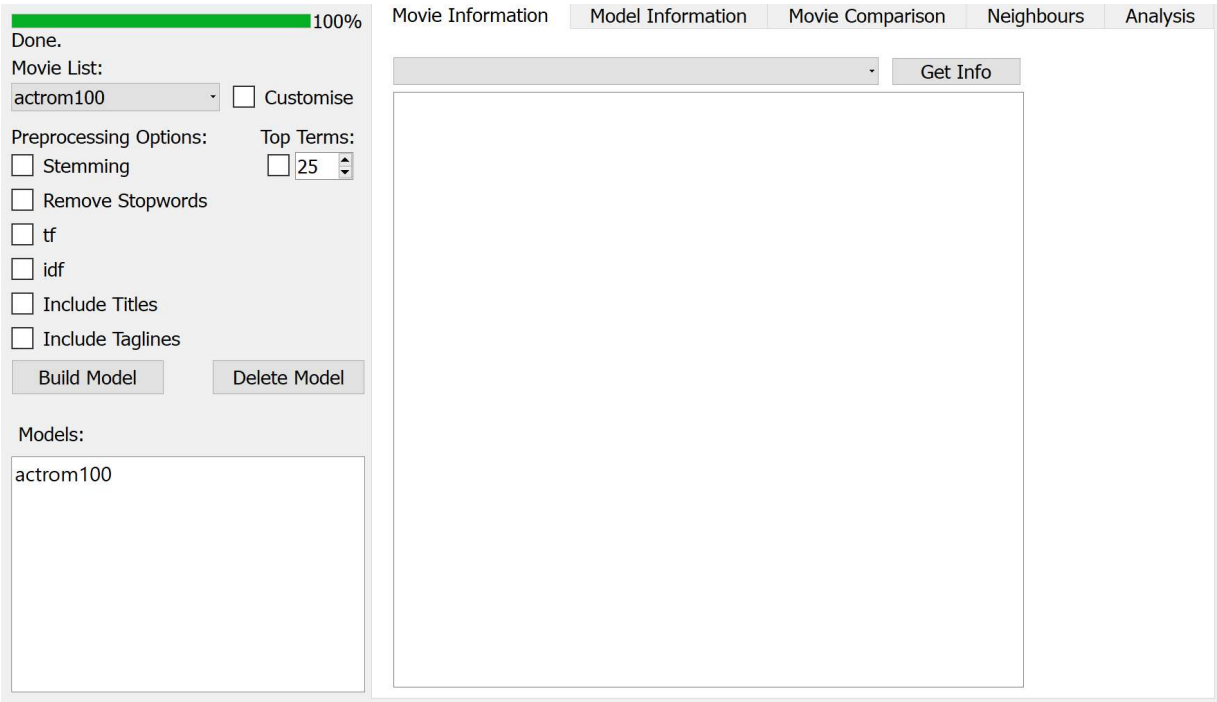
Part I - Analysing Movies

In this lab, you will be using various classifiers to predict the genre of a movie. We will be using the TMDB dataset of the 5000 most popular movies of all time (`themoviedb.org`), which is included in the file `movies.zip`, along with all the code you will need to run the classifiers. To install this package, see the instructions on the accompanying document.

Building a Model

Once you have installed everything correctly, we must start by building a model to train our classifiers on. This process will extract a short description of each movie from the dataset, apply linguistic preprocessing (such as tokenization and stemming) and build the term-document matrix used by the classifiers.

