Hons. Degree in Computing H4016 Text Mining & Information Retrieval

Lab sheet #2 = Text Mining with Rapid Miner

G.Gray

Recap

- Last weeks lab used Rapid Miner to run classification algorithms on a structured data set (risk.csv)
  - Read in a dataset
  - Use visualisation options to view the data graphically
  - Run the data set through a decision tree algorithm
  - Output the results to a decision matrix
  - Optimise parameter settings

- The week's lab looks at preparing text data for data mining, and links with lecture: Unit 2-1

Overview

Objective:

- Introduce and Use the Rapid Miner Text Plugin

- Agenda:
  - Install the text plugin
  - Read in sample texts
  - Manipulate texts in preparation for data mining

Exercises

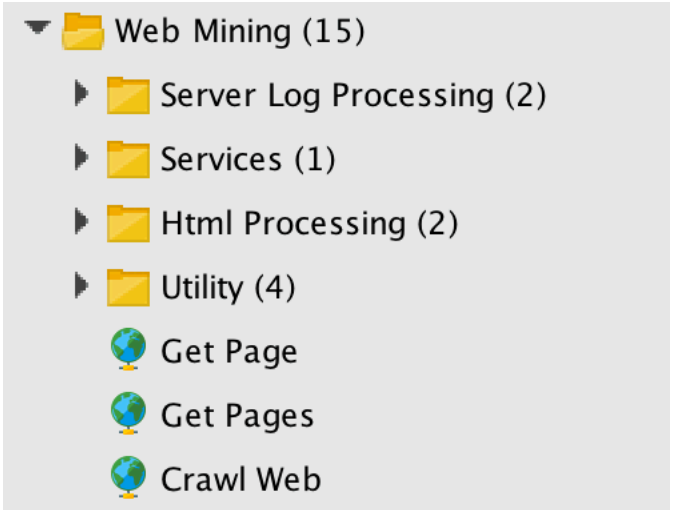
There are a number of exercises through out the slides, and repeated again on the last slide.

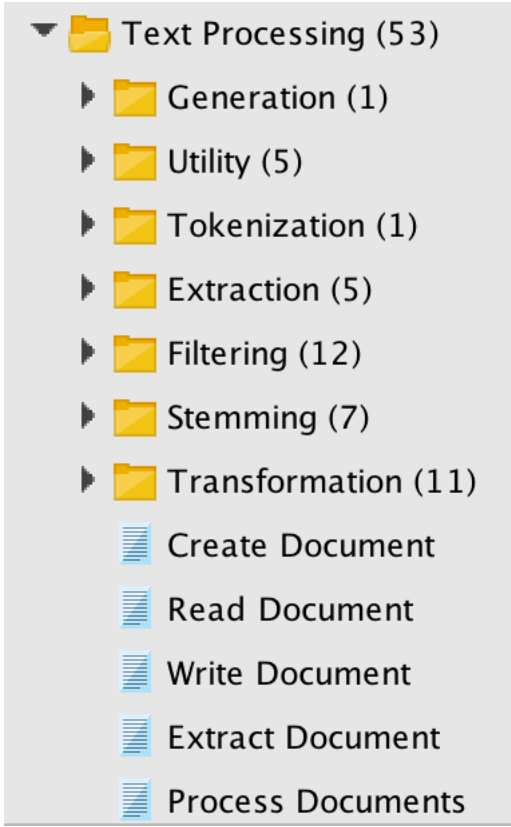
Put answers to these in a separate word document called ExerciseSolutionLab#2-B000*nnnnn* (student number) and upload to moodle at the end of the lab.

Installing the Text Mining Plugin

- To install/update the text operators, start up RapidMiner. Under Extensions select Marketplace.
- Enter 'text' in the search box, and select Text Processing and Web Mining extensions to install.
- Click on Install, accept Terms and Conditions, and restart RapidMiner.
- You should see two additional folders under Extensions on the operator tree.

