Hons. Degree in Computing H4016 Text Mining & Information Retrieval

Lab sheet #3 = Text Mining with Rapid Miner continued . . .

Context

- **Done to date**:
  - Use IO functions to read in structured data (lab1)
  - Use IO functions to read in unstructured data which is organised as text files in a folder(s) (lab2)
  - Apply text processing techniques to the dataset (tokenisers, filters & stemmers) (lab2)
  - Use visualisation options to view data graphically (lab1*lect 2.3)
  - Run a data set through a classification algorithm & evaluate the results in a decision matrix (lab1&2)

- To do:
  1) Investigate options for attribute counts (measures of how frequently a term occurs in a document)
  2) Convert a document into a document vector.
  3) Using web data
  4) Clustering text documents

Objective

1) Look in more detail at the Process Documents from Files parameters, to look at practical implementations of topics covered in lecture 3.

2) Generate a document vector of unlabelled data.

. . but first, the following slides look at some additional menu options and short cuts in RapidMiner:

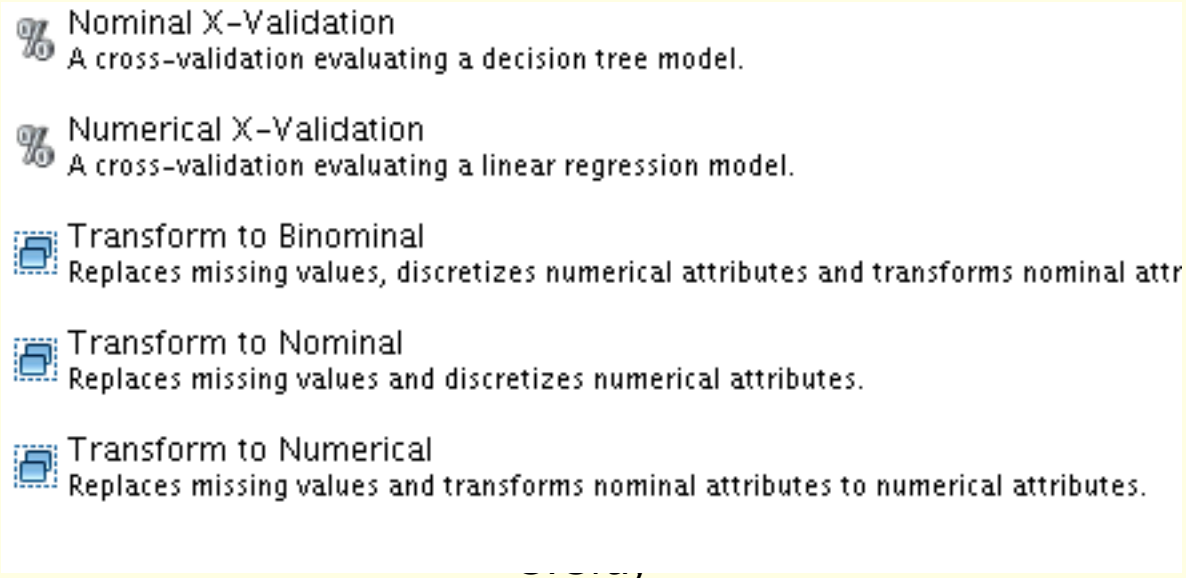
Exercises

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#3-B000*nnnnn* (student number) and upload to moodle at the end of the lab.

3. Building blocks

- As you may know, a building block is a group of operators which are commonly used together.
- To save you creating the group each time, you can save the operators as a building block. This block can then be added to other processes as needed.
- Rapid Miner has 5 building blocks pre defined. You can also add user defined blocks to this list, found under edit / save as building block.



Creating a building block

- Start a new process.
- Add a Process Documents from Text File operator (used in lab 2).
- Double click on this operator, and add the following as child nodes:
  1) A tokenize operator (string tokeniser)
  2) A Filter Stop Works (English) operator
  3) A Filter Tokens (by Length) operator.
- This is going to be our basic building block for all text mining processes.
- Navigate back up to Process Documents from Text File and click on the operator to select it.
- Go to Edit / Save as building block to save this group of operators. Call it Simple Text Input. With a description of input docs from files, tokenize and filters.

Some of the other building block:

- Return to the process and add the building block NominalXValidate.
- Expand the new operator. This is the list of operators that was required at the end of lab2 to test a decision tree classifier on the training data. The learner used (decision tree) can be replaced.

Other building blocks

- Numerial X-Validate: as above but uses linear regression instead of a decision tree.
- Transform to Binomial: data preparation stream which replaces missing values, and converts numeric data first into bins, and then into binary values.
- Transform to Nominal: as above, but without the last step to convert into binary data.
- Transform to Numeric: as above, but convert nominal data to numeric data rather than the other way around as above.

