Programming Assignment 1:
Interactive Naïve Bayes Text Classification

In this assignment, you will develop an interactive naïve bayes classifier for text documents. This assignment is designed based on Kulesza et al. (IUI 2015). You are highly encouraged to read this paper first before starting the assignment.

You will use a customized version of the DBPedia dataset as the training and test data. Please download the DBPedia dataset from this dropbox link. This dataset is a csv file with two columns. The first column is the category ID, and the second column is the text document. There are a total of eight categories in this dataset. Each category includes 1K text documents originally from Wikipedia.

Category ID	Category Name
0	Company
1	Education Institution
2	Artist
3	Athlete
4	Office Holder
5	Mean of Transportation
6	Building
7	Natural Place

Step 1. Implementing a Multinomial Naïve Bayes (MNB) Classifier

As the first step, you need to implement a MNB classifier for the DBPedia dataset. If you haven't heard about multinomial naïve bayes before, please read this tutorial. This MNB classifier will use words as features and learn the posterior probability based on word frequency in documents from different categories. Given a text document, the MNB classifier will predict the probability of the document belonging to each category and pick the category with the highest probability as the final prediction.

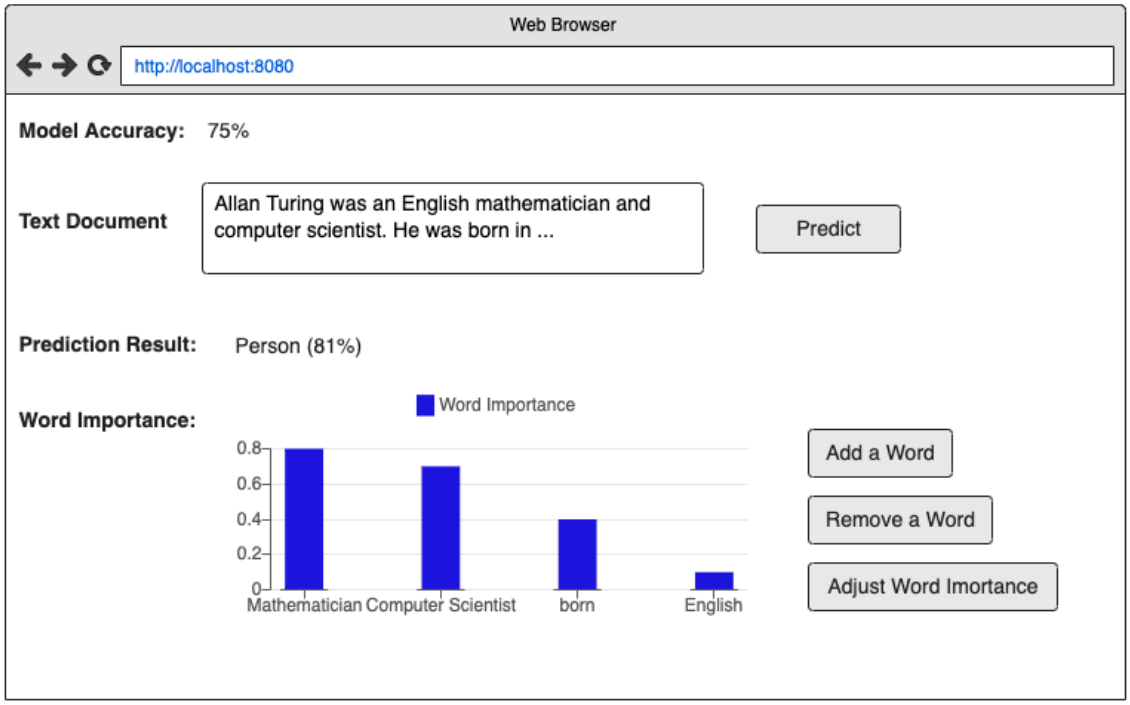
DO NOT use a naïve bayes classifier from ML libraries like sklearn, since you will need to modify the source code of the classifier to make it interactive. But feel free to search online and use some open-source implementation of a MNB classifier.

