Microsoft

Applied Machine Learning

Lab 3 – Working with Text Data

Overview

In this lab, you will use R or Python to work with text data. Specifically, you will use code to clean text, remove stop words, and apply Porter stemming to the remaining words. You will then create an Azure ML web service to classify tweets based on sentiment analysis.

What You'll Need

To complete this lab, you will need the following:

- An Azure ML account
- The files for this lab

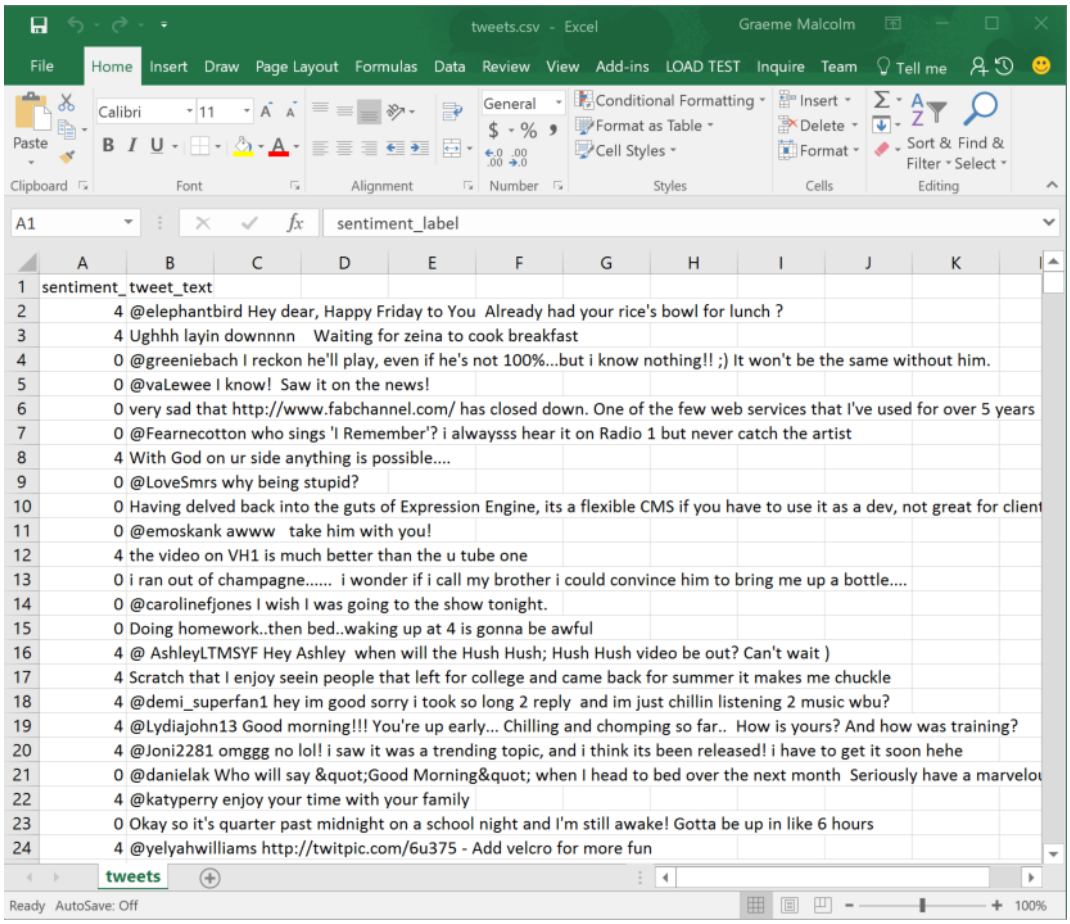
Note: To set up the required environment for the lab, follow the instructions in the Setup Guide for this course.

Exploring Text Data

The text data used in this lab consists of a collection of tweets that have been categorized as positive or negative.