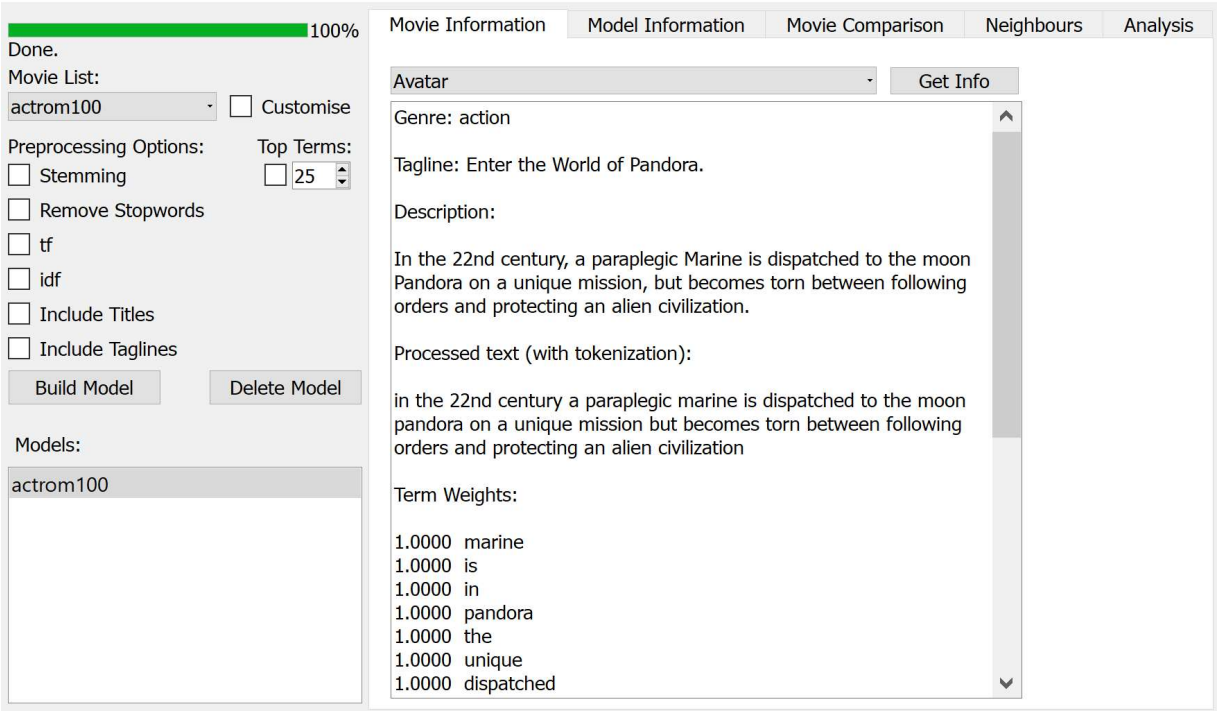
To begin with, we will build a simple model to classify Action and Romance movies. A list of 100 Action movies and 100 Romance movies are provided in the file `lists/actrom100.lst`. Build a model with these films by selecting the appropriate movie list from the drop-down menu, and clicking `Build Model`



This will save the model in the file `models/actrom100.mdl`, and make it accessible in the model list below. To load this model, simply select it from the list.

Viewing Movie Information

A drop-down menu containing each of the movies in this model should appear opposite. Click the `Get Info` button to print information about the selected movie. For instance, if we selected the movie Avatar, the output should look like this:



Linguistic Preprocessing

As you can see, the current model only tokenizes the description text, which represents the simplest unweighted bag-of-words approach. However, this tool allows you to use further linguistic preprocessing when building your model, such as stemming and stopword removal. Check the relevant boxes before pressing `Build Model` to include these features.

Here is an explanation of all the preprocessing options available:

- `tf`: Take into account the frequency of each term in a description
- `idf`: Weight each term by its inverse document frequency
- `Stopword Removal`: Remove common stopwords such as 'on' or 'the'
- `Stemming`: Reduce each word to its stem, e.g. 'following' ⇒ 'follow'
- `Titles`: Include the title of the movie in the set of terms representing the movie.
- `Taglines`: Include the tagline (e.g. "Welcome to Pandora") in the movie's text.

For example, after all preprocessing options are applied, this is what the movie avatar now looks like:

```
Genre: action

Tagline: Enter the World of Pandora.

Description:
```

```
In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a
unique mission, but becomes torn between following orders and protecting an alie
n civilization.

Processed text (with tokenization, tf, idf, stopwords, stemming):

22nd centuri parapleg marin dispatch moon pandora uniqu mission becom torn follo
w order protect alien civil

Term Weights:

4.6052  pandora
4.6052  dispatch
4.6052  parapleg
4.6052  22nd
4.1997  marin
4.1997  moon
4.1997  uniqu
3.6889  centuri
3.6889  alien
3.5066  civil
3.5066  torn
3.1011  order
2.9957  protect
2.9004  follow
2.6593  mission
2.1203  becom
```

The other options, `Customise` and `Top Terms`, will be used later in Part III.

