NLP Day 2019  
Predicting typed words using n-grams  
Instruction

## Building Azure Machine Learning Studio Model

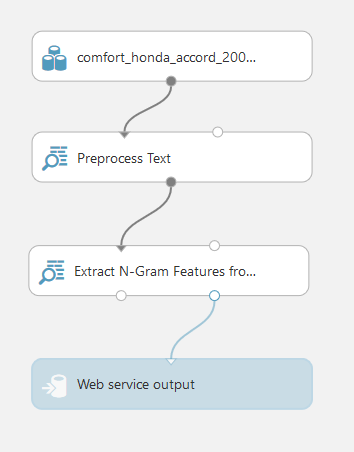
1. Sign in to Azure Machine Learning Studio

* Guest workspace – sign in using Microsoft account
* Free workspace – 8 hour trial without signing in (setting up webservice not available)

1. Create new Experiment

### Azure Machine Learning Experiment

#### N-Gram Extractor



##### Import data source

##### Preprocess Text

###### Select column: **[Column index 1]**

###### Select language: **[English]**

###### Remove by part of speech: **[False]**

Select this option if you want to apply part-of-speech analysis. You can then use the part-of-speech tags to remove certain classes of words.

* + **Remove nouns**: Select this option to remove nouns.
  + **Remove adjectives**: Select this option to remove adjectives.
  + **Remove verbs**: Select this option to remove verbs.

###### Remove stop words: **[True]**

Select this option if you want to apply a predefined stopword list to the text column. Stop word removal is performed before any other processes.

###### Lemmatization: **[True]**

Select this option if you want words to be represented in their canonical form. This option is useful for reducing the number of unique occurrences of otherwise similar text tokens.

###### Detect sentences: **[True]**

Select this option if you want the module to insert a sentence boundary mark when performing analysis.

This module uses a series of three pipe characters ||| to represent the sentence terminator.

###### Normalize case to lowercase: **[True]**

Select this option if you want to convert ASCII uppercase characters to their lowercase forms.

If characters are not normalized, the same word in uppercase and lowercase letters is considered two different words: for example, “AM” is the same as “am”

###### Remove numbers: **[True]**

Select this option to remove all numeric characters for the specified language.

###### Remove special characters: **[True]**

Use this option to replace any non-alphanumeric special characters with the pipe | character.

###### Remove duplicate characters: **[True]**

Select this option to remove any sequences that repeat characters. For example, a sequence like "aaaaa" would be removed.

###### Remove email addresses: **[True]**

Select this option to remove any sequence of the format <string>@<string>.

###### Remove URLs: **[True]**

Select this option to remove any sequence that includes the following URL prefixes:

* + http, https
  + ftp
  + www

###### Expand verb contractions: **[True]**

This option applies only to languages that use verb contractions; currently, English only.

For example, by selecting this option, you could replace the phrase "wouldn't stay there" with "would not stay there".

###### Normalize backslashes to slashes: **[True]**

Select this option to map all instances of \\ to /.

###### Split tokens on special characters: **[True]**

Select this option if you want to break words on characters such as &, -, and so forth.

For example, the string MS-WORD would be separated into two tokens, MS and WORD.

##### Extract N-Gram Features from Text

###### Select column: **[Column index 2]**

###### Vocabulary mode: **[Create]**

###### N-Grams size: **[3]**

Number that indicates the maximum size of the n-grams to extract and store.

For example, if you type 3, unigrams, bigrams, and trigrams will be created.

###### K-Skip size: **[0]**

Maximum number of characters that can be different when identifying variants of n-grams. If the value of k is set to 0, n-grams can be created only from a unique, contiguous sequence of characters.

For example, assume that your dictionary contains the unigram "computer". A k value of 0 would mean that "computer" is the only valid unigram. If you increase the value of k to 1, you can skip over one intervening character, which lets you find more similar sequences. A skip-gram with a k value of 1 would differ by one character from the 0-k unigram. Thus, the skip-grams "conputer" and "compuuter" would both be considered part of the same dictionary entry as "computer". Setting the k value to 2 would match even more dissimilar words.

###### Weighting function: **[Binary Weight]**

Required only if you merge or update vocabularies. It specifies how terms in the two vocabularies and their scores should be weighted against each other.

###### Minimum word length: **[2]**

Minimum word length of strings that can be analyzed.