Explore the Tweets Dataset

1. In the folder where you extracted the lab files for this module (for example, C:\DAT203.3x\Lab03), open the **tweets.csv** file, using either a spreadsheet application such as Microsoft Excel, or a text editor such as Microsoft Windows Notepad.
2. View the contents of the **tweets.csv** file, noting that it contains tweets with a numeric indication of sentiment in which 4 indicates a positive tweet, and 0 indicates a negative tweet:



3. Close the data file without saving any changes.

Explore the Stopwords Dataset

1. Open the **stopwords.csv** file and review its contents. Note that this file contains a list of common words such as “a”, “the”, “it”, and so on, which are generally not helpful in determining the meaning or sentiment of a sentence or paragraph.
2. Close the file without saving any changes.

Upload the Datasets to Azure Machine Learning

1. Browse to https://studio.azureml.net and sign in using the Microsoft account associated with your free Azure ML account.
2. If the **Welcome** page is displayed, close it by clicking the **OK** icon (which looks like a checkmark). Then, if the **New** page (containing a collection of Microsoft samples) is displayed, close it by clicking the **Close** icon (which looks like an X).
3. At the bottom left, click **NEW**; and in the **NEW** dialog box, in the **DATASET** tab, click **FROM LOCAL FILE**. Then in the **Upload a new dataset** dialog box, browse to select the **tweets.csv** file from the folder where you extracted the lab files on your local computer. Enter the following details, and then click the ✓icon.
    - **This is a new version of an existing dataset**: Unselected
    - **Enter a name for the new dataset**: tweets.csv
    - **Select a type for the new dataset**: Generic CSV file with a header (.csv)
    - **Provide an optional description**: Tweets.
4. Wait for the upload of the dataset to complete, then click **OK** on the status bar at the bottom of the Azure ML Studio page.

5. Repeat the previous steps to upload the stopwords.csv file as a new dataset with the following properties:
    - **This is a new version of an existing dataset**: Unselected
    - **Enter a name for the new dataset**: stopwords.csv
    - **Select a type for the new dataset**: Generic CSV file with a header (.csv)
    - **Provide an optional description**: Stopwords.

Working with Text Data in Jupyter

Now you are ready to use R or Python code in a Jupyter notebook to work with the text data.

Upload a Jupyter Notebook

1. In Azure ML Studio, click **NEW**; and in the **NEW** dialog box, in the **NOTEBOOK** tab, click **Upload**. Then in the **Upload a new notebook** dialog box, browse to select the notebook file for your preferred language (R or Python) from the folder where you extracted the lab files on your local computer – the R version of the notebook is named **TextPrep_R.ipnyb**, and the Python version is named **TextPrep_Py.ipynb**. Enter the following details, and then click the ✓icon.
    - **Enter a name for the new notebook**: TextPrep_R or Text_Prep_Py
    - **Select a language for the new notebook**: R or Python 2
2. Wait for the upload of the notebook to complete, then click **OK** on the status bar at the bottom of the Azure ML Studio page.

Use Code to Work with the Time Series Data

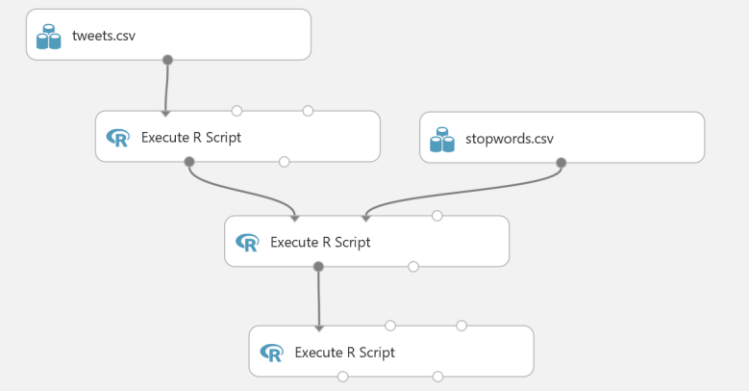
1. In Azure ML Studio, on the Notebooks tab, open the **TextPrep_R** or **Text_Prep_Py** notebook you uploaded in the previous procedure.
2. Follow the instructions in the notebook to work with the time series data.
3. When you have completed all of the coding tasks in the notebook, save your changes and then close and halt the notebook.