Exercise 1

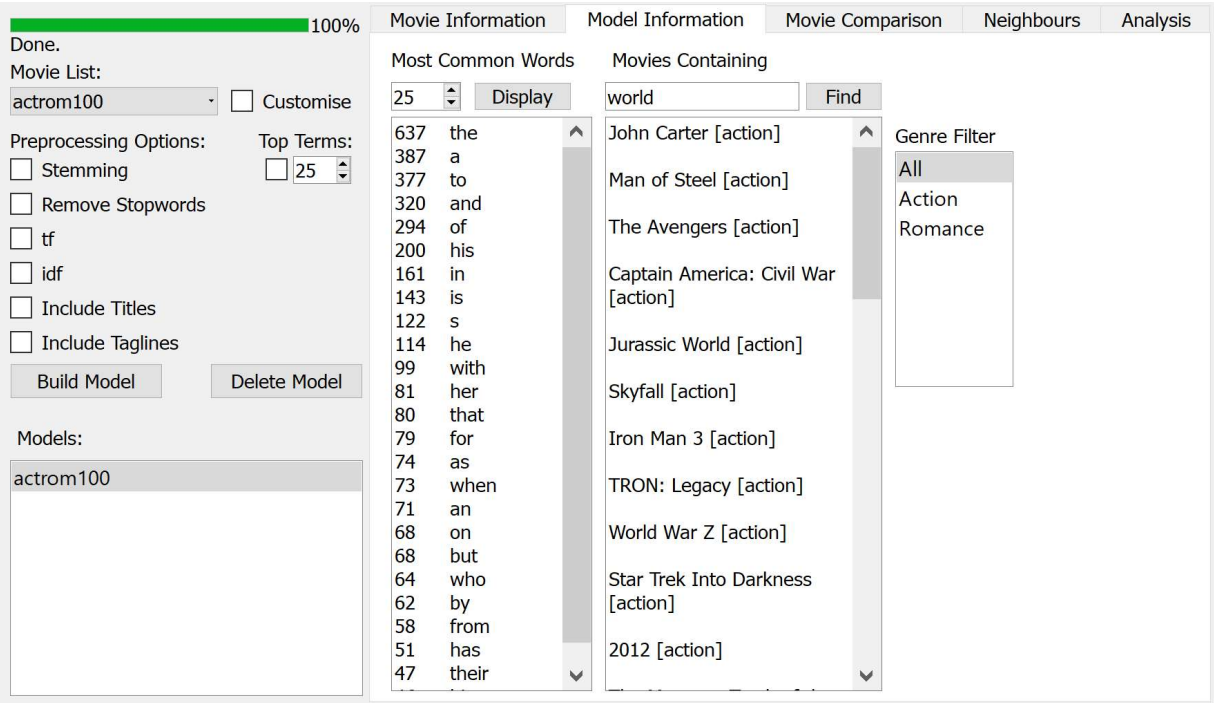
a) Build two models using the movie list `actrom100`. Both models should use the preprocessing options `tf`, `idf` and `Stopword Removal` – but include the `stemming` option in only one of the models.

b) Get information on The Lone Ranger with both models. Look at the weight of the word 'ranger' in each model. Does the weight of this term increase or decrease after applying stemming? Why?

c) Now try the same with Edge of Tomorrow, and note the weight of the term 'die'. Does this increase or decrease with stemming, and why?

Analysing Movies and Datasets

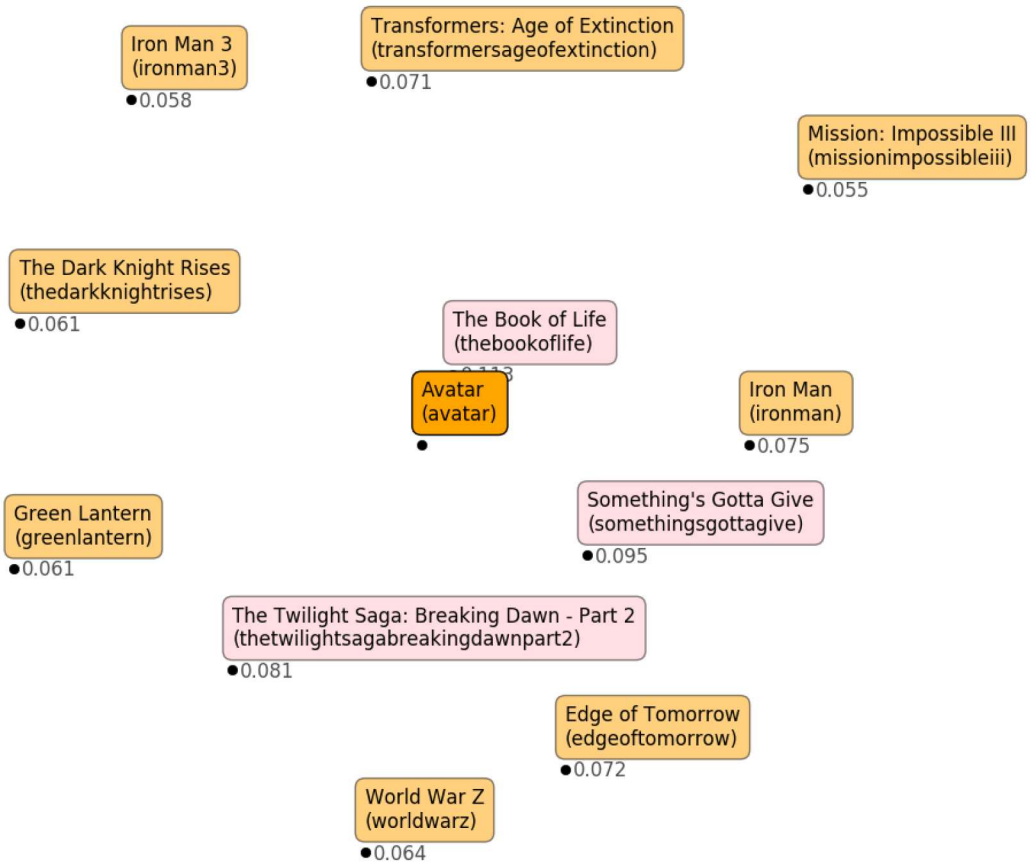
Once we have built a model, we can obtain additional information about the distribution of terms, or the position of movies in the term-vector space. For instance, the term-frequencies across the dataset can be viewed using the `Model Information` screen:



Which will display the 25 most common terms across the selected genres. You can also search for which movies contain a given term on this screen.

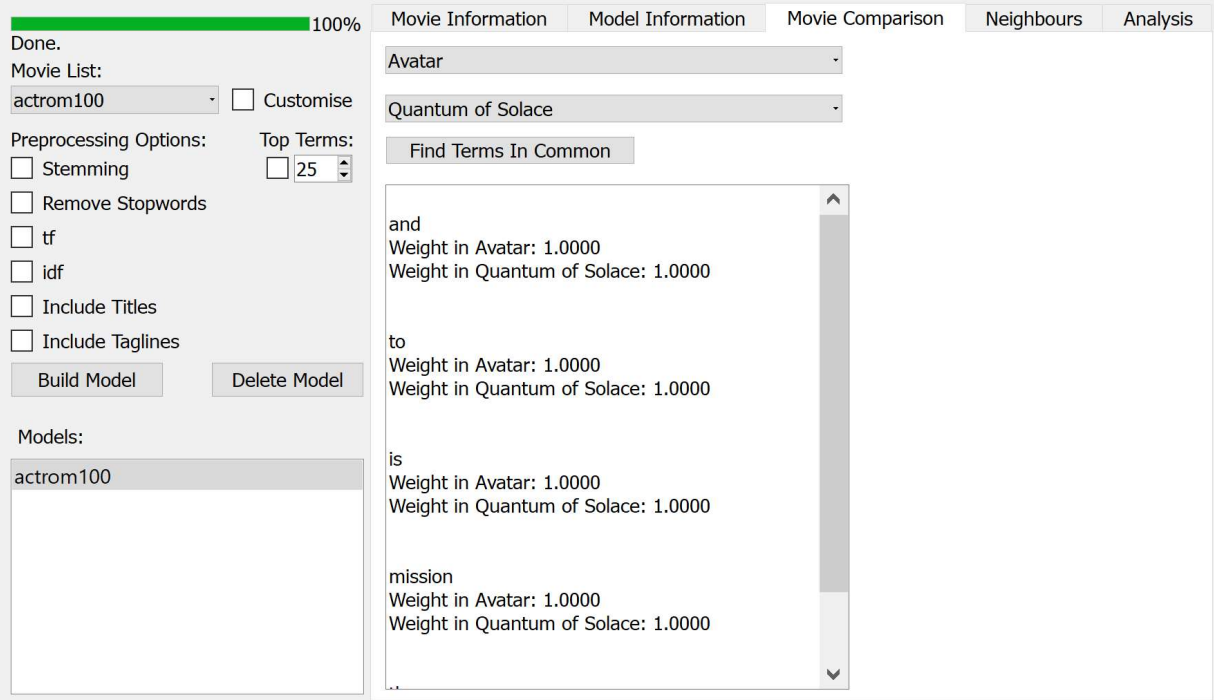
Nearest Neighbours

We can also get an idea of how the movies are distributed in the term-vector space using the `Neighbours` tab. Pressing `Find Neighbours` will generate both a textual and a graphical representation of the closest neighbours to the selected movie, like so:




```
0.1130  romance  The Book of Life
0.0947  romance  Something's Gotta Give
0.0809  romance  The Twilight Saga: Breaking Dawn - Part 2
0.0751  action   Iron Man
0.0724  action   Edge of Tomorrow
0.0707  action   Transformers: Age of Extinction
0.0644  action   World War Z
0.0613  action   The Dark Knight Rises
0.0605  action   Green Lantern
0.0577  action   Iron Man 3
0.0554  action   Mission: Impossible III
```

In this case, we can see that despite being primarily an action movie, one of Avatar's closest neighbours is Something's Gotta Give, a romance. To see why this might be the case, use the `Movie Comparison` tab, which tells us the words in common between two movie descriptions, and the weight of that term in each movie.