For example, assume the minimum word length was set to 3 (the default value), and you had one input that had a single word, and another that had some short text like "nice place". Both rows would be ignored.

###### Maximum word length: **[25]**

Maximum number of letters that can be used in any single word in an n-gram.

By default, up to 25 characters per word or token are allowed. Words longer than that are removed, on the assumption that they are possibly sequences of arbitrary characters rather than actual lexical items.

###### Minimum n-gram document absolute frequency: **[1]**

Number that indicates the minimum occurrences required for any single word or token to be included in the n-gram dictionary.

For example, if you use the default value of 5, any n-gram or skip-gram must appear at least five times in the corpus to be included in the n-gram dictionary.

###### Maximum n-gram document ratio: **[1]**

Number that represents this ratio: the number of rows that contain a particular n-gram, over the number of rows in the overall corpus.

For example, a ratio of 1 would indicate that, even if a specific n-gram is present in every row, the n-gram can be added to the n-gram dictionary. More typically, a word that occurs in every row would be considered a noise word and would be removed. To filter out domain-dependent noise words, try reducing this ratio.

Important

The rate of occurrence of particular words is not uniform, but varies from document to document. For example, if you are analyzing customer comments about a specific product, the product name might be very high frequency and close to a noise word, but be a significant term in other contexts.

###### Detect out-of-vocabulary rows: **[True]**

Select this option, if you want to generate an indicator for any rows that contain words not in the n-gram vocabulary, which are called "out of vocabulary" (OOV) words.

All lexicons are finite; therefore, your text corpus is almost guaranteed to include words that are not in the lexicon or n-gram dictionary. However, such words can have various effects on language models, including higher error rates compared to in-vocabulary (IV) words. Depending on your domain, these OOV words might represent important content words.

By identifying rows that contains these words, you can either compensate for the effects of these terms, or handle the terms and related rows separately.

###### Mark begin-of-sentence: **[False]**

Select this option to add a special character sequence that indicates the beginning of a sentence in your n-gram dictionary. Prefixing n-grams that start a sentence with a special character is common in text analysis and can be useful in analyzing discourse boundaries.

Azure ML Studio inserts the symbol |||. You cannot specify a custom character.

###### Normalize n-gram feature vectors: **[True]**

Select this option, if you want to normalize the feature vectors. When you do this, each n-gram feature vector is divided by its L2 norm.

Normalization is used by default.

###### Use filter-based feature selection: **[False]**

Enable additional options for managing the size of your text feature vector.

* + Feature selection can be helpful in reducing the dimensionality of your n-grams.
  + When you do not apply filter selection, all possible n-grams are created, increasing coverage at the expense of making the dictionary longer and possibly including many infrequent terms.
  + In a small corpus, using feature selection can greatly reduce the number of terms that are created.
  + For more information, see Filter Based Feature Selection.

If you are using feature selection, you must select a method from the Feature scoring method drop-down list:

* + **PearsonCorrelation**: Computes Pearson's correlation based on the label column value and the text vector.
  + **MutualInformation**: Computes a mutual information score, based on the label column value and the text vector.
  + **KendallCorrelation**: Computes Kendall's correlation, based on the label column value and the text vector.
  + **SpearmanCorrelation**: Computes the Spearman correlation, based on the label column value and the text vector.
  + **ChiSquared**: Uses the chi-squared method to calculate the correlation between the label column value and the text vector.
  + **FisherScore**: Computes the Fisher score for the label column value and the text vector.
  + **Count-based feature selection**: Creates new features based on the counts of values. A label column is not required with this method.

Depending on the method you choose, set one of the following options:

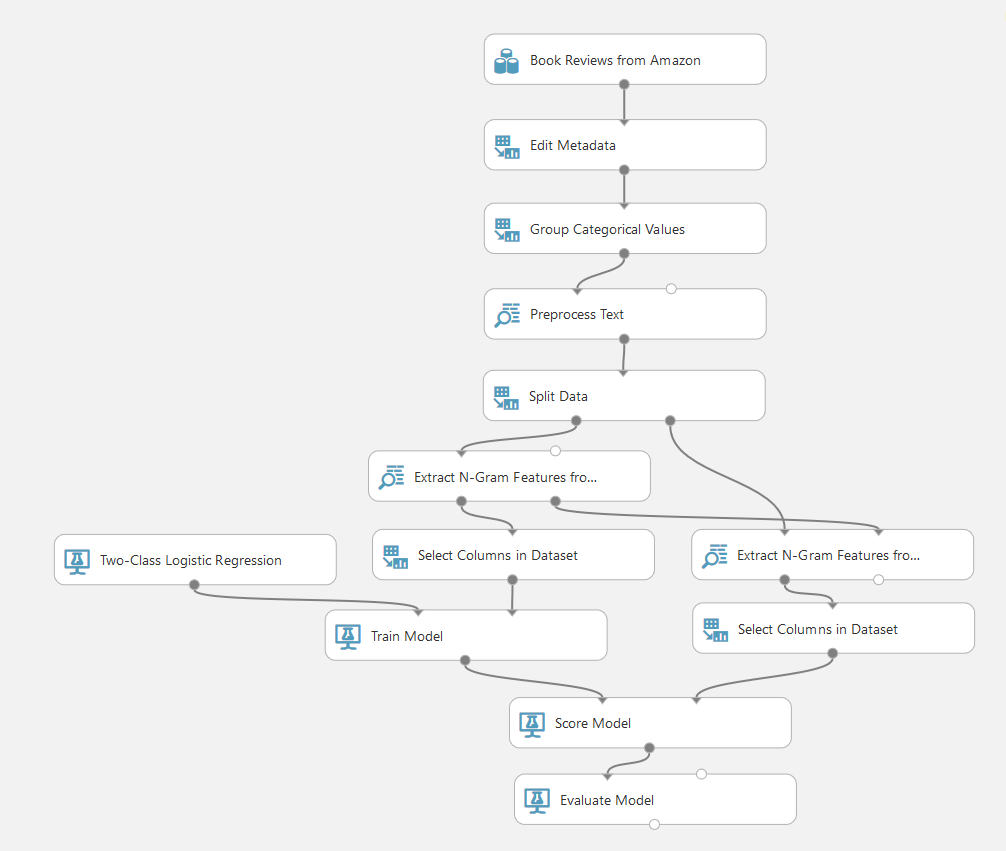
* + **Number of desired features**: Required if you use any feature selection method other than count-based feature selection.

In the process of feature selection, all n-grams get a feature score, and n-grams are ranked by score. The value you set here determines how many of the most highly-ranked features are output. N-grams with lower feature scores are discarded.

* + **Minimum number of non-zero elements**: Required if you use count-based feature selection.

Type a whole number that represents the minimum number of total instances required to tabulate counts for a potential feature.

#### Sentiment Analyzer



##### Use sample data source: [Book Reviews from Amazon]

##### Edit Metadata

Convert data in column 1 from numeric to categorical.

###### Select column: **[Column name Col1]**

###### Data type: **[Unchanged]**

###### Categorical: **[Make categorical]**

###### Fields: **[Unchanged]**

##### Group Categorical Values

Group categories into classes.

###### Select column: **[Column name Col1]**

###### Output mode: **[Inplace]**

###### Default level name: **["low"]**

###### New number of levels: **[2]**

###### Name of new level 1: **["high"]**

###### Comma-separated list of old levels to map to new level 1: **[4,5]**

##### Preprocess Text

Normalize text.

###### Select column: **[Column name Col2]**

##### Split Data

Split data to training and testing collection.

###### Fraction of rows in the first output dataset: **[0,75]**

##### a. Extract N-Gram Features from Text

###### Select column: **[Column name Preprocessed Col2]**

###### Vocabulary mode: **[Create]**

###### N-Grams size: **[2]**

###### K-Skip size: **[0]**

###### Weighting function: **[TF-IDF Weight]**

###### Minimum word length: **[3]**

###### Maximum word length: **[25]**

###### Minimum n-gram document absolute frequency: **[5]**

###### Maximum n-gram document ratio: **[0,8]**

###### Detect out-of-vocabulary rows: **[True]**

###### Mark begin-of-sentence: **[False]**

###### Normalize n-gram feature vectors: **[False]**

###### Use filter-based feature selection: **[True]**

###### Select column: **[Column name Col1]**

###### Number of desired features: **[1000]**

##### b. Extract N-Gram Features from Text

###### Select column: **[Column name Preprocessed Col2]**

###### Vocabulary mode: **[ReadOnly]**

###### N-Grams size: **[2]**

###### K-Skip size: **[0]**

###### Weighting function: **[TF-IDF Weight]**

###### Minimum word length: **[3]**

###### Maximum word length: **[25]**

###### Minimum n-gram document absolute frequency: **[1]**

###### Maximum n-gram document ratio: **[1]**

###### Detect out-of-vocabulary rows: **[True]**

###### Mark begin-of-sentence: **[False]**

###### Normalize n-gram feature vectors: **[False]**

###### Use filter-based feature selection: **[False]**

##### a. Select columns in Dataset

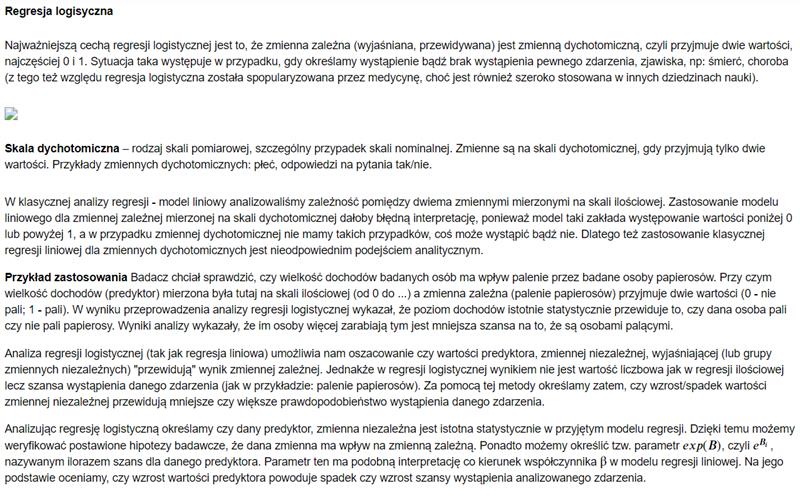
###### Select column: **[All columns except: Col2, preprocessed Col2, NumUniqueNgrams, NGramsString]**

##### b. Select columns in Dataset

###### Select column: **[All columns except: Col2, preprocessed Col2, NumUniqueNgrams, NGramsString]**

##### Two-class logistic regression

An untrained model. Default parameter values.



##### Train Model

###### Select column: **[Column name Col1]**

###### Create trainer mode: **[Single Parameter]**

Specify how you want the model to be trained, by setting the Create trainer mode option.

* Single Parameter: If you know how you want to configure the model, you can provide a specific set of values as arguments.
* Parameter Range: If you are not sure of the best parameters, you can find the optimal parameters by specifying multiple values and using the Tune Model Hyperparameters module to find the optimal configuration. The trainer iterates over multiple combinations of the settings and determines the combination of values that produces the best model.

###### Optimization tolerance: **[1e-7]**

For **Optimization tolerance**, specify a threshold value to use when optimizing the model. If the improvement between iterations falls below the specified threshold, the algorithm is considered to have converged on a solution, and training stops.

###### L1 regularization weight: **[1]**

###### L2 regularization weight: **[1]**

For L1 regularization weight and L2 regularization weight, type a value to use for the regularization parameters L1 and L2. A non-zero value is recommended for both.

Regularization is a method for preventing overfitting by penalizing models with extreme coefficient values. Regularization works by adding the penalty that is associated with coefficient values to the error of the hypothesis. Thus, an accurate model with extreme coefficient values would be penalized more, but a less accurate model with more conservative values would be penalized less.

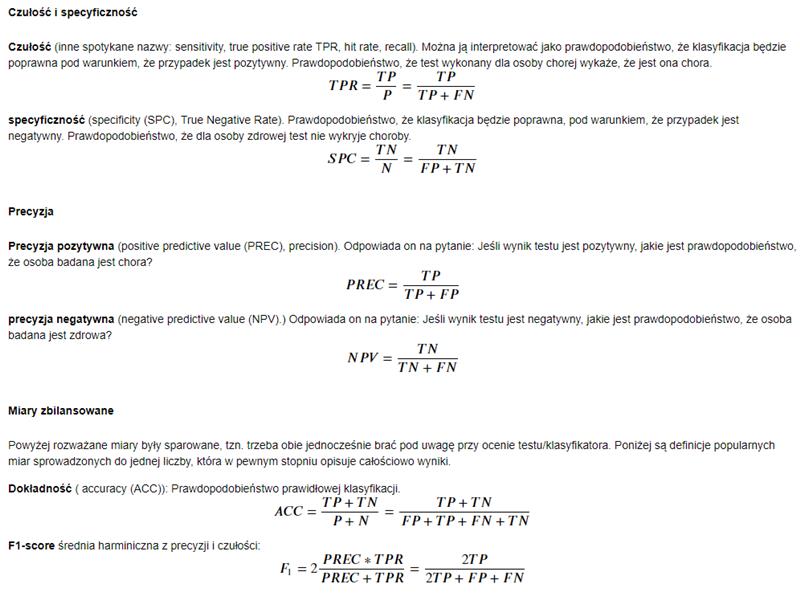
L1 and L2 regularization have different effects and uses.

* + L1 can be applied to sparse models, which is useful when working with high-dimensional data.
  + In contrast, L2 regularization is preferable for data that is not sparse.

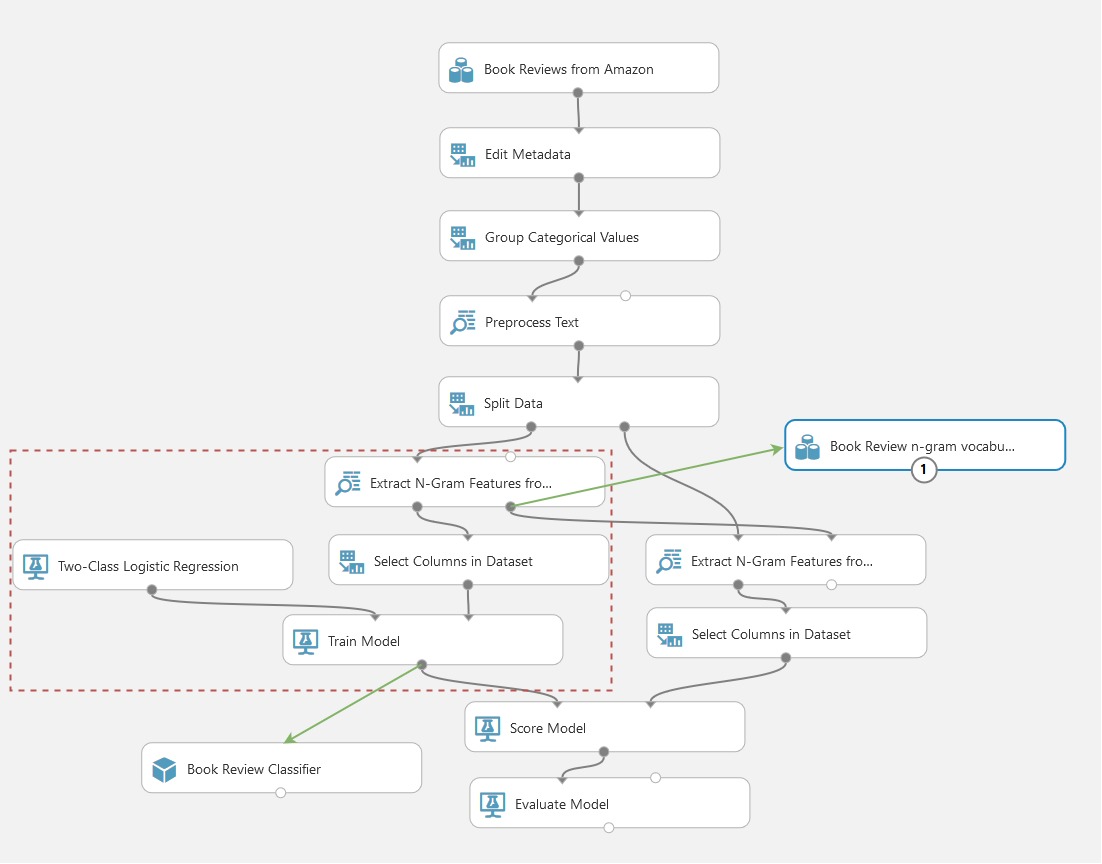
This algorithm supports a linear combination of L1 and L2 regularization values: that is,   
if x = L1 and y = L2, then ax + by = c defines the linear span of the regularization terms.

##### Score Model

##### Evaluate Model



#### Sentiment Analyzer – web service



Modified for webservice purpose.