Back to text mining . . ., and a more detailed look

at Process Documents from Files

Counts and weights

- Open lab3BasicProcess which you created during lab2.
- As child nodes of 'Process Documents from Files', make sure you have a: tokenize, filter stop word (English), filter stop (dictionary) to read in the synonyms file from last week, and a porters stemmer.

- Run the process to check it.

Counts and weights

- Return to edit mode, and select the Process Documents from Files operator to view its parameters. Make sure you are in expert mode.

- The seventh parameter down is called '**vector creation**'. Select the drop down box for this parameter to view the options:

  1. TermFrequency: Normalised count of occurrences
  2. TermOccurrences: Count of occurrences
  3. TF-IDF: Counts adjusted using IDF, and then normalised.
  4. BinaryOccurrences: 0 if word is not present in the document; 1 if word is present in the document.

Counts and weights

1) Set it first to count TermOccurrences. Run and view the data view of the example set.
2) Scroll over to rest and rift.
3) Click on rift twice to order the documents by the count in this column (in descending order)
4) **Exercise 1**: Both Risen and Rift have a count of 1 in K4.txt. Risen does not appear in any another documents, Rift appears in two other documents. Based on you knowledge of IDF, explain which term will get the higher weighting, and why?

Counts and weights

- Return to edit mode.
- Change vector_creation to TFIDF.
- Run the process.
- **Exercise 2**: What has happened to the count of '1' for both risen and rift? Explain how this figure is derived.

Pruning very frequent and very infrequent terms

Before completing the following exercise, add a X-Validation block to your process, and replace the Decision Tree with k-NN. Record the model accuracy.

Prune above and below

- Unit 3 looked at the usefulness of words based on their frequency: Very frequent terms, or very infrequent terms, are not useful.
- The 'prune' parameters on Process Documents from Files allows you to filter infrequent (prune below) and very frequent (prune above) terms.
- Change prune method to absolute.
- Pruning is done based on the number of documents a term appears in, .e.g.
  - Prune below = 5 means prune all terms that appear in less the five documents.
  - Prune above = 10 means prune all terms that appear in more the ten documents.

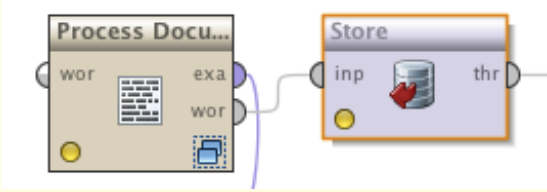
- Set prune below to 3. This will remove all terms that appear in less than 3 documents, leaving 15 terms.

Exercise 3: How accurate is a classification model on 15 terms compared to model accuracy before pruning the terms?

Next topic: Saving the reduced list of attributes.

Save word list

- This final list of terms in the document vector is the data set on which classification / clustering models are trained (e.g. k-NN, decision tree, etc.).
- If a new document is to be tested against one of these models, its document vector must be an **exact** match to the vector on which the model was trained.
- Therefore, you need to save this final list of words as follows...



  - Add a store operator to the process. Connect the 'wor' output port of the process document operator to the 'inp' input of 'store'.
  - In the repository entry parameter of 'store', click the browse button, navigate to where in your repository you want to save the word list, and call it lab3wordlist.

Save word list

- Save the process and run it.

- The stored object (a word list) can now be used as the wordlist in a new process, **dictating the terms that will be used in the document vector.**

What have you just saved?

The definition of your dataset (word vectors), i.e. the column names, as shown below.

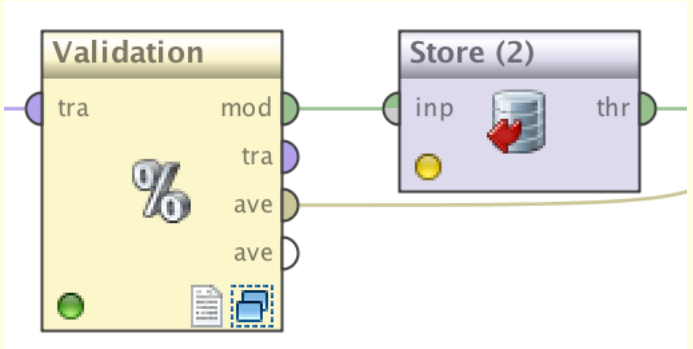
word	total occurrences	document occurences	crime	healthcare	kneya
annan	4	3	0	0	4
clashes	3	3	0	0	3
court	3	3	2	1	0
crime	3	3	3	0	0
former	3	3	0	0	3
hse	3	3	0	3	0
kenya	13	5	0	0	13
kofi	3	3	0	0	3
people	3	3	0	0	3
rights	4	3	2	1	1
un	3	3	0	0	3
valley	4	3	0	0	4
western	3	3	0	0	3

Saving a classification model . . .