Optionally you can also install the Wordnet and AYLIEN text analysis extensions





Operators under 'Text Processing'

The following slides give an overview of the operators available under the Text Processing extension:

Tokenisation

Tokenise splits a text input into tokens. Some of the options are:

- non-letters: Takes each individual word to be a token. The splitting points are any non-letter character
- Specific characters: The user specifies the character(s) at which tokens should be split. You may do this is you wanted tokens to be sentences rather than individual words.
- Regular expression: this is the most specialised option that may be needed in less typical circumstances. Tokens are defined using a regular expression (following the syntax for Java REGEX)

Extraction

- Extractors add meta data about the document such as its length or number of tokens, and adds this information as a special attribute in the dataset.

- There are a number of operators which predefine what information is to be extracted, and one operator, extract information, which allows the user to define what should be extracted.

Filtering

Filters determine which words are to be ignored when considering terms for the document vector. Filters can be a list of stopwords, or can be based on word length (ignoring smaller words)

Filters include the following operators:

- Filter Tokens (by length): Filters terms based on a minimal number of characters they must contain to be considered (kept).
- **Filter Stopword (English)**: Standard stopwords list for English texts.
- Filter Stopword (German): Standard stopwords list for German texts.
- Filter Stopword (Dictionary): Filters terms based on a list of expressions/terms provided in an external file.

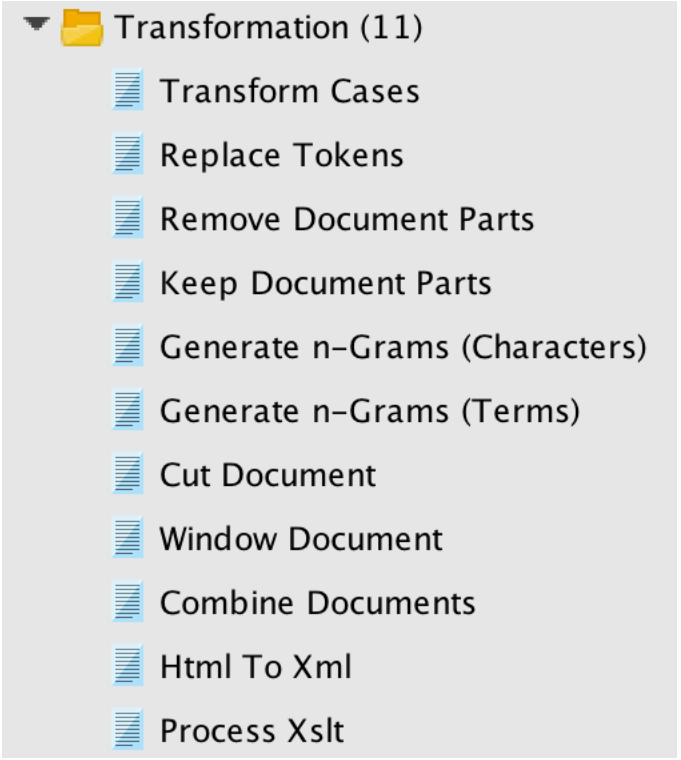
Stemmers

Stemmers replace words with an agreed 'base' term for the word.

- Stem (dictionary) uses a dictionary to decide on the base term. Dictionary entries are of the form
  - <base_form>: <extression>
- Stem (German): A stemmer for German texts.
- Stem (Lovins): automatically stems words to a base form, based on a predefined set of rules. See http://snowball.tartarus.org/algorithms/lovins/stemmer.html for more information.
- Stem (Porters): an commonly used alternative to the Lovins algorithm above. For more information see http://tartarus.org/martin/PorterStemmer/
- Stem (Snowball) for different languages. Snowball is a language for defining stemmers. http://snowball.tartarus.org/
- **Note**: you may need to change all text to lower case before using a stemmer to avoid errors in case sensitive comparisons using transform cases under Text Processing / transformations.

Transformations

- Many of the transformation operators are self explanatory as they are standard string functions, apart possibly from the two n-gram operators which are explained in lecture notes.



The following slides illustrate how to use these operators on three folders of text documents.

Take note of:

1. The order the operators are applied

and

2. What each output represents.

Order operators are applied

1) Process document to read in the text files. All other operators are embedded within, this operator, such as:

2) Tokenise

3) Filters

4) Transformations and extractions

5) Stemmers

Reading in data

- Take a look at the following video to view the four methods to read unstructered data into Rapid Miner.
- We will be using the last of these methods, Process Documents from Files:

Loading Text in Rapid Miner, Neil McGuigan

This web site has a series of videos on Text Mining using Rapid Miner.

