# [https://avatars2.githubusercontent.com/u/4156894?v=3&s=100](http://www.calstatela.edu/centers/hipi) CIS5560 Term Project Tutorial CIS5560 Term Project Tutorial

#### Authors:  Ruchi Singh; Kulwinder Kaur; Ovanes Mikaelian

#### Instructor: [Jongwook Woo](https://www.linkedin.com/in/jongwook-woo-7081a85)

#### Date: 05/18/2017

**Lab Tutorial**

Ruchi Singh ([rsingh26@@calstatela.edu](mailto:rsingh26@@calstatela.edu))

Kulwinder Kaur ([kkaur7@@calstatela.edu](mailto:kkaur7@@calstatela.edu))

Ovanes Mikaelian ([omikael@calstatela.edu](mailto:omikael@calstatela.edu))

**Text Analysis using AzureML algorithms**

**Objectives**

In this hands-on lab, you will learn how to:

* Text Classification to assign a text instance into one or more class(es) in a predefined set of classes.
* Provide users thoughts based on their tip.

**Requirements**

* An Azure ML account
* A web browser and Internet connection

**Platform Specifications**

* Microsoft Azure Machine Learning Studio
* Number of nodes: 1
* Total Memory Size: 10 GB
* Number of modules per experiment: 100

**Creating a Model:**

This tutorial shows how to create a Model in AzureML using text analysis modules. Applications of text classification include categorizing newspaper articles and news wire contents into topics, organizing web pages into hierarchical categories, filtering spam email, sentiment analysis, predicting user intent from search queries, routing support tickets, and analyzing customer feedback.

**Each instance in the data set has 6 fields:**

* business\_id – unique id for the business
* date – date of the tip
* likes – number of likes given to a tip range from 0 to 10.
* text- text of the users’s thoughts about business
* type- named as tip for the text provided
* user\_id - the user who posted the tip

Two columns text and likes are used in text analysis.

**Step 1: Data Preparation**

In this step the data is uploaded into AzureML, then it is transformed and cleaned so that it is suitable for the text analysis algorithms.

1. Open a browser and browse to [https://studio.azureml.net.](https://studio.azureml.net/) Then sign in using the Microsoft account associated with your Azure ML account.
2. Create a new blank experiment, and give it the title **Text analysis**.
3. Download the tip*.*csvfile and drag it to canvas.
4. Search for the **Select Columns in dataset(Project Column)**module and drag it onto the canvas.
5. Connect the output of the **tip** dataset to the **Dataset**input of **Select Columns in dataset(Project**

**Column).**

1. Select columns to be required by the experiment by clicking **Launch Column selector By name:**

* text
* likes
* Business\_id
* User\_id
* Date
* type

1. The records with missing text values are removed using **Clean Missing Values** module. Search for **Clean Missing data module** and drag it to the canvas. Connect the output of **Select Columns in dataset module** to the input of the **Clean missing data module**. **Select Launch Column Selector**. With Rules all Columns and select following:

**Minimum missing value ratio – 0**

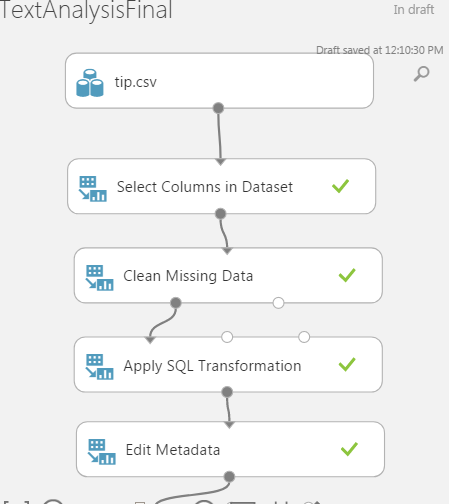
**Maximum missing value ratio – 1**

**Cleaning mode – Remove entire row**

1. Search for **Apply SQL Transformation module** and drag it to canvas. Connect the output of **Clean missing data** to the input of the **Apply SQL Transformation**. Copy and paste the SQL query below to select likes from 1 to 10 in default table t1 made for dataset:

select \* from t1 where likes IN ('1','2','3','4','5','6','7','8','9','10');

1. Search for **Edit Metadat(Metadata Editor) module** and drag it to the canvas. Connect the output of the **Apply SQL transformation module** to the input of the **Edit metadata module**. **Launch Column selector** By name and select likes columns only.
2. Run the experiment.