Sentiment Analysis in Azure ML

In the previous exercises, you used a Jupyter notebook to explore text data. Now you will use Azure ML to publish a classification model that uses similar code to prepare the text data, and then applies sentiment analysis to classify tweets as positive or negative.

Create an Azure ML Experiment

1. In your Web browser, open the gallery experiment at https://aka.ms/edx-dat203.3x-tweets, and then open it in Azure ML Studio, copying it to your workspace. The copied experiment should look like this:



2. Note that the experiment contains a number of **Execute R Script** modules to prepare the text data by removing stopwords and stemming the remaining words. Review the code in these modules.
3. Save and run the experiment, and visualize the **Results Dataset** (left) output of the final **Execute R Script** module to see the cleaned text.

Create Features for Classification

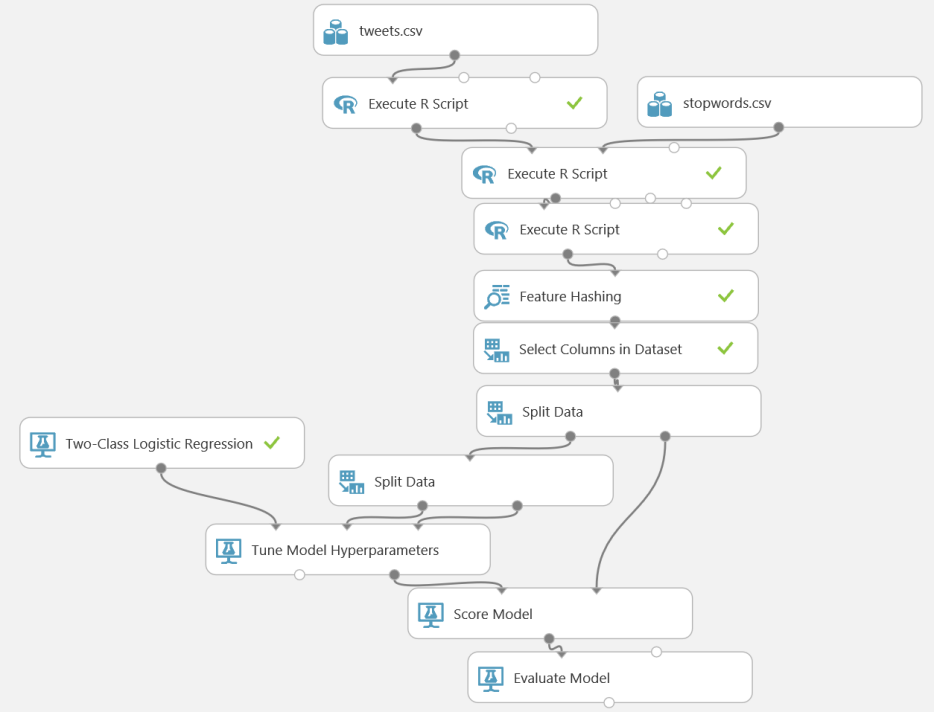
1. Add **Feature Hashing** module to the experiment and connect the **Results Dataset** (left) output of the final **Execute R Script** module to its input.
2. On the properties pane of the **Feature Hashing** module set the parameters as follows:
   - **Target column(s**): tweets
   - **Hashing bitsize:** 15
   - **N-grams:** 2
3. Save and Run the experiment.
4. Visualize the output of the **Feature Hashing** module, noting that there are now around 33,000 features. The hash has compressed the approximately 135,000 unique words into a smaller number of features.
5. Add a **Select Columns in Dataset** module to the experiment and connect the output of the **Feature Hashing** module to its input. Then configure the **Select Columns in Dataset** module to exclude the **tweets** column, which is not required for classification now that you have generated features (note that the column selector might take some time to open due to the large number of columns generated by feature hashing).
6. Save and run the experiment. Then visualize the output of the **Select Columns in Dataset** module and verify that the tweets column is no longer included.

