Hands-on Sentiment Analysis Tutorial

If you run into issues, try to figure it out with your neighbor first but if you cannot then put your hand up and I will try to come and help you.

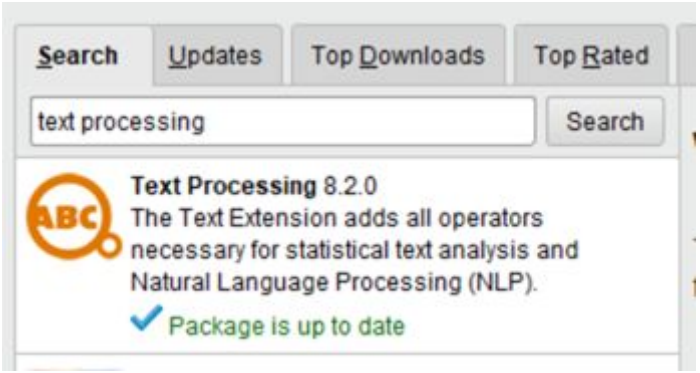
Step 1: Getting all you need to get started

- Go to www.annedirkson.nl
- Find the Data button at the top and go to Sentiment Analysis Tutorial in the drop down menu
- Click on the Download button next to 'IMBD_movie_reviews_sample' to download the data. Save it somewhere on your computer.

If you have not yet installed RapidMiner Studio, use the provided link to do so. This will give you a 30 day free trial.

Step 2: Setting up RapidMiner Studio

- Open RapidMiner and go to Extensions.
- Click Marketplace
- Type "Text Processing" in the search bar
- Click on Text Processing and install



- Restart RapidMiner when asked

Great so now you have data and a tool to analyze it! You are ready to start.

Conducting sentiment analysis on movie reviews

In this tutorial you will practice with:

1. Preprocessing text data
2. Looking at the most frequent words in the data
3. Making a word cloud of these words
4. Training an algorithm to do sentiment analysis of the movie reviews
5. Looking at the output of the sentiment analysis

Sentiment analysis is the task of categorizing pieces of text as positive or negative. As with any research, we should start by looking at the data and writing down what we expect the outcome to be. The first step will be to load the data into the program.

Step 1: Loading the Data

- Start up the RapidMiner Studio
- Select Blank Project

In the bottom left corner, your screen is a box called 'Operators'. This is where you will find all the tools you need. (Operators will always be in blue from now on)

- Use the search bar in this box to find the 'Read Excel' Operator
- Drag and drop Read Excel into the middle of the white screen. Attach it on the left to the little *inp* button on the left of the screen.
- Now click on the Read Excel
- Press import configuration wizard
- Select the Excel data file that you downloaded from my website by browsing through your file manager
- Press Next
- Press Finish
- Connect the Read Excel on the right side to the res button on the right of the white screen

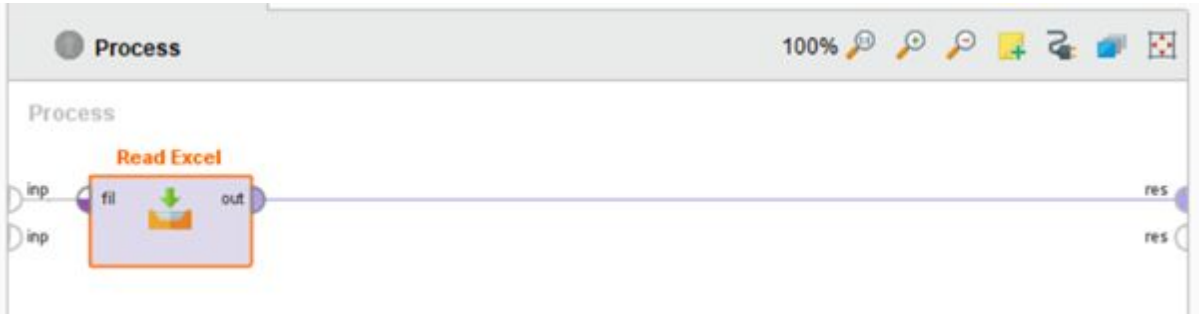


Figure 1: Reading an Excel File

- Press Run button (the blue triangle at the top) to see what the output will be
- It should be a table with every sentence as a separate row.

Step 2: Writing down what you expect

- Have a look at some of the data by scrolling through the table
- What do you think the most frequent words will be in this dataset?
- Which words do you expect to be able to separate positive from negative tweets? Which words do you use to distinguish the two?
- Click on Design at the top to go back to making the pipeline.

So now we have loaded the data and we have taken a quick look at it. Before we can start doing sentiment analysis or figuring out which words are the most frequent in the data, we will need to clean or preprocess the texts. This is usually the most time consuming part of the data analysis.

Step 3: Preprocessing the Data

Before we can do anything with our text data, we need to tell RapidMiner to treat the data as text. To do so, we will use the Nominal to Text operator (again use the search bar in the bottom right to find it). Drag it to the right of the Read Excel operator and connect the *out* dot from Read Excel to the left *exa* dot of Nominal to Text. This is how we send data between operators. The next step is to add the Process Document from Data operator; find it and add it to the right of Nominal to Text. Connect the right *exa* from Nominal to Text to the left *exa* of Process

Documents from Data, and connect the *wor* to *res* on the top right. It should look something like this:

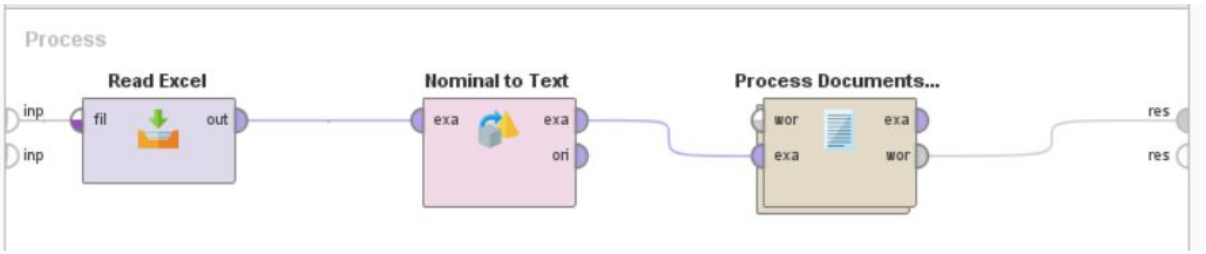


Figure 2: Preprocessing overview

- Now click on Process Documents from Data, and on the right, uncheck 'create word vector' and 'add meta information'.
- Also set the prune method to 'absolute', and fill in 2 for 'prune below absolute' and 99999 for 