1. Search for **Apply SQL Transformation module** and drag it to the canvas. Copy and paste the Sql query below to delete unnecessary business\_id and user\_id in t1:

DELETE FROM t1

WHERE business\_id = '#NAME?';

DELETE FROM t1

WHERE user\_id = '#NAME?';

select \* from t1;

1. Search for **Select Columns in dataset(Project Columns)** module and drag it to the canvas. Connect the output of the **Apply SQL Transformation module** to the input of the **Select Columns in dataset(Project Columns)** module. Launch Column selector and with Rules No columns include text and likes columns.
2. Use the **Partition and Sample** module to select the top record in the dataset. Connect the output of the **Select Columns in dataset(Project Columns)** module to the input of the **Partition and Sample** module.
3. Search for Split Data module and drag it to the canvas. Connect the output of **Select Columns in dataset(Project Columns)** to the input of Split data module and apply following:

**Splitting mode : Split Rows**

**Fraction of rows in first output dataset : 0.2**

**Randomized Split : selected**

**Random Seed : 0**

**Stratified split : False**

**Step 2: Creating R scripts to Data Preprocessing:**

Unstructured text such as tips, tweets, product reviews, or search queries usually requires some preprocessing before it can be analyzed. This experiment includes a number of optional text preprocessing and text cleaning steps, such as replacing special characters and punctuation marks with spaces, normalizing case, removing duplicate characters, removing user-defined or built-in stop-words, and word stemming. By preprocessing the text, you can more easily create meaningful features from text. For example, the **Preprocess Text** module supports these common operations on text. These steps are implemented using the R programming language. We’ll use default dataset named Stopwords in the Azureml to compare the text and remove the stopwords in our dataset.

1. Search for **Execute R script module** and drag it to the canvas. Connect the first output of split data module to the first input od Execute R script, Copy and paste the R script below:

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

# Please determine the required text preprocessing steps using the following flag

replace\_special\_chars <- TRUE

remove\_duplicate\_chars <- TRUE

replace\_numbers <- TRUE

convert\_to\_lower\_case <- TRUE

remove\_default\_stopWords <- FALSE

remove\_given\_stopWords <- TRUE

stem\_words <- TRUE

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

# Map 1-based optional input ports to variables

dataset1 <- maml.mapInputPort(1) # class: data.frame

# get the label and text columns from the input data set

text\_column <- dataset1[["text"]]

label\_column <- dataset1[["likes"]]

stopword\_list <- NULL

result <- tryCatch({

dataset2 <- maml.mapInputPort(2) # class: data.frame

# get the stopword list from the second input data set

stopword\_list <- dataset2[[1]]

}, warning = function(war) {

# warning handler

print(paste("WARNING: ", war))

}, error = function(err) {

# error handler

print(paste("ERROR: ", err))

stopword\_list <- NULL

}, finally = {})

# Load the R script from the Zip port in ./src/

source("src/text.preprocessing.R");

text\_column <- preprocessText(text\_column,

replace\_special\_chars,

remove\_duplicate\_chars,

replace\_numbers,

convert\_to\_lower\_case,

remove\_default\_stopWords,

remove\_given\_stopWords,

stem\_words,

stopword\_list)

data.set <- data.frame(

label\_column,

text\_column,

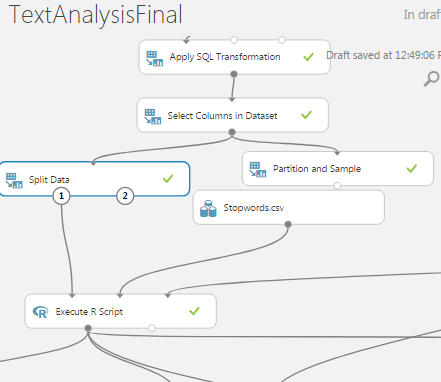
stringsAsFactors = FALSE

)

# Select data.frame to be sent to the output Dataset port

maml.mapOutputPort("data.set")

1. Load the dataset that contains the stopwords.
2. Attach the loaded dataset to the second port of the **Execute R Script** module.
3. Run the experiment.