Exercise 2

The following questions concern the k Nearest Neighbours algorithm. Recall: in kNN, the class of a sample is determined by the most popular class within its k closest neighbours.
In this exercise, consider only odd-numbered values of k [Bonus question: why?]

a) Under a model with all preprocessing options, what is the minimal k-value we would need to choose so that Avatar is classified correctly by the kNN algorithm?

b) Find the reason for the high similarity between Avatar and Something's Gotta Give. Is the system correct in predicting such a high similarity score? If not, can you make changes to the model to fix this problem?

c) We define the k-range of a movie to be the set of all possible k-values, for which a movie would be classified correctly in a leave-one-out scenario (i.e. all the other movies are used as training data). In the following questions, use a model with all preprocessing options, and restrict k to a maximum of 17.

   i) Verify that the k-range of I am Legend is $1 \leq k \leq 7$.

   ii) What is the k-range of Avatar?

   iii) What is the k-range of Mamma Mia?

   iv) What is the k-range of The Book of Life?

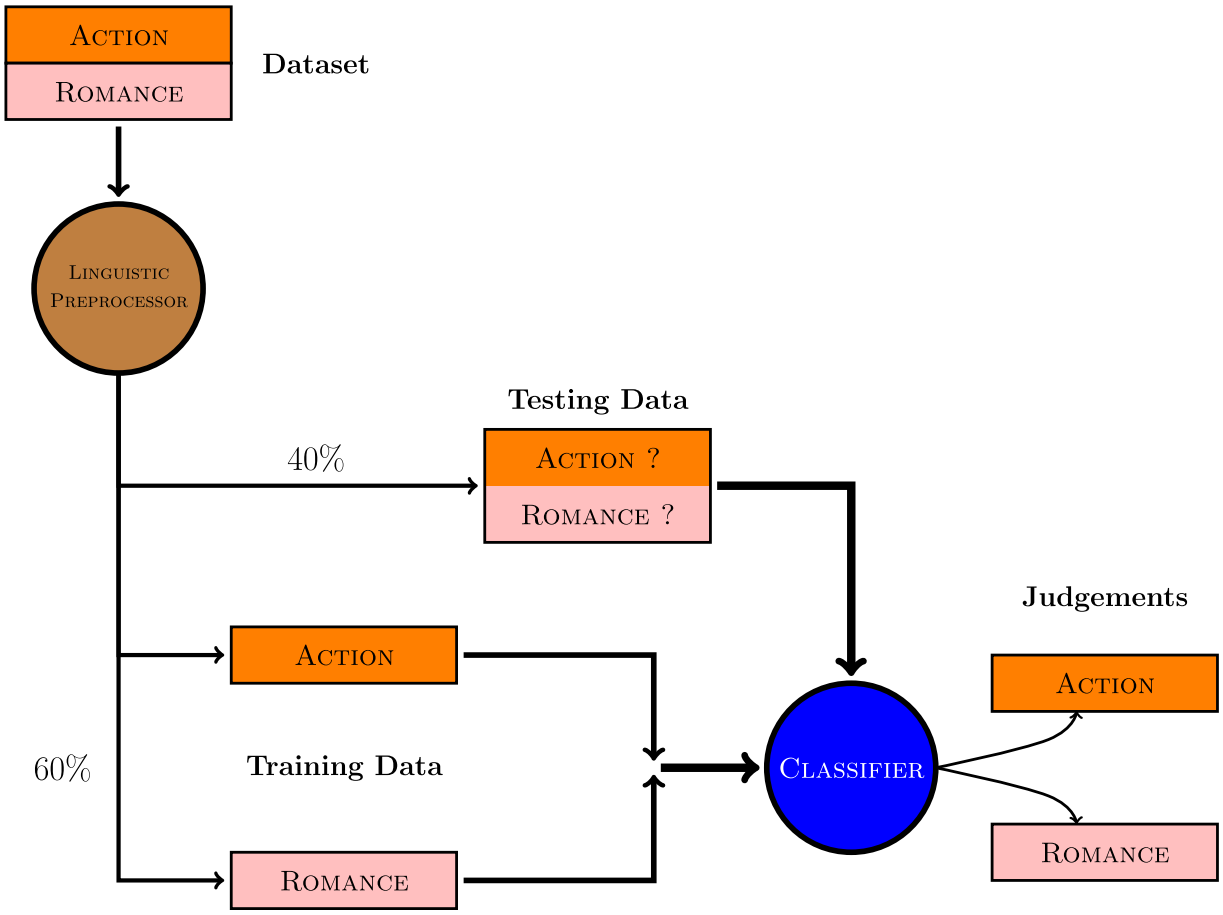
   v) What is the k-range of Superman Returns?

