Practical Natural Language Processing
Instructor: Rasika Bhalerao
Assignment 2
Due September 21 (at the beginning of class at 2pm)

Please submit your solutions using NYU Classes.

Please include your code, with brief explanations (such as "We compute the topic vector using the following code").
The easiest way to submit this assignment will be a pdf printout of a Jupyter notebook (https://jupyter.org/). You can also write a document with code snippets in it or print out a Python file with comments for the written parts.

"By hand" means you should use Python, but not a pre-built API that does the NLP algorithms we learned for you.

For this assignment, we will use this dataset of real and fake news:
https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset
Download Fake.csv and True.csv from this webpage (note the "Skip & continue download" option if you don't want to create a Kaggle account).

We will use Pandas for data manipulation. Pandas is a popular and powerful data analysis tool. Find details here: https://pandas.pydata.org/

Read the dataset into a Pandas dataframe like this:

```
import pandas as pd
df_real = pd.read_csv('True.csv')
df_real['RealNews?'] = True
df_fake = pd.read_csv('Fake.csv')
df_fake['RealNews?'] = False
df = df_real.append(df_fake)
```

`df` should now be a dataframe with all the columns that the two CSVs came with (title, text, subject, and date), and an added column for whether the news is real or fake.
You can verify this by looking at the first five rows of the dataframe like this:

```
df.head()
```

You can verify that you have all 44,898 rows of the dataframe by checking its length:

```
len(df)
```

You can select just the rows of real news like this:

```
df[df['RealNews?']]
```

You can select just the rows of fake news like this:

```
df[~df['RealNews?']]
```

You can select just the “title” column of the dataframe like this:

```
df[‘title’]
```

You can create a new column for the entire document (title + text) like this:

```
df['document'] = df[['title', 'text']].agg(' '.join, axis=1)
```

In this assignment, we will classify news articles as real or fake news, using the words of the document (title + text) as features.

For simplicity, let’s ignore case for this assignment:

```
df['document'] = df['document'].apply(lambda x: x.lower())
```

We will also use Scikit learn: https://scikit-learn.org/stable/

Split the dataframe into a training set and test set like this:

```
from sklearn.model_selection import train_test_split
df_train, df_test = train_test_split(df, test_size=0.2, shuffle=True)
```

For this assignment, you can also use Scikit-learn to calculate precision, recall, and F1 score:

```
from sklearn.metrics import precision_recall_fscore_support
precision_recall_fscore_support(y_true, y_pred)
```

Part 1: Naive Bayes “by hand”

Train a Naive Bayes classifier on the training set. Use Laplace (add-1) smoothing. Report the precision, recall, and F1 score on the test set.

Hints:

- Use a regular expression such as `re.split(r"\W+", document)` to tokenize documents.
- You may use lists, dictionaries, sets, and any other built-in Python data structures.
- The key to knowing you are doing it "by hand" is that you should be writing code to calculate probabilities.

Part 2: Tf-idf "by hand"

For this part, use linear logistic regression from Scikit-learn:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

You can import it like this:

```
from sklearn.linear_model import LogisticRegression
```

Prepare a vector for each document in the training set using tf-idf. Do this by hand, and make sure to only take the training set into account when calculating training frequencies!

When calculating the tf-idf vectors, you may choose to include only words that appear at least twice in your training set. (You might find that the code takes too long to run without this limitation.)

Once you have a list `X_train` of training set tf-idf vectors, and a corresponding vector `y_train` of training set targets, train the logistic regression using:

```
clf = LogisticRegression(random_state=0).fit(X_train, y_train)
```

Create tf-idf vectors for the test set by hand. You will need to use frequencies from the training set when calculating inverse document frequencies for the test set (you will also need to use the same word index order for the vectors that you used in the training set).

Make predictions for the test set using:

```
clf.predict(X_test)
```

Report the precision, recall, and F1 score.

Part 3: Naive Bayes and Tf-idf using Scikit-learn

Now, do the classification using the relevant tools available in Scikit-learn!

First, do this for Naive Bayes: https://scikit-learn.org/stable/modules/naive_bayes.html

Then, use Tf-idf (https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html) with Linear Logistic Regression.

Make sure to know the difference between `fit`, `transform`, and `fit_transform`, so that the test set frequencies are not included when calculating the training set vectors!

Report the precision, recall, and F1 scores for both methods.

For the last point on this assignment, experiment with one of the parameters in the Tfidf vectorizer (character ngram analyzer, stop words, norm, etc.). Briefly explain what the parameter is, and show how it affected the scores when you changed it.