1. Search for the text.oreprocessing.zip dataset that already exist in Azure and drag it to the canvas. Connect the output of the Dataset to the third output of the **Execute R script module.**
2. Search for the **Execute R script module** and drag it to the canvas. Connect the first output of first **Execute R script module** with first input of the new **Execute R script module**. Connect the output of the text.preprocessing.zip to the third input of new **Execute R script module**. We’ll make word cloud with this R script. Its taking label and text columns from the first input data set. The source is **text.preprocessing.R**. The script will draw word cloud with max words 50. Copy and paste the r script below :

# Map 1-based optional input ports to variables

dataset1 <- maml.mapInputPort(1) # class: data.frame

# get the label and text columns from the first input data set

text\_column <- dataset1[,"text\_column"]

label\_column <- dataset1[,"label\_column"]

# Load the R script from the Zip port in ./src/

source("src/text.preprocessing.R");

#freq = data.frame(sort(colSums(as.matrix(text\_column), decreasing=TRUE))

#wordcloud(rownames(freq), freq[,1], max.words=50, colors=brewer.pal(1, "Dark2"))

drawWordCloud(text\_column,label\_column, maxWords=50)

data.set <- dataset1

# Select data.frame to be sent to the output Dataset port

maml.mapOutputPort("data.set")

1. Go to the second output port of the last **Execute R Script** module named and select visualize if you need to see the most frequent words for each class. In the adopted use case, the first word cloud represents the top positive words and the second word cloud shows the most frequent negative words in the input training corpus.
2. Search for **Edit Metadata Module(Metadata Editor) module** and drag it to the canvas. Connect the first output of the first Execute R script to the input of the **Edit Metadata Module(Metadata Editor)** module. Launch Column Selector with Rules No cloumns select text\_column with following options:

**Data type: string**

**Categorical: Make non-categorial**

**Fields: unchanged**

**New column names: blank**

**Step 3: Creating Models:**

* **First Model :  N-grams TF feature extraction**

In the sample experiment, we set the number of hashing bits to 15, and set the number of n-grams to 2. With these settings, the hash table can hold 2^15 or 32,768 entries, in which each hashing feature represents one or more n-gram features and its value represents the occurrence frequency of that n-gram in the text instance. For many problems, a hash table of this size is more than adequate, but in some cases, more space might be needed to avoid collisions.

1. Search **Feature Hashing module** to transform a stream of English text into a set of features represented as integers. You can then pass this hashed feature set to a machine learning algorithm to train a text analysis model. The feature hashing functionality provided in this module is based on the Vowpal Wabbit framework. For more information, see [Train Vowpal Wabbit 7-4 Model](https://msdn.microsoft.com/en-us/library/azure/dn905861.aspx) or [Train Vowpal Wabbit 7-10 Model](https://msdn.microsoft.com/en-us/library/azure/mt674683.aspx). Connect the output of the **Edit Metadata Module(Metadata Editor**) module to theinput of the **Feature Hashing module.** Launch column selector and select text\_column with following options:

**Hashing bit size: 15**

**N- grams: 2**

The classification time and complexity of a trained model depends on the number of features (the dimensionality of the input space). For a linear model, such as a support vector machine, the complexity is linear with respect to the number of features. For text classification tasks, the number of features resulting from feature extraction is high because each word in the vocabulary and each n-gram is mapped to a feature. To select a more compact feature subset from the exhaustive list of extracted hashing features, we used the **Filter Based Feature Selection** module. The aim is to avoid the effects of the curse of dimensionality and to reduce the computational complexity without harming classification accuracy.

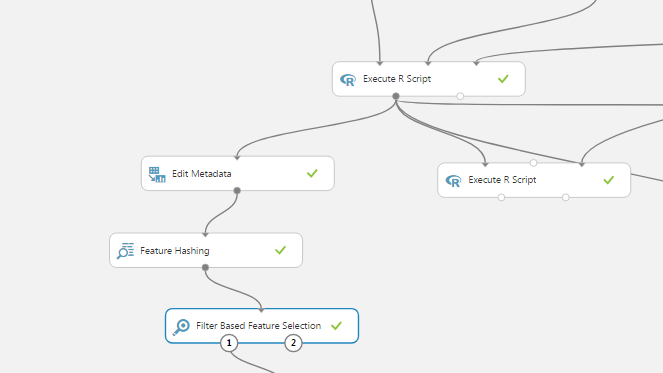
1. Search **Filter based Feature selection module** and drag it to the canvas. Connect the output of the **Feature Hashing module** to the input of **Filter based Feature selection module**. To get the top 1,000 most relevant features with respect to the sentiment label out of the 2^15 extracted features, we used the Chi-squared score function to rank the hashing features in descending order. Launch Column selector andselect label\_column. Select following:

**Feature Scoring Method : Chi Squared**

**Operate on Feature columns only : Selected**

**Number of desired features : 1000**

1. Run the experiment.



* **Second Model: Unigrams TF-IDF feature extraction**

**Create the Word Dictionary**

First, extract the set of unigrams (words) that will be used to train the text model. In addition to the unigrams, the number of documents where each word appears in the text corpus is counted (DF). It is not necessary to create the dictionary on the same labeled data used to train the text model. TF-IDF Calculation. When the metric word frequency of occurrence (TF) in a document is used as a feature value, a higher weight tends to be assigned to words that appear frequently in a corpus (such as stop-words). The inverse document frequency (IDF) is a better metric, because it assigns a lower weight to frequent words. You calculate IDF as the log of the ratio of the number of documents in the training corpus to the number of documents containing the given word. Combining these numbers in a metric (TF/IDF) places greater importance on words that are frequent in the document but rare in the corpus. This assumption is valid not only for unigrams but also for bigrams, trigrams, etc.

This experiment converts unstructured text data into equal-length numeric feature vectors where each feature represents the TF-IDF of a unigram in a text instance.

1.Search for **Execute R script module** and drag it to the canvas. Connect the output of the very first **Execute R script module** to the first input of the **Execute R script module**. Connect the output of the text.preprocessing.zip to the last input of the new **Execute R script module.**

Specify the following parameters in script for a word to be included in the dictionary created from the input dataset:

**a.** the minimum minWordLen and maximum maxWordLen length of a word.

**b.** the minimum minDF and the maximum maxDF document occurrence frequency.

Copy and paste the script below:

# Map 1-based optional input ports to variables

dataset <- maml.mapInputPort(1) # class: data.frame

##################################################

# Determine the following input parameters:-

# minimum length of a word to be included into the dictionary.

# Exclude any word if its length is less than \*minWordLen\* characters.

minWordLen <- 3

# maximum length of a word to be included into the dictionary.

# Exclude any word if its length is greater than \*maxWordLen\* characters.

maxWordLen <- 25

# minimum document frequency of a word to be included into the dictionary.

# Exclude any word if it appears in less than \*minDF\* documents.

minDF <- 9

# maximum document frequency of a word to be included into the dictionary.

# Exclude any word if it appears in greater than \*maxDF\* documents.

maxDF <- Inf

##################################################

# we assume that the text is the second column in the input data frame

text\_column <- dataset[[2]]

# Contents of optional Zip port are in ./src/

source("src/text.preprocessing.R");

# the output dictionary includes each word, its DF and its IDF

input.voc <- create.vocabulary(text\_column, minWordLen,

maxWordLen, minDF, maxDF)

# the output dictionary includes each word, its DF and its IDF

data.set <- calculate.IDF (input.voc, minDF, maxDF)

# Select the dictionary to be sent to the output Dataset port

maml.mapOutputPort("data.set")

2. Search second **Execute R Script module** and drag it to the canvas. Connect the first output of the last **Execute R Script module** to the second input of the new **Execute R Script module.** Connect thefirst output of the very first **Execute R Script module** to the first input of the new **Execute R Script module.** Connect the output of the text.preprocessing.zip dataset to the last input of the new **Execute R Script module**. Copy and paste the code below:

In this module, we’ll do TF-IDF Calculation, make sure to specify the same values for minWordLen and maxWordLen.

# Map 1-based optional input ports to variables

dataset <- maml.mapInputPort(1) # class: data.frame

input.dictionary <- maml.mapInputPort(2) # class: data.frame

##################################################

# Determine the following input parameters:-

# minimum length of a word to be included into the dictionary.

# Exclude any word if its length is less than \*minWordLen\* characters.

minWordLen <- 3

# maximum length of a word to be included into the dictionary.

# Exclude any word if its length is greater than \*maxWordLen\* characters.

maxWordLen <- 25

##################################################

# we assume that the text is the second column in the input data frame

label\_column <- dataset[[1]]

text\_column <- dataset[[2]]

# Contents of optional Zip port are in ./src/

source("src/text.preprocessing.R");

data.set <- calculate.TFIDF(text\_column, input.dictionary,

minWordLen, maxWordLen)

data.set <- cbind(label\_column, data.set)

# Select the document unigrams TF-IDF matrix to be sent to the output Dataset port

maml.mapOutputPort("data.set")

3.Search for **Filter based Feature Selection module** and drag it to the canvas.  specify the feature scoring method and the number of desired features. In the sample experiment, we selected the top 1,000 most relevant features. You may increase the number of desired features to get better classification performance. Connect the first output of the last **Execute R Script module** to the input of **Filter based Feature Selection module.** Launch Column selector and select label\_column.