d) Are there any k-values which result in a correct classification for all of the movies in part c)? If so, what are they?

e) A standard technique for finding the optimal k-value is leave-one-out cross-validation. In this case, considering only the movies in part c), what k-values would be selected by this process?

Part II - Classifying Movies

In this section, we move on to the automatic classification of movies. You will be using two classifiers here: kNN and Rocchio. The architecture of these systems is shown below:

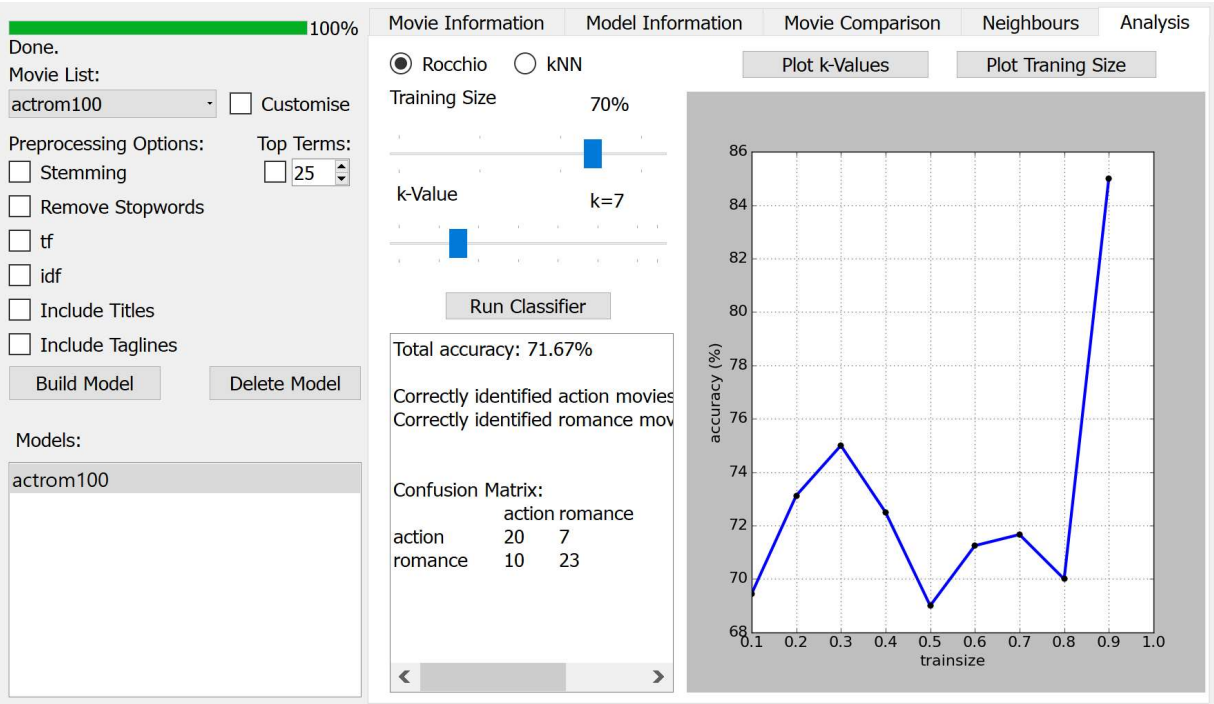


We can build a model using the linguistic preprocessor in the same manner as the previous section, and the included tool will use this to train and run a classifier on that model. To execute the classifier, it is enough to specify the algorithm needed, the size of the training data, and any parameters, e.g. the k in kNN. We start by taking a look at the Rocchio algorithm.

The Rocchio Classifier

The Rocchio classification method, also known as the Nearest Centroid Classifier, is one of the simplest text classification methods. The training phase consists of computing an 'average', or centroid, of the term vectors for each class. Classification simply consists of comparing a movie's term vector to each of the centroids, and outputting the most similar class.

We can train and run a Rocchio classifier using the `Analysis` screen.