Process Documents from Files

Download the four folders of text files from moodle (lab 2 datasets). The four folders are called 'crime', 'healthcare', 'kenya', and 'unlabelled'.

Start a new process, call it Lab2-ReadInFiles.

Add Text Processing / Process Documents from Files to the process window.

Select 'Edit List' in the Text Directories parameter.

Give the class name as Crime, and under directory, navigate to where the folder of crime texts was saved.

Click 'Add Entry', and repeat the process for both Healthcare and Kenya.

Loading text files

- Process Documents is a meta process which expects other processes to be nested with in it.
- For now, double click on the operator, and just connect the input port (doc) to the output port (doc) and run the process.
- This creates a data set with:
  - 15 examples (rows) one for each document
  - 2 special attributes, ID and Label
  - 15 other attributes, each of which is the entire text for that document
- The 15 attributes need to be replaced with terms from the bag of words for the document, rather than the entire text.

Reading in data

- Click on the link below to take a look at a video on Rapid Miner preprocessing options.

Processing Text in Rapid Miner, Neil McGuigan

Tokeniser

- The first step is to add a Tokenize operator as a child node of Process Documents from Text (i.e. nested with in it).
- Double click on Process Documents from Text, and add the tokenise operator.
- Leave mode at the default value of 'non letters'
- Connect the ports and run the process.

Output from tokeniser

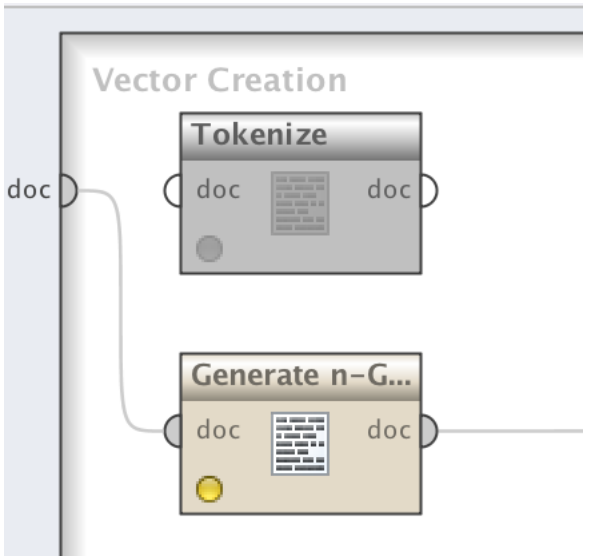
There are two outputs from "process documents":

1. Bag of words, which is called the word list. A list of each term, and how often it occurs within documents in your dataset. THIS IS NOT THE DATASET, it provides information to the user, but is not used by mining algorithms.
2. The dataset (also called the document vector or example set), which has:
   - One row for each document in your collection (15 rows in this case)
   - One attribute for each term in your bag of words.
   - 2 special attributes, ID and Label
   - The data in the dataset represents how often that attribute (term) appeared in that document.

Using a character n-gram

- Take a note of the number of attributes (terms) in the dataset.

- Disable **tokeniser,** instead use the operator Generate n-grams (Characters) under transformations.
- Decide on a length for your n-grams. The default is 3.



- Run the process again.
- How many attributes do you have now?
- Disable the n-gram operator, and restore the tokeniser. We will be using tokeniser for the remaining exercises

Filters

- Many of the tokens in the bag of words here are not useful for classifying documents by topic. For example terms like: are, the, by, a etc
- These stop words can be removed. Again this operator will be a child operator of Process Documents from Text.
- Add the operator Filter Stop Words (English) to the process after the tokenise operator.
- Connect the ports and run the process.

Filters

- The dataset is now down to about 227 attributes.
- In addition to the predefined stop words, Rapid Miner allows you to specify your own stop words, which can be tailored to the domains of the texts.
- The format for this list is a text file, with one word / term per line.
- **Exercise 1:** Have a look at the seven terms in stopwordLab2.txt from the lab2 files on studentshare and moodle. If not already done, copy this file to your own working directory, and add additional terms to this list. Paste this list into a word document as your answer to exercise 1.
- To use this list, add the operator Filter Stop Words (Dictionary) to the process.
- This operator has one parameter, file, which accepts the full path name to the file of stopwords. Direct it to your list of stopwords.

Filters

- Connect the ports and run the process.
- How many attributes are now in the dataset? Include this figure in your answer to **exercise 1**.

Other filters provided by rapid miner:

- Filter Token (by Length) which removes all tokens smaller than (or larger than) a specified length. The default value is remove words with:
  - less than 4 characters, meaning terms of length 3 or less are removed,
  - or more than 25 characters.
- Exercise 2: Add this filter to the process. How many attributes are left? Do you think this filter is useful?

Phrases

- Currently, the bag of words includes single terms only. To add phrases, add the text processing/ transformations / generate n-Grams(terms) operator.
  - This operator has one parameter, specifying the maximum number of words to join together to make a phrase. Set this to '2'.
  - Run the process. How many additional phrases added to our bag of words?

Phrases

- Rapidminer has a filter operator that allows you to filter tokens based on their part of speech.
- Add Filter Tokens (by POS)
  - To include nouns only, set the expression parameter to N.*
  - To include nouns and verbs, set the expression parameter to N.*|V.*
  - Try including NOUNS ONLY. If this is placed BEFORE N-gram(token) then phrases will be limited to Nouns and Verbs are are next to each other in the original text.

WARNING: This operator takes a while to run. You can remove it from your process when you finish Exercise 3.

- Exercise 3: Include some of the phrases generated in your lab report.

Stemmers

- Stemmers are a group of algorithms with convert terms into another format.
- A Dictionary Stemmer can be used to specify homonyms, or other base words to be used, and the list of words to be converted to this base form. The list is in the format: <base_form>:<expression>
  - Note: phrases can not be included, just single terms.
- Have a look at the list in synonymsLab2.txt in the lab2 files on moodle. If not already done so, copy it to your own working directory.
- Add a Stem (Dictionary) to the process after the filters, and direct the file parameter to synonymsLab2.txt.

Stemmers

- Connect the ports and run the process.
- Exercise 4: Add additional lists to synonymsLab2.txt. They can be synonyms, or other conversions such as converting towns in Kenya to the word Kenya.
- Give your additional lists as a solution to the exercise, and state how many attributes now remain in the file.

- RapidMiner implements a Lovins stemmers and a Porters stemmer.
- Exercise 5: Add a Stem (Lovins) – evaluate the words produced. Do you think its an improvement?
- Exercise 6: Replace Stem (Lovins) with Stem (Porters). Compare the results.

SAVE

- Before continuing with the remaining exercises:
  - SAVE the process,
  - SAVE a copy of the process, as you will need this in next weeks lab. Call is lab3BasicProcess.
- Return to your original process - Lab2-ReadInFiles

Mining the text

- Exercise 7: As per last week, generate a decision tree to predict each class, and then apply and evaluate the model. How accurate is the classifier?
  - Attach an X-Validation block to Process Documents

- Exercise 8: Does the inclusion / removal of stemmers effect the accuracy?
- Exercise 9: Can you improve the accuracy using another classification algorithm?

Note: Some learners can only be applied to a binary class. In this dataset, the class can have three values (healthcare, kenya, crime).

Learners you can use: NaiveBayes, LibSVMLearner, nearestNeighbour, basicRuleLearner

Exercises (repeated from slides)

- Exercise 1: Give your list of stopwords for the 15 texts. Also state how many attributes remain once you apply your stopword list.
- Exercise 2: How many attributes are left when words of length 3 or less are removed? Do you think this filter is useful?
- Exercise 3: Include some of the phrases generated in your lab report.
- Exercise 4: Give your list of stemwords/word conversion for the 15 texts. Also state how many attributes remain once you apply these conversions.
- Exercise 5: Add a LovinsStemmer – evaluate the words produced. Do you think it its an improvement?
- Exercise 6: Replace LovinStemmers with PortersStemmer. Compare the results.
- Exercise 7: As per last week, generate a decision tree to predict each class, and then apply and evaluate the model. How accurate is the classifier?
- Exercise 8: Does the inclusion / removal of stemmers effect the accuracy?
- Exercise 9: Can you improve the accuracy using another classification algorithm?