**Assignment 2: Building a network from email communication**

In this assignment, you are writing a code based on classes about (1) Pandas, (2) Regular Expression, and (3) Email data analysis. The goal of this assignment is to help students integrate the concepts and real-world data that they did not encounter in the class. Students will be given datasets that contain email threads and the columns are (1) thread id: unique ID for each thread, (2) thread name: the first subject of the email, (3) body: the content of the email, (4) account: the email account of the sender, (4) url: the URL of the email, and (5) date: the date of the email was sent.

**1. Regular Expression (RegEx) (2pt)**

1-1. RegEx to extract usernames (1pt)

In the ‘account’ column, the value is stored in a string even though it looks like a list object. For instance, if you do data[‘account’].iloc[0] then the output will be like “[‘chojiro1990’ ‘soronel.haetir’ …]” (See Figure 1)

**A computer screen shot of a computer code

Description automatically generated**

<Figure 1>

This means we have to use Regular Expression to match the username. Please note that there are usernames that have a period mark (.) in the middle and digits in the username. Define your pattern by using re.compile() and apply str.findall() on the ‘account’ column. Save your result in ‘account\_list’ column. Once you save it, the output should look like <Figure 2>.

A computer screen shot of a computer code

Description automatically generated

<Figure 2>

1-2. RegEx to extract date (1pt)

In the ‘date’ column, the value is stored in a string even though it looks like a list object. For instance, if you do data[‘date’].iloc[0] then the output will be like “[‘2013-06-05T09:50:53.000000000’ ‘2013-06-05T15:42:35.000000000’\n …]” (See Figure 3)

**A black screen with white numbers

Description automatically generated**

<Figure 3>

This means we have to use Regular Expression to match the date-time format. The date has consistent format. It starts with the year, which is four digits and there is a hyphen before it expands to month. Define your pattern by using re.compile() and apply str.findall() on the ‘account’ column. Save your result in ‘date\_list’ column. Once you save it, the output should look like <Figure 4>.

**A screen shot of a computer

Description automatically generated**

<Figure 4>

**2. Pandas and Text mining (6pt)**

2-1. Count the number of users involved in the email conversation (1pt)

Now that we have the list of usernames in ‘account\_list’ column, we can use .apply(lambda x: len(x)) to count the number of elements. The email thread that has the most number of users has 38 users involved.

Get the thread\_id for the thread that has the most number of users (38 users) involved in the email conversation.

2-2. Lowercase the body column (1pt)

Lowercase the string in the ‘body’ column and save the output in ‘body\_lower’ column. The output should look like <Figure 5>.

A screenshot of a computer screen

Description automatically generated

<Figure 5>

2-3. Removing stopwords (1pt)

Apply lambda x: ‘ ’.join([word for word in word\_tokenize(x) if word not in stop]) on ‘body\_lower’ column and save the output in ‘stopword’ column. The output should look like <Figure 6>.

A screenshot of a computer screen

Description automatically generated

<Figure 6>

2-4. Tokenization and removing punctuations (1pt)

Use lambda x: word\_tokenize(x) on ‘stopword’ column and save the output in ‘token’ column.

After that, use ‘token’ column to use lambda x: [word for word in x if word.isalnum()] to find tokens that are not punctuations and save tokens if they are not punctuations in ‘punct\_token’ column.

The output should look like <Figure 7>.

A screenshot of a computer program

Description automatically generated

<Figure 7>

2-5. Lemmatization (1pt)

Use ‘punct\_token’ column to lemmatize the word. Use a lemmatizer from WordNet. But remember NLTK’s WordNetLemmatizer only lemmatize nouns. Therefore, we need to specify the part of speech (POS) for each token. Use nltk\_tag\_to\_wordnet\_tag and lemmatize\_sentence function in text-analysis.ipynb file to lemmatize the value in ‘punct\_token’ column.

Once lemmatization is finished, save the output in the ‘lemma’ column. The output should look like <Figure 8>.

A screenshot of a computer screen

Description automatically generated

<Figure 8>

2-6. Count the frequency of unigrams (1pt)

What is the most frequent unigram in the ‘lemma’ column? List up only the 10 most frequent unigrams and their counts.

3. Building edges (2pt)

3-1. Combination (1pt)

We will build edges among all users involved in the thread. We will use itertools library and use itertools.combinations() to group two users into tuples. Let’s save edges in the variable called ‘edges’. Remember that we have saved the usernames in the list in ‘account\_list’ column.

Compute how many edges can be built by simply applying itertools.combinations()

3-2. Removing self-loop (1pt)

Finally, let’s remove self-loops. Remember self-loops are nodes connected to themselves.

Compute how many edges there are after removing self-loops.