The `Training Size` parameter indicates how many movies will be reserved for training. In this case, the first 70% of the movies will be used for training, and the remaining 30% will be used to test its accuracy. Generate the results with the `Run Classifier` button.
The confusion matrix shows the actual genres on the columns, and the predicted genres on the rows. For instance, 20 action movies were correctly predicted as action movies, and 10 romance movies were incorrectly predicted as action movies.

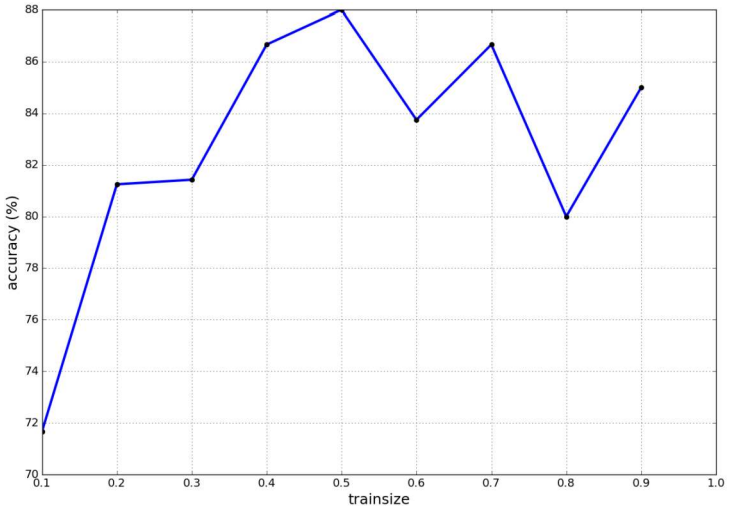
Exercise 3

a) Build two models for the classifier – a baseline model with no preprocessing options, and one with only stopword removal. Run a Rocchio classifier with a 60% training split on each of these models. How much improvement does the removal of stopwords give over the baseline model?

b) Now build two more models: both with `tf` and `idf` options, but only one with stopword removal. Run both as in part a). Is the improvement given by stopword removal smaller or larger than in part a)? Why do you think this is?

Plotting Performance

In addition to running a classifier, this tool can also test a batch of training sizes and plot the result, as shown here:



To do this, simply press the `Plot Training Size` button. It may take a short while to generate the plot.

Exercise 4

a) Plot the effect of the training size on a model with *no preprocessing options*. Observe the accuracy when the training split is increased to 90%. Which of these are possible explanations for this situation:

   i) The model is now underfitted.

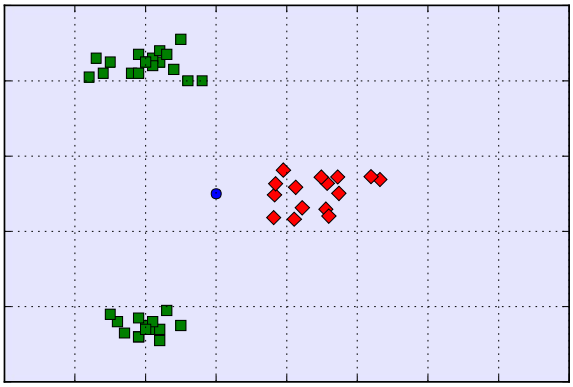
   ii) The model is now overfitted.

   iii) There is insufficient testing data.

   iv) The model was underperforming before, now it is better trained.

   v) The testing data is too easy.

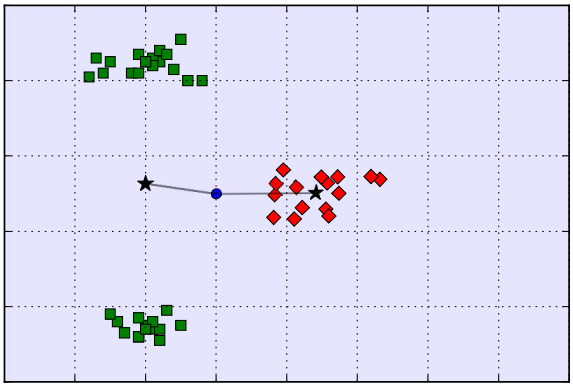
b) In the diagram below, we have two classes (green squares, red diamonds) of training data, and a new sample we wish to classify (blue circle).



i) Which class is the sample most similar to?

ii) What class would the Rocchio algorithm place it in, given the centroids (black stars) in the diagram below?

iii) If the answers to i) and ii) are different, why?



c) Build two models, with all preprocessing options, from the lists `mystactfam` and `mystactfammerged`, containing Mystery, Action and Family movies. In the first list, there are 3 distinct classes of 100 movies each.