Train and Evaluate a Classifier Model

You have created a feature set for classifying the sentiment of the tweets. Perform the following steps to construct and evaluate a classification model for tweet sentiment:

1. Add a **Split Data** module to the experiment and connect the output of the **Select Columns in Dataset** module to its input.
2. On the properties pane of the **Split Data** module set the following parameters:
   - **Splitting mode**: Split Rows
   - **Fraction of the rows in the first output dataset**: 0.8
   - **Randomized split**: Checked
   - **Random seed**: 1234
   - **Stratified split**: false
3. Add a second **Split Data** module and connect the **Results Dataset1** (left) output of the first **Split Data** module to its input.
4. Set the properties of the second **Split Data** module as follows:
   - **Splitting mode**: Split Rows
   - **Fraction of the rows in the first output dataset**: 0.7
   - **Randomized split**: Checked
   - **Random seed**: 1234
   - **Stratified split**: false
5. Add a **Tune Model Hyperparameters** module and connect the **Results Dataset1** (left) output of the second **Split Data** module to its **Training dataset** (middle) input. Then connect the **Results Dataset2** (right) output of the second **Split Data** module to its Optional validation dataset (right) input.

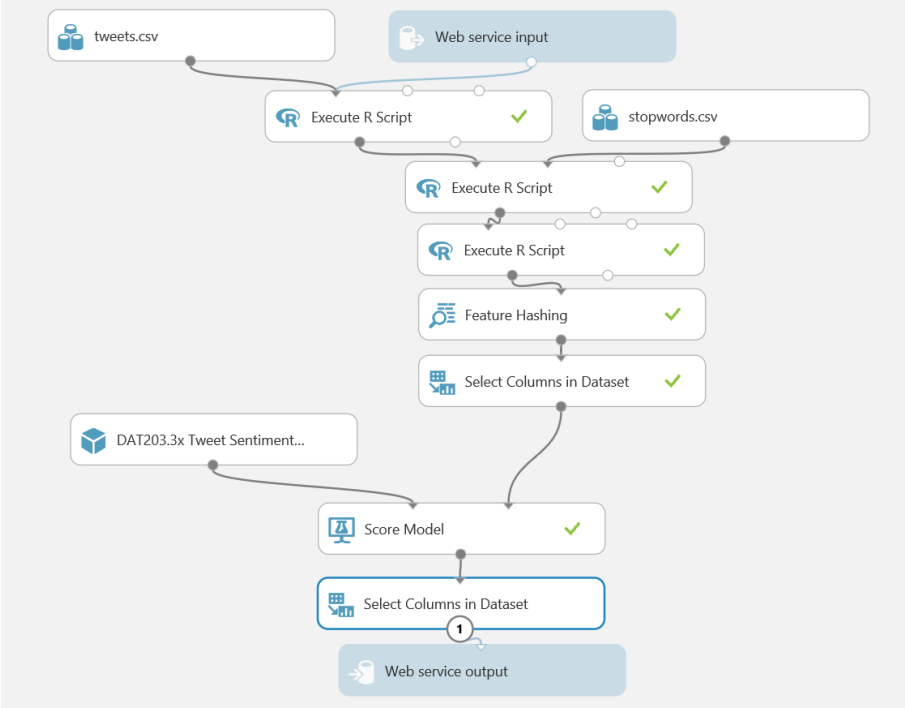
6. Add a **Two-Class Logistic Regression** module and connect its output to the **Untrained Module** (left) input of the **Tune Model Hyperparameters** module.
7. On the properties pane for the **Two-Class Logistic Regression** module set the following parameters:
   - **Create trainer mode**: Parameter Range
   - **Optimization tolerance**: **Use Range Builder**; Unchecked
   - **Optimization tolerance**: 0.0001, 0.0000001
   - **L1 regularization weight**: **Use Range Builder**; Unchecked
   - **L1 regularization weight**: 0.0, 0.01, 0.1, 1.0
   - **L2 regularization weight**: **Use Range Builder**; Unchecked
   - **L2 regularization weight**: 0.01, 0.1, 1.0
   - **Memory size for L-BFGS**: **Use Range Builder**; Unchecked
   - **Memory size for L-BFGS**: 2, 20, 50
   - **Random seed**: 1234
   - **Allow unknown levels in categorical features**: Checked
8. On the properties pane for the **Tune Model Hyperparameters** module set the following parameters:
   - **Specify parameter sweeping mode**: Random sweep
   - **Maximum number of runs on random sweep**: 50
   - **Random seed**: 1234
   - **Label column**: sentiment
   - **Metric for measuring performance for classification**: Accuracy
   - **Metric for measuring performance for regression**: Mean absolute error
9. Add a **Score Model** module and connect the **Results Dataset2** (right) output of the first **Split Data** module to its Dataset (right) input. Then connect the **Trained best model** (right) output of the **Tune Model Hyperparameters** module to its **Trained model** (left) input.
10. Add an **Evaluate Model** module and connect the output of the **Score Model** module to its **Scored dataset** (left) input.
11. Verify that your experiment resembles the following:



12. Save and run the experiment.
13. Visualize the output of the **Evaluate Model** module. Scroll down until you see the performance statistics and verify that the model is performing at least better than a random guess.

Create a Predictive Web Service

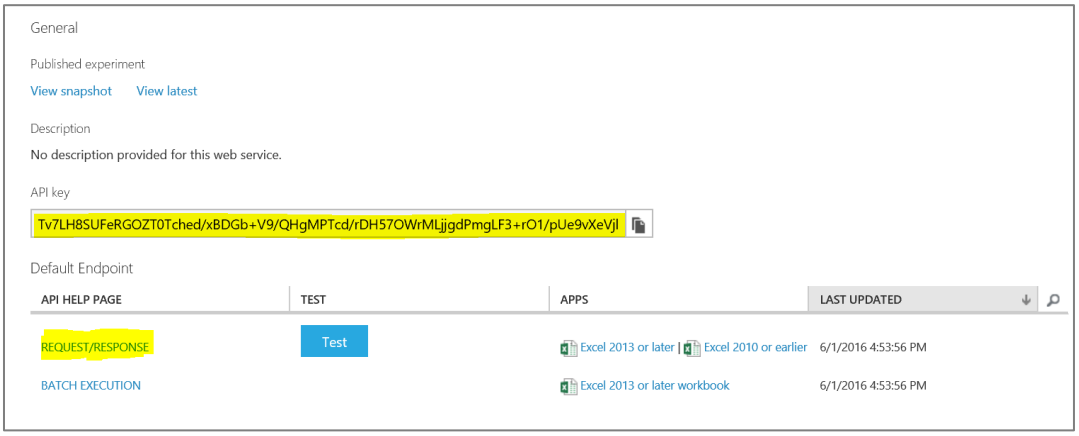
1. With the **DAT203.x:Tweet Sentiment** experiment still open, click **Set Up Web Service**, and then click **Predictive Web Service (Recommended)**. When a banner at the bottom of the screen notifies you that the experiment has been created, click **Close** to remove it.
2. Save and run the experiment to read the data and pass it through the workflow. Then visualize the output of the **Score Model** module and note that the web service returns all of the feature columns.
3. Add a **Select Columns in Dataset** module to the predictive experiment, and connect the output from the **Score Model** module to its input. Then connect its output to the **Web service output**.
4. In the properties for the **Select Columns in Dataset** module, use the column selector to select only the **Scored Labels** and **Scored Probability** columns.
5. Verify that your predictive experiment now looks like this:



6. Save and run the experiment, and visualize the output of the **Select Columns in Dataset** module to verify that only the **Scored Labels** and **Scored Probability** columns are returned by the web service. Positive tweets are indicated by a value of 1, and negative tweets are indicated by a value of -1.