Once you have implemented the MNB classifier, you can now train it with 80% of the DBPedia dataset and test it with the rest 20% of the dataset. Don't forget to do 10-fold validation.

Step 2. Implementing a User Interface for the MNB Classifier

In the second step, you will implement a simple user interface for your MNB classifier. The user interface should render the overall accuracy of the MNB classifier on the test dataset. It should also allow users to type in a document for classification and show the predicted category with its probability. Furthermore, the user interface should generate a visual explanation of the prediction by plotting the top 10 words that contributes the most to the prediction based on their word importance in a column chart. The word importance is essentially $P(w|c)$, where w is the word and c is the class.

Please use the UI sketch below as a reference for your UI design.[1] You should feel free to improve it and make it prettier. But it is completely fine if you stick with the UI sketch below.



You can use any web development framework you are familiar with to implement your UI. If you don't have no web development experience before, I recommend using Python Flask to implement the back-end server and then implement the frond-end webpage using HTML/CSS/JavaScript. Here is a tutorial on creating a webpage using Flask. You can also use React to implement the front end.

To plot the column chart, you can use visualization libraries such as Chart.JS and D3. Pick a library you feel comfortable with.

Step 3. Making the MNB Classifier Interactive

[1] You don't have to implement the Add a Word, Remove a Word, and Adjust Word Importance buttons in Step 2.

Now it's time to make your model interactive! In this step, you need to implement three interactions---add a word as a new classification feature, remove a word, and adjust the importance of a word. A simple way to implement these three interactions is to add three buttons, as shown in the UI sketch above. A user can first select a word feature in the column chart and then click one of the three buttons to customize the model. Specifically, when a user clicks on Add a Word or Adjust Word Importance, a pop-up window will show up to let users type in the new word or expected word importance. Once the user is done with the customization, the MNB classifier should be trained and tested again. The model accuracy number should be updated in the UI accordingly.

Instead of implementing the interactions as three explicit buttons, there are two alternative design options. You will receive bonus points if you implement one of them.

- **(10 bonus points)** You can implement a context menu with three options---add a new word, remove a word, and adjust word importance. When a user right clicks on a word in the column chart, the context menu will be rendered.
- **(20 bonus points)** Instead of having a pop-up window to type in an expected word importance, you can make each column in the column chart adjustable---*users can drag the column to adjust its height*.

Adding and removing a word to the feature space are easy to implement. Adjusting word importance can be a little tricky. Recall the equation to calculate the word importance.

$$P(w|c) = \frac{\alpha_{wc} + F_{wc}}{\sum_{x=1}^{N} \alpha_{xc} + \sum_{x=1}^{N} F_{xc}}$$

In this equation, F_{wc} is the word count of w in the training documents that belong to the category c. α_{wc} is the smoothing parameter for word w in category c. It is often set to 1 by default. This smoothing parameter actually serves two purposes. First, it can prevent the equation from yielding 0 if no documents from class c contain the word w. Second, it can act as a Bayesian prior, adding the number of virtual occurrences of a word to its actual occurrences in the training data.

In this step, we will leverage α_{wc} to adjust the word importance. By allowing the user to set the value of α_{wc}, we are letting the user increase the number of virtual occurrences of word w in category c. The result is a classifier that considers word w to be stronger evidence in favor of category c than it had before the user's feedback.

Submission Instruction

First, please create a PDF document to describe how to set up and run your interactive classifier. Specifically, please specify what libraries and packages to install (and their versions) and the commands to run the application. Please also include the accuracy of your classifier on the test data. There is no page requirement on this PDF document. My suggestion is to think of it as a README file and keep it short.

Second, please make a demo video to demonstrate (1) how to use your UI to predict the category of a new document and (2) how to customize the model using the three interactions.

Finally, please put your source code, the PDF document, and the demo video into a zip file and upload it to BrightSpace.