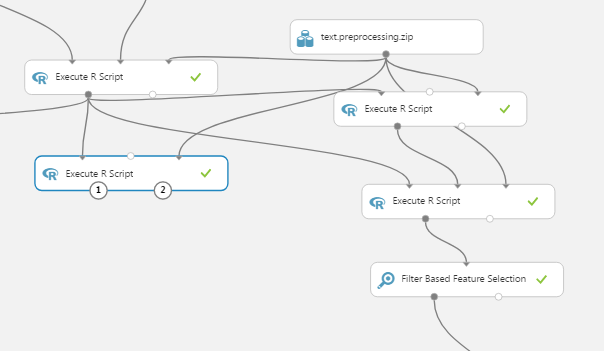
**Select:**

**Feature Scoring Method : Chi Squared**

**Operate on Feature columns only : Selected**

**Number of desired features : 1000**

4.Run the experiment.



### **Step 4: Train and evaluate models:**

Use the first **Split module** to split the data into two subsets. The first subset will be used to train the model and the second subset will be split in the next step into development/validation set and test set. In the sample experiment, we split the data into 70% and 30% respectively.

1.Search for **Split module** and drag it to the canvas. Connect the first output of first **Filter based Feature Selection module** to the input of **Split module.** Launch Column selector and label\_column. Select:

Splitting mode : Split Rows

Fraction of the rows in first output dataset : 0.7

Randomized split : Selected

Random seed : 0

Stratified split : True

Use the second **Split** module to split the data into two subset. The first subset will be used later by the **Sweep Parameters** module. The second subset is used as test set to evaluate the performance of the trained model. In the sample experiment, we split the 30% data sample into two halves. That is, each of the development set and the test set represents 15% of the input data.

2. Search for **Split module** and drag it to the canvas. Connect the second output of the first **Split module** to the input of second **Split module.** Launch Column selector and label\_column. Select:

Splitting mode : Split Rows

Fraction of the rows in first output dataset : 0.5

Randomized split : Selected

Random seed : 0

Stratified split : True

3.Search for **Tune Model Hyperparameters module** to get the optimal values for the underlying learning algorithm parameters. Connect the first output of the first **Split module** to the second input of **Tune Model Hyperparameters module**. Connect the first output of second **Split module** to the last input of **Tune Model Hyperparameters module.** Launch Column selector and label\_column. Select:

**Specify parameter sweeping mode: Random Sweep**

**Maximum number of runs on sweep parameters: 5**

**Random seed: 0**

**Metric for measuring performance for classification: AUC**

**Metric for measuring performance for regression: Mean Absolute Error**

4.Search for **Two-Class Logistic Regression** **module** for binary-class classification tasks. Connect the output of the **Two-Class Logistic Regression** **module** to thefirst input of the **Tune Model Hyperparameters module.** Select default settings with:

**Create trainer mode: Parameter Range**

5.Copy and paste the **Split modules, Tune Model Hyperparameters module, Two-Class Logistic Regression** **module** on the other side of the canvas. Connect the first output of the second Filter based Feature selection to the new **Split module** pasted.