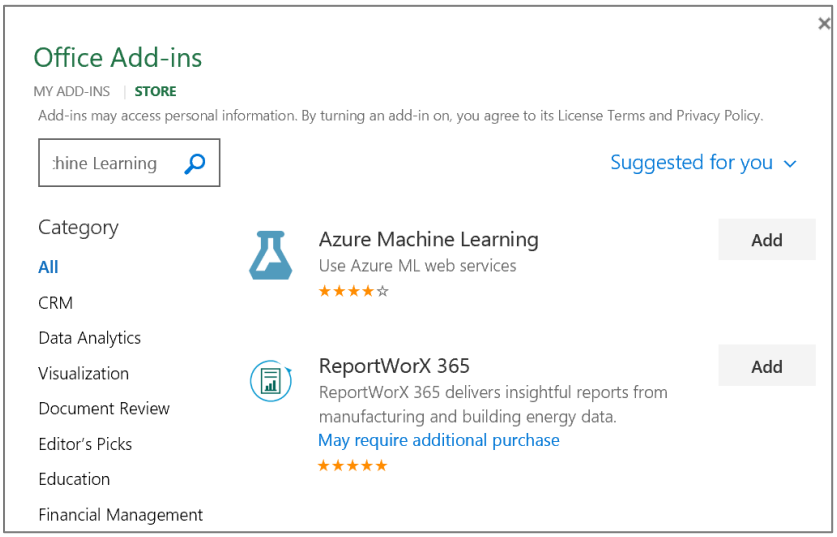
Deploy and Use the Web Service

1. In the **DAT203.x:Tweet Sentiment [Predictive Exp.]** experiment, click the **Deploy Web Service** icon at the bottom of the Azure ML Studio window.
2. Wait a few seconds for the dashboard page to appear, and note the **API key** and **Request/Response** link. You will use these to connect to the web service from a client application.



3. Leave the dashboard page open in your web browser, and open a new browser tab.
4. In the new browser tab, navigate to https://office.live.com/start/Excel.aspx. If prompted, sign in with your Microsoft account (use the same credentials you use to access Azure ML Studio.)
5. In Excel Online, create a new blank workbook.

6. On the **Insert** tab, click **Office Add-ins**. Then in the **Office Add-ins** dialog box, select **Store**, search for *Azure Machine Learning*, and add the **Azure Machine Learning** add-in as shown below:



7. After the add-in is installed, in the **Azure Machine Learning** pane on the right of the Excel workbook, click **Add Web Service**. Boxes for the URL and API key of the web service will appear.
8. On the browser tab containing the dashboard page for your Azure ML web service, right-click the **Request/Response** link you noted earlier and copy the web service URL to the clipboard. Then return to the browser tab containing the Excel Online workbook and paste the URL into the URL box.
9. On the browser tab containing the dashboard page for your Azure ML web service, click the **Copy** button for the **API key** you noted earlier to copy the key to the clipboard. Then return to the browser tab containing the Excel Online workbook and paste it into the **API key** box.
10. Verify that the **Azure Machine Learning** pane in your workbook now resembles this, and click **Add**:

11. After the web service has been added, in the **Azure Machine Learning** pane, click **1. View Schema** and note the *inputs* expected by the web service (**sentiment_label** and **tweet_text**) and the *outputs* returned by the web service (**Scored Labels** and **Scored Probability**).
12. In the Excel worksheet select cell A1. Then in the **Azure Machine Learning** pane, collapse the **1. View Schema** section and in the **2. Predict** section, click **Use sample data**. this enters some sample input values in the worksheet.
13. Modify the sample data as follows:

sentiment_label	tweet_text
	I love machine learning. It's great!
	This course is so much fun I love taking classes that are great it's my favorite thing to do
	I'm sad because the course will be over soon

14. Select the cells containing the input data (cells A1 to B4), and in the **Azure Machine Learning** pane, click the button to select the input range and confirm that it is **'Sheet1'!A1:B4**.
15. Ensure that the **My data has headers** box is checked.
16. In the **Output** box type **C1**, and ensure the **Include headers** box is checked.
17. Click the **Predict** button, and after a few seconds, view the predicted sentiment (**Scored labels**) and the associated confidence (**Scored Probability**) for each tweet.

Summary

In this lab, you have used R or Python in a Jupyter notebook to work with text data. You then created an Azure ML web service to classify tweets based on sentiment.