Stage 4- Data Merging and Analysis

Sukanya Venkataraman (venkatarama5@wisc.edu)
Yogesh Chockalingam (ychockalinga@wisc.edu)
Shantanu Singhal (singhal5@wisc.edu)

Show how did you combine the two tables A and B to obtain E? Did you add any other table? When you did the combination, did you run into any issues? Discuss the combination process in detail, e.g., when you merge tuples, what are the merging functions?

We performed the merging process differently for each set of attributes in the table. We also wrote 2 helper functions to help with the merging process and describe them first.

- `getOlderDate(time1, time2)`
  - This function returns the older of the two datetime parameters.
- `noneHandler(cell1, cell2)`
  - We have many values set to None in the tables and to choose between two cells when one or both are None, this function picks the non-None cell and if both are None, then returns None.

We now describe the merging process for each attribute.

- For 'Name' and 'Author', we pick the larger of the two strings. Empty cells are handled by the *noneHandler* function.
- For 'Publishing_Date', 'Publisher', and 'Format', we first determine the older of the two publishing dates using the *getOlderDate* function. The merged value of 'Publisher' and 'Format' were then set to the value of the 'Publisher' and 'Format' in the tuple corresponding to the older date.
  - In case one of the two dates was None, we used the other date, and set the 'Publisher' and 'Format' corresponding to the non-None date.
  - In case both the date values were None, we then set the 'Publishing_Date' to None and pick the longer of the 'Publisher' and 'Format'. Null values in 'Publisher' and 'Format' are handled using the *noneHandler* function.
- For 'Pages' and 'Rating', we compute the average of the values present in the two cells and use the average in the merged tuple.
  - For 'Pages', we then round the average to the nearest integer as a book cannot have fractional number of pages. E.g. 127.5 pages was rounded to 128 pages. None values were handled using the *noneHandler* function.
  - For 'Rating', we round up the average to have a precision of 1 after the decimal. None values are handled using the *noneHandler* function.

The code for the merging process can be found at the end of this document.

Issues

- We ran into a few issues due to missing cells present in the set of matching tuples. We decided to use the older pf the publishing dates to determine the value of 'Format' and 'Publisher'. Due to this dependency, we had to handle multiple edge cases where the publishing date was null and how this affected the values of the publisher and format.

- We also had some issues in determining the older of the two dates because certain date values were not formatted properly. Hence, we had to fix the format before we determine which date is older. We picked the longer of the two strings as it increases the descriptiveness of the information contained in our table.

We did not add any other table during the merging stage. Once the merging was complete and we had a single table E, we added a new column called 'Gender', which describes the gender of the author. This was done to help with the analysis phase. More details can be found in the explanation of the analysis task below.

Statistics on Table E: specifically, what is the schema of Table E, how many tuples are in Table E? Give at least four sample tuples from Table E.

Schema

S No.	Attribute	Type	Description
1	Name	String	Name of the book
2	Author	String	Name of the author(s) of the book
3	Publisher	String	Name of the book publisher
4	Publishing_Date	String	Date published. Format - year-month-day
5	Format	String	Format of the book - Paperback, Kindle etc.
6	Pages	Int	Number of pages
7	Rating	Float	Book rating, out of 5

Number of tuples

Table E has 6309 tuples.

Sample tuples

Name	Author	Publisher	Publishing_Date	Format	Pages	Rating
Age of Myth: Book One of The Legends of the First Empire	Michael J. Sullivan	Del Rey	2017-1-31	Paperback	464	4.5
Rise of the Dragons (Kings and Sorcerers--Book 1)	Morgan Rice	Morgan Rice	2017-8-4	Hardcover	217	4.1
The Book of Deacon (Volume 1)	Joseph Lallo	CreateSpace Independent Publishing Platform	2012-3-18	Kindle	322	4.3

A Quest of Heroes: Book #1 in the Sorcerer's Ring	Morgan Rice	Morgan Rice	2012-12-3	Hardcover	234	3.6

What was the data analysis task that you wanted to do? For that task, describe in detail the data analysis process that you went through.

Pornography classification

- **Description:** We scraped books from the fantasy genre from Amazon and Goodreads in stage 2 and hence quite a few pornographic books also got scraped and added to the dataset. One of the analysis we performed was to build a deep learning classifier to detect if a book is pornographic or not.
- **Classifier:** We used the title of the book as the sole feature, as the other attributes were not relevant to this analysis. We used an embedding layer as the first layer in the neural net and a built a simple network with 2 hidden layers to classify the books. To obtain the training data, we searched the dataset for titles containing the words (sex, erotic, gay, lesbian, porn, adult, hot, intimate, romance, romantic, nsfw). We obtained 289 books which matched this query. The tuples corresponding to these books were labelled as positive and the remaining tuples were labelled as negative and we trained the classifier.
- **Result:** Unfortunately, because of the major class imbalance (289 positive vs 6030 negative), the classifier predicted every book as not pornographic and we weren't able to obtain any insight from this analysis. The precision and recall are reported below.

