Lab Session – Text Mining: Regular Expression & Topic Models

Please complete the tasks below. Provide your Python code and your answers to the questions as two separate documents.

Task 1: Regular Expressions

Data: Please download the data set titled "fraudulent_emails.txt" to work on this task. The data is a large repository of scam e-mails that have circulated online. It includes e-mail meta-data (e.g., From / To addresses, names, timestamp, etc.) as well as the entire e-mail message. Notice that the file is an unstructured format and e-mails had been continuously added to the end of the document.

1.1. Your task is to extract the sender's name, e-mail address, and the e-mail address domain from the fraudulent e-mails in the document. This information can then be used a simple e-mail filter. Notice that all of the required information is stored in a single line in the meta-data – i.e., the "From:" line.

```
From r  Mon Nov 25 22:32:21 2002
Return-Path: <software-seminar-request@eecs.UM>
X-Sieve: cmu-sieve 2.0
Return-Path: <software-seminar-request@eecs.UM>
    Mon, 25 Nov 2002 22:20:56 -0500
Resent-Date: Mon, 25 Nov 2002 22:20:56 -0500
X-Authentication-Warning: smtp.eecs.UM: list set sender to software-seminar-request@smtp.eecs.UM using -f
Message-Id: <200211260320.gAQ3KoVV001933@smtp.eecs.UM>
From: "DR. JAMES  ALBERT" <jamesalbert0@eircom.net>
Reply-To: jamesalbert_5@email.com
To: software-seminar@eecs.UM
Date: Tue, 26 Nov 2002 04:34:05 +0100
```

Write a regular expression based program that reads the data set, extracts the required information, and stores in a Pandas DataFrame. Only match and extract the sender information if both of the following arbitrarily chosen conditions hold:

1) Only match the senders who argue that they hold a doctorate title. In other terms, extract the records whose names start with any combination of "Dr." (e.g., "dr", "dr.", "Dr.", "DR", etc.) and
2) Only match the senders, if their e-mail addresses end with ".com" or ".net".

The final outcome should look like the image below.

Index	Name	E-mail Address	E-mail Domain
0	Dr.Sam jordan	sjordan@diplomats.com	diplomats.com
1	DR. JAMES ALBERT	jamesalbert0@eircom.net	eircom.net
2	DR. SANNI SABO	davidmensoh@mail.com	mail.com
3	DR. SANNI SABO	davidmensoh@mail.com	mail.com
4	DR. SANNI SABO	davidmensoh@mail.com	mail.com
5	DR. BELLO MUSA	dr_bellomusa01@omaninfo.com	omaninfo.com
6	Dr mariam abacha	marabac011@hotmail.com	hotmail.com
7	DR DOUGLAS NWANKWO	douglankwo@yahoo.com	yahoo.com
8	Dr.Moses	moses1970@englishexpert.com	englishexpert.com
9	Dr.Moses	moses1970@englishexpert.com	englishexpert.com

Hint:

1) You can extract a partial section of a matched string using grouping (i.e., parentheses) in the pattern definition. Refer to lecture notes – Groups, to see examples.

QUESTION 1: How many records have been matched? Please provide a screenshot of the first 21 records in the dataframe.

OPTIONAL QUESTION: For the e-mails that were identified and extracted in the 1st question, extract the corresponding sent timestamp (e.g., "Fri, 10 Feb 2006 16:04:48") from the "Date:" line and add to the previous dataframe.

Task 2: Topic Modeling with LDA

Data: Please download the data set titled "headlines.csv" to work on this task. The data includes a number of headlines collected from news websites. Each row is a unique headline (i.e., document).

2.1. Import and convert the data into a "document-term" matrix, where each row corresponds to a single document and columns are the term frequency counts. You can use scikit-learn's CountVectorizer() functionality for this purpose. For pre-processing, the required steps are:

1) Remove the stop words
2) Convert all characters to lowercase
3) Use "min_df=5", and "max_df=0.8" options within the CountVectorizer() method to exclude both rare and very common terms from the analysis.

QUESTION 2: How many terms (i.e., tokens) exist in the document-term matrix? Provide a screenshot of the first 5 rows and first 10 columns. Note that after fit() and transform() methods with scikit-learn, the output will be a sparse matrix. To convert it to a dataframe with column names as terms, you can use the following code snippet:

```
countVectorizer = CountVectorizer(min_df=5,max_df = 0.8,analyzer=stemmedStoppedStripped)
terms = countVectorizer.fit(data)
X = terms.transform(data)
termList = terms.get_feature_names()
dfX = pd.DataFrame(X.toarray(),columns=termList)
```

2.2. Assume that we'd like 10 topics to be generated from the document-term matrix. Use the LDA model (from sklearn.decomposition import LatentDirichletAllocation) to create a new LDA based matrix (called ldaMat). This matrix should have the dimensions of 299 x 10.

QUESTION 3: Print out the top 15 words in each topic using the code snippet below. Provide a screenshot of the output.

```
topWordCount = 15
for idx, topic in enumerate(lda_model.components_):
    sortedIndices = topic.argsort()[::-1]
    print('Topic {}:'.format(idx+1),[(termList[i]) for i in sortedIndices[:topWordCount]])
```

OPTIONAL QUESTION: Use the ldaMat as an input to the k-means clustering algorithm to cluster the documents into 3 groups.