Generative AI Disclosure: Homework 3

1. tf-idf definition

Instance 1.

Asked ChatGPT how to write the reduce_phase function in the MapReduce Functions and how to iterate through the documents given this shuffle_and_sort function:

```
def shuffle_and_sort(mapped_data): #mapped_data is pairs from map phase
   term_dict = {} #empty dictionary
   for term, doc_tf in mapped_data:
       term_dict.setdefault(term, []).append(doc_tf)
   return term_dict #dictionary with key being a word, and value being a list of list of (doc_id, tf)
```

It gave me this chunk of code, which I commented:

```
#Computes the TF-IDF score for each term in each document
def reduce_phase(shuffled_data, total_docs):
   tf_idf_results = {} #empty dictionary
   for term, doc_list in shuffled_data.items():
       doc_freq = len(doc_list)
       idf = math.log(total_docs / doc_freq) #compute inverse document frequency
       for doc_id, tf in doc_list:
           tf_idf_results.setdefault(doc_id, {})[term] = tf * idf
   return tf_idf_results #dictionary with key being the document id, and a dictionary of pairs for the document
```

I was confused about the .setdefault function so I asked ChatGPT to explain it, and it gave me the following functionally equivalent code to explain it:

```
if doc_id not in tf_idf_results:
   tf_idf_results[doc_id] = {}
tf_idf_results[doc_id][term] = tf * idf
```

With the explanation:

After `setdefault` returns the dictionary for that `doc_id`, this part sets the TF-IDF score for the specific `term` in that document.

Instance 2.

Asked ChatGPT: How do I save the measures in a new column?

Was able to generate the following code (which I commented with my general understanding):

```
from pyspark.sql import Row #import row

rows = [Row(doc_id=doc_id, tfidf=tfidf_dict) for doc_id, tfidf_dict in
tf_idf_matrix.items()] #create list of row objects with document id and
dictionary with tfidf

tfidf_df = spark.createDataFrame(rows) #convert list of rows into spark
dataframe

from pyspark.sql.functions import col #import col

agnews = agnews.withColumnRenamed("_c0", "doc_id") #rename _c0 column with
doc_id
agnews = agnews.withColumn("doc_id", col("doc_id").cast("string"))
#ensures doc_id column is a string

agnews_with_tfidf = agnews.join(tfidf_df, on="doc_id", how="left")
#perform a left join on doc_id
```

I had to go through multiple iterations of this code due to the confusion between the column being auto named _c0 versus the actual title of doc_id. This was shown in the following error:

```
---------------------------------------------------------------------------
-
AnalysisException                         Traceback (most recent call
last)
<ipython-input-15-03ba9a401627> in <cell line: 0>()
      9 agnews = agnews.withColumn("_c0", col("_c0").cast("string"))
     10
---> 11 agnews_with_tfidf = agnews.join(tfidf_df, on="_c0", how="left")

/usr/local/lib/python3.11/dist-packages/pyspark/errors/exceptions/captured
.py in deco(*a, **kw)
    183                 # Hide where the exception came from that shows a
non-Pythonic
    184                 # JVM exception message.
--> 185                 raise converted from None
    186             else:
    187                 raise

AnalysisException: [UNRESOLVED_USING_COLUMN_FOR_JOIN] USING column `_c0`
cannot be resolved on the right side of the join. The right-side columns:
[`doc_id`, `tfidf`].
```

I therefore inputted this again into ChatGPT which said:

It means that Spark can't find `_c0` in the `tfidf_df` DataFrame — because the column there is called `doc_id`. You need the join column names to **match exactly** on both sides. So, update the column name in `agnews`.

It gave me the following new code:

```
from pyspark.sql.functions import col

# Rename _c0 to doc_id and cast to string to match tfidf_df
agnews = agnews.withColumnRenamed("_c0", "doc_id")
agnews = agnews.withColumn("doc_id", col("doc_id").cast("string"))

# Now join will work
agnews_with_tfidf = agnews.join(tfidf_df, on="doc_id", how="left")
```

Instance 3.

Received the error: NameError: name 'doc_id' is not defined

When addressing this question on the homework: Print out the tf-idf measure for the first 5 documents.

I used the GoogleColab function "Explain Error" which suggested I change my code from:

```
for doc_id[:5], term_scores in tf_idf_matrix.items(): #for first 5
documents
   print(f"\nDocument: {doc_id}") #print document id
   for term, score in sorted(term_scores.items(), key=lambda x:
-x[1])[:5]:
       print(f"  {term}: {score:.4f}") #print tf-idf measure
```