Give any accuracy numbers that you have obtained (such as precision and recall for your classification scheme). Report which learning based matcher you selected after that cross validation.

Pornography Classification

	Precision	**Recall**	**F1-score**	**Support**
0	1.00	1.00	1.00	6030
1	0.00	0.00	0.00	289
Total	1.00	1.00	1.00	6030

As seen in the table above, the classifier classified everything as not porn.

What did you learn/conclude from your data analysis? Were there any problems with the analysis process and with the data?

Pornography Classification

- Using just the title is not descriptive enough to build a model and classify the books. We need additional features.
- The problem of class imbalance also affected the model severely.

If you have more time, what would you propose you can do next?

Pornography Classification

- We would add new features or train the classifier on an external corpus and then use it to predict if each book is pornographic or not.
- The class imbalance problem can be corrected at least partially using class weighing, over sampling the underrepresented class, under sampling the overrepresented class or by adding more training data.

Append the code of the Python script (that merges the tables) to the end of this pdf file.

```
def getOlderDate(time1, time2):
    """
    Helper function to get the older of the 2 dates, time1 and time2.
    """
    # We have imputed missing day, month or year with a 0; Replacing the 0 with 1 here
    time1 = '-'.join(index if index != '0' else '1' for index in time1.split('-'))
    time2 = '-'.join(index if index != '0' else '1' for index in time2.split('-'))

    minValue =  min(datetime.strptime(time1, '%Y-%M-%d'), datetime.strptime(time2, '%Y-%M-%d'))
    return minValue.strftime('%Y-%M-%d')



def noneHandler(cell1, cell2):
            """
            Handles nan present in the cells
            """
    if cell1 == 'nan' and cell2 == 'nan':
        return None

    return cell1 if cell1 != 'nan' else cell2



def merge(matches):
    """
    Takes in the matches dataframe, extracts the matching
```

```
rows one by one from each table and then merges them.
"""
output = []

# Get the indices of the matching rows from each table as a tuple: e.g. (31, 26)
indices = list(zip(matches['ltable_ID'].tolist(), matches['rtable_ID'].tolist()))

for index1, index2 in indices:
    # Holds the merged row
    merged = []

    # take max length
    for column in ['Name','Author']:
        cell1 = str(A.at[index1, column])
        cell2 = str(B.at[index2, column])

        if cell1 != 'nan' and cell2 != 'nan':
            merged.append(max(cell1, cell2))
        else:
            merged.append(noneHandler(cell1, cell2))

    # take the oldest publishing date, publisher and format
    for column in ['Publishing_Date']:
        cell1 = str(A.at[index1, column])
        cell2 = str(B.at[index2, column])

        if cell1 != 'nan' and cell2 != 'nan':
            olderDate = getOlderDate(cell1, cell2)
            if olderDate == cell1:
                merged += [A.at[index1, 'Publisher'], olderDate, A.at[index1, 'Format']]
            else:
```

```
            merged += [B.at[index2, 'Publisher'], olderDate, B.at[index2, 'Format']]
    elif cell1 == 'nan' and cell2 == 'nan':
        # Publishing date is none for both
        pubCell1 = A.at[index1, 'Publisher']
        pubCell2 = B.at[index2, 'Publisher']
        formatCell1 = A.at[index1, 'Format']
        formatCell2 = B.at[index2, 'Format']

        # Take max of the publisher and format
        if pubCell1 != 'nan' and pubCell2 != 'nan' and formatCell1 != 'nan' and formatCell2 != 'nan':
            merged += [max(pubCell1, pubCell2), None, max(formatCell1, formatCell2)]
        else:
            merged += [noneHandler(pubCell1, pubCell2), None, noneHandler(formatCell1, formatCell2)]
    elif cell2 == 'nan':
        merged += [A.at[index1, 'Publisher'], A.at[index1, 'Publishing_Date'], A.at[index1, 'Format']]
    else:
        merged += [B.at[index2, 'Publisher'], B.at[index2, 'Publishing_Date'], B.at[index2, 'Format']]

for column in ['Pages','Rating']:
    cell1 = str(A.at[index1, column])
    cell2 = str(B.at[index2, column])

    if cell1 != 'nan' and cell2 != 'nan':
        avgValue = np.mean([float(cell1), float(cell2)])
        # Round the number of pages to nearest int, as we cannot have 127.5 pages
        if column == 'Pages':
            merged.append(str(round(avgValue)))
        else:
            merged.append(str(round(avgValue, 2)))
    else:
        merged.append(noneHandler(cell1, cell2))
        output.append(merged)

    return pd.DataFrame(output, columns=['Name', 'Author', 'Publisher', 'Publishing_Date', 'Format','Pages', 'Rating'])

merged = merge(matches)
```