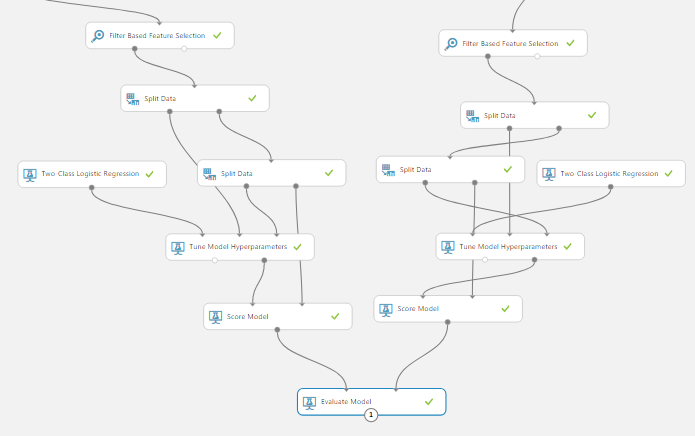
**Step5: Score Models**

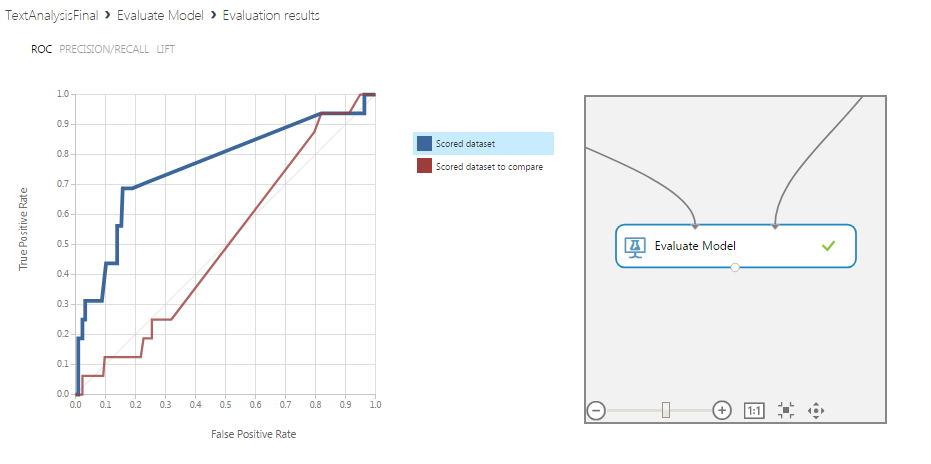
1.Search for **Score model module** and drag it to the canvas. Connect the second output **Tune Model Hyperparameters module** to the first input of the **Score model module.** Connect the second output of the second **Split module** to the second input of the **Score model module** Select Append score column.

2. Search for **Score model module** and drag it to the canvas. Connect the second output **Tune Model Hyperparameters module** to the first input of the **Score model module.** Connect the second output of the second **Split module** to the second input of the **Score model module** Select Append score column.

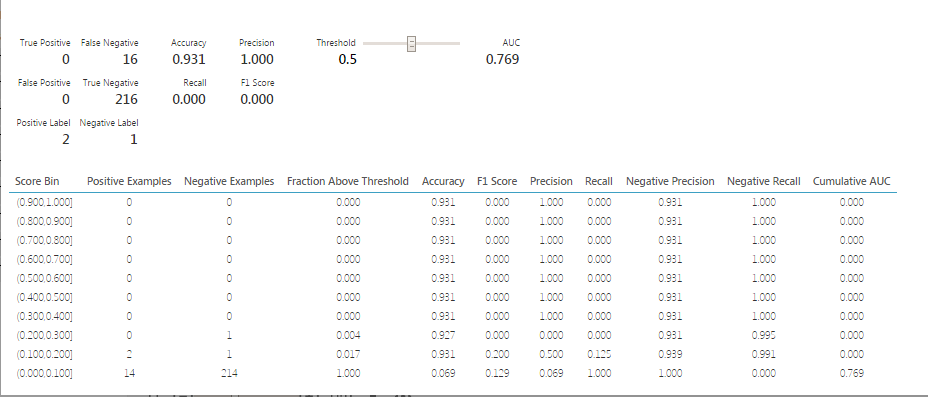
**Step 6: Evaluate Model**

Search for the **Evaluate Model module.** Connect the outputs of the Score models to the input of the **Evaluate Model module.**

****



Click the output port of the **Evaluate Model** module and visualize the results of comparison between the two trained models: N-grams model (in blue color in the graphs below) and unigrams model (in red color in the graphs below)



**Summary:**

In this tutorial, we have constructed and evaluated a Yelp tip text analysis by comparing uni-gram and n-gram feature extraction algorithms. Specifically, we:

* Evaluated the score models for both algorithms and found that N-gram feature extraction using R scripts is best model.
* Determined the most frequent and relevant words from the text that can easily help the users and business.
* Helped to understand customer sentiment and satisfaction of a business.

**References**:

1. Dataset URL: ww[w.yelp.com/dataset\_challenge/dataset](https://www.yelp.com/dataset_challenge/dataset)

1. Github: <https://github.com/rsingh26/DataScience/tree/master/MachineLearning>
2. URL of Refereces :   <https://gallery.cortanaintelligence.com/Experiment/Text-Classification-Step-1-of-5-data-preparation-3>