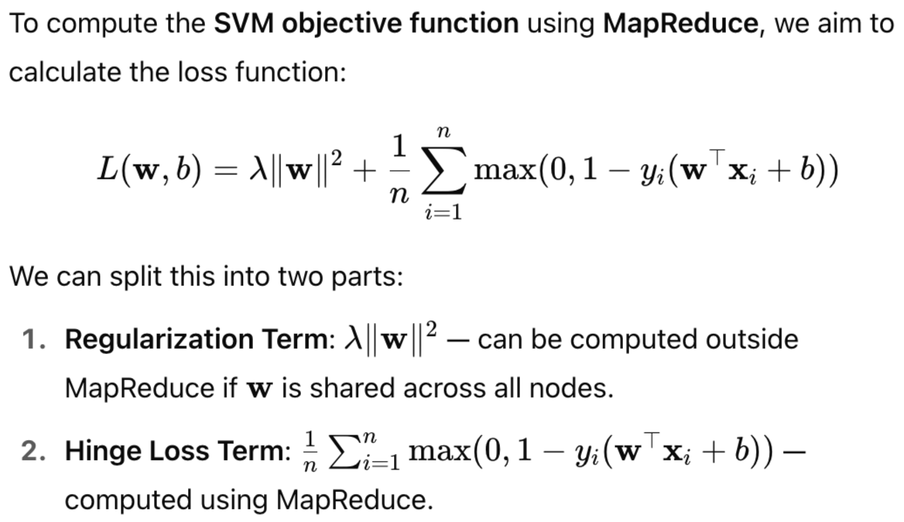
To:

```
for doc_id, term_scores in list(tf_idf_matrix.items())[:5]: #for first 5
documents
   print(f"\nDocument: {doc_id}") #print document id
   for term, score in sorted(term_scores.items(), key=lambda x:
-x[1])[:5]:
       print(f"  {term}: {score:.4f}") #print tf-idf measure
```

Which ran correctly.

2. SVM Objective Function

Instance 1.
I wanted to break down the SVM objective function, since I was confused on how to start writing the map reduce functions due to a lack of understanding of the objective function. ChatGPT gave the following output:



To compute the **SVM objective function** using **MapReduce**, we aim to calculate the loss function:

$$L(\mathbf{w}, b) = \lambda \|\mathbf{w}\|^2 + \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b))$$

We can split this into two parts:

1. **Regularization Term:** $\lambda \|\mathbf{w}\|^2$ — can be computed outside MapReduce if $\mathbf{w}$ is shared across all nodes.
2. **Hinge Loss Term:** $\frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b))$ — computed using MapReduce.

Which helped me understand how to start the code.

Instance 2.
I was confused on how to write the reduce_phase function, which computes the final SVM loss, while using lambda as an inputted variable. I asked ChatGPT to write the reduce_phase function given the following shuffle_and_sort function:

```
def shuffle_and_sort(mapped_data):
   grouped = {}
   for key, value in mapped_data:
       grouped.setdefault(key, []).append(value)
   return grouped
```

ChatGPT gave me the following code as an output:

```
#Computes final SVM loss
def reduce_phase(shuffled_data, total_docs, w, λ):
   hinge_losses = shuffled_data.get("hinge_loss", [])
   average_hinge_loss = sum(hinge_losses) / total_docs
   reg_term = λ * sum(w_j ** 2 for w_j in w)
   total_loss = reg_term + average_hinge_loss
   return total_loss #Final total loss
```

Instance 3.

I asked ChatGPT how to define documents given the function:

```
def loss_SVM(w, bias, X, y, λ):
```

ChatGPT told me to use the following line of code:

```
documents = list(zip(X, y))
```

I asked ChatGPT why, and what the list(zip()) function did, and it gave me the following output:

Great question! The Python function `list(zip(...))` is commonly used to **combine multiple iterables (like lists or arrays) element-wise into tuples**.

Instance 4.

With the prompt: You are given the following dataset data_for_svm.csv, where the first 64 columns contain X and the last column contains y, I tried to write the following code:

```
X = data_svm.iloc[:, :-1].values
```

I received the error `AttributeError: 'DataFrame' object has no attribute 'iloc'`

I inputted this error into ChatGPT which told me:

Fix: Convert to pandas DataFrame If you want to use .iloc and .values, first convert the PySpark DataFrame to a pandas DataFrame using .toPandas(). And therefore gave me this new code:

```
# Convert Spark DataFrame to pandas DataFrame
data_pd = data_svm.toPandas()
# Split into features and labels
X = data_pd.iloc[:, :-1].values
y = data_pd.iloc[:, -1].values
```

Instance 5.

I asked ChatGPT what a normal value for lambda would be for the SVM function to understand what I should be inputting into the code. It gave me the following output:

If **λ is small** (e.g., `0.0001`), you're allowing more complex models — could overfit.

If **λ is large** (e.g., `10`), you're forcing simplicity — could underfit.

Therefore I defined the following value:

```
λ = 0.01
```

Instance 6.

I was receiving an error message with the following line of code:

```
rdd = data_svm.rdd.map(lambda row: (row[:-1], row[-1]))
```

I inputted this line of code into ChatGPT and I received the suggestion to change it to the following line of code:

```
rdd = data_svm.rdd.zipWithIndex().map(lambda row: (row[1], row[0]))
```

I was confused about what the .zipWithIndex() function did, so I asked ChatGPT to explain it. It gave me the following explanation:

🔹 `.zipWithIndex()`

This is a Spark **RDD transformation** that:

> Adds a unique, **0-based index** to each row in the RDD.

It returns a new RDD where each element is a tuple

This is useful because it transforms the RDD into (index, row) pairs which is helpful for our predictions in the map_predict function.