In `mystactfammerged`, there are only two classes. The first contains 100 Mystery movies. The second contains 100 ActionFamily movies, comprised of 50 Action movies and 50 Family movies merged into a single class.

i) Plot the effect of the training size on both models, using the Rocchio algorithm.

ii) Which model is more sensitive to changes in the training data? Why do you think this is?

iii) Based on this, which model do you think will generalise better to unseen data?

iv) Would a Nearest Neighbours algorithm perform better in this scenario?

Nearest Neighbours

In the Rocchio algorithm, we attempt to build a general picture of the attributes of a class by creating a single, representative vector. But for multimodal data, where there may be multiple distinct clusters of vectors, Rocchio underperforms, as seen in the previous exercise. Because of this, we now explore an alternative: kNN. To use this, simply select `kNN` on the `Analysis` screen and use the slider to adjust the value of k.

Exercise 5

a) Plot the effect of the training size on an Action/Romance model with all preprocessing options, using both the Rocchio and kNN algorithms, with $k = 13$. Using this data, which algorithm is:

i) Less sensitive to the distribution of the training data?

ii) More accurate overall?

iii) More likely to generalise well to new data?

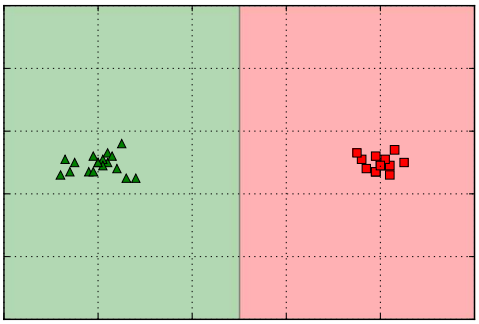
In general, which algorithm is more memory intensive? And which is faster to train and test?

b) Given a set of 200 samples, 100 from each class, for what values of `k` and `Training Size` is a kNN classifier exactly equivalent to a Rocchio classifier? Assume the training set is evenly split between classes.

Run each classifier on a model of your choice to verify this.

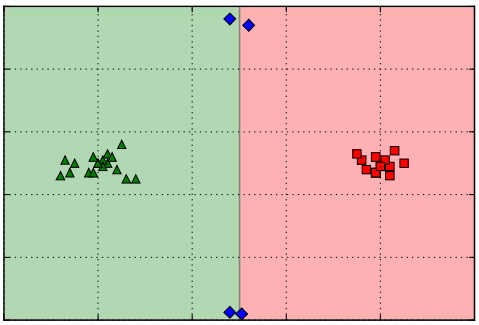
c) If a kNN classifier is trained on n% of the given samples, and tested on the remainder, what value of n gives the longest running time for this procedure?

d) Suppose we are given a set of samples, and we reserve half of these for training a kNN system, which creates the (approximate) decision boundary, as shown:



Does this represent a good classifier? Will it generalise well to future data?

e) Suppose we want to verify the performance of this classifier by testing it on the other half of the data. The first four samples we see are as follows:



In this scenario, we have two pairs of samples which are very similar to each other, but are given opposite classifications. In general, this would not be a desired result. To remedy this problem, we could either:

i) Change to a Rocchio classifier, which handles single clusters of data better.

ii) Use a high value for k.

iii) Use a cross-validation method to measure the error instead.

iv) Use a very low value for k (such as $k = 1$).

v) Ignore these outliers, since it will handle all other cases very effectively, and any changes will decrease the overall performance.

Select the solution you think would be most effective, and justify your answer.

f) Suppose we have 20 action movies and 10 romance movies in a training set, with $k = 30$. What would be the testing accuracy with:

i) A test set of 90 action movies and 10 romance movies?

ii) A test set of 10 action movies and 90 romance movies?

Is the accuracy of a classifier always a good way of measuring its quality?

g) In addition to the training size, we can also use the plotting tool to measure the effect of k. Fix the training size using the slider, and then use the `Plot k-Values` button.

i) Plot the effect of k on the same model used in part a), up to a maximum of $k = 19$.

ii) Select the most optimal k-value, and now plot the effect of the training size with this k-value.

iii) Run a single classifier with the best-performing k and training size from parts i) and ii). Is this the highest-performing classifier possible with this data? Why or why not?

iv) What method would you use to automate the process of finding the best possible pair of values?

h) In a two-class scenario, we can eliminate ambiguous classifications by selecting odd-numbered values for k, to ensure there are no ties. What values could we choose in a three-class model? How about an n-class model?