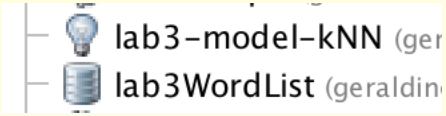
Saving a classification model

The k-NN model must also be saved to apply it to ‘new’ data.

Add another store operator to your process.
Connect the ‘mod’ model port of X-Validation to the ‘inp’ input port of the store operator. Call the repository entry ‘lab3-model-kNN’



Run the process. The model should appear in your repository.



Saving a classification model

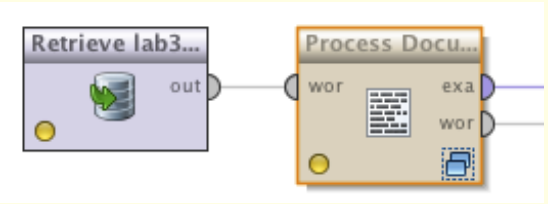
Try saving a decision tree model as well:

1. Replace k-NN with DT in X-Validation
2. Rename the repository entry to lab3-model-DT in the store operator
3. Run the process again.

Load word list to a new process

The following steps will read in new documents, and generate document vectors for those documents using the terms saved in the last slide.

- Start a new process called mineUnseenData.
- Drag lab3wordlist from your repository onto the process window to retrieve it (as you would do with a dataset).
- Copy the process document from files operator from your training process, as it will have the correct preprocessing steps embedded in it.
- Connect the output of the retrieve operator to the 'wor' input port of a new process document from file operator.



Load word list to a new process

- Click on process document from files to view its parameters.
    - Click on Edit List (the first parameter).
- Remove the three entries and add an entry pointing to the 'unLabelled' folder of documents in the lab2 files. Give it a label of unknown

Load word list to a new process

Exercise 4: Do you need to keep the filters which are embedded in the process document from files operator? Explain your answer.

- Run the process to view the document vector. Does the data set have the same attributes?

Load list of terms in a new process

- Save and Run the process.
- A document vector has been created for the four 'unseen' documents using the terms generated from the 15 labelled documents.

ExampleSet (4 examples, 2 special attributes, 15 regular attributes)

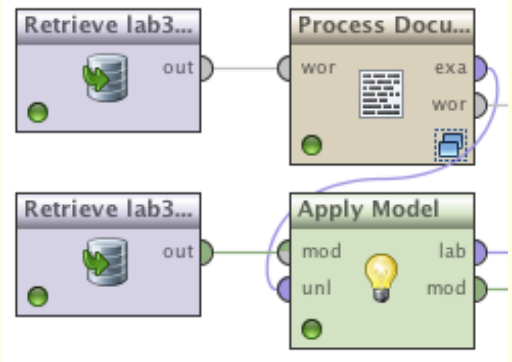
row no.	id	label	hse	court	right	cons
1	1	unknown	1	0	0	0
2	2	unknown	0	0	0	0
3	3	unknown	0	0	0	0
4	4	unknown	0	1	0	0

Note: It is important that the same tokenisers and filters are applied to these documents as were applied to the training documents. Occurrences will only be counted if the tokens here match the attributes in the attribute list.

Mining Unseen Data

- Next, apply the stored model to this dataset by:
  - Retrieve the Decision Tree Model (lab3-model-DT) from the repository (drag it on to screen)
  - Apply the model to the document vector (using Apply Model).

- The unseen texts should have been classified as follows:
  - U1: healthcare
  - U2: kenya
  - U3: healthcare
  - U4: crime



Exercise 5: How well did your classifier do? Can you explain the misclassifications?

Try a different classification algorithm

Exercise 6: Can you improve accuracy by using a different learner? Try one other learner, and report on the results. You can do this as follows:

Retrieve the k-NN model (lab3-model-kNN) from the repository and connect it to apply model instead.

To try other learners: return to lab3BasicProcess, replace the learner in the X-Validation block, and update the name on the store operator so you don't overwrite an existing stored model

- In the process 'mineUnseenData', retrieve the new model, connect it to apply model, and run the process again.

Exercise 7

Below are two processes, one to prepare data for training a classification model; the other is to prepare data to test that model.
Where there are differences between the two processes, explain if that will affect the results of testing the model.
If the Dictionarystemmer was used in the test process rather than the training process, would that be OK?

Process for training dataset

- Root
  - Process Document (texts=../training; purnebelow=3; vector_creation=TFIDF;)
    - StringTokeniser
    - EnglishStopWordFilter
    - DictionaryStemmer
    - PortersStemmer
  - Save word list

Process for test datasets

- Root
  - Process Document – create a word list from the word list already saved
  - Process Document (texts=../unlabelled; purnebelow=-1; vector_creation=occurrences)
    - StringTokeniser
    - StopWordFilterFile
    - LovinsStemmer