"
like,"1,6,7,8"
horizon,1
crew,1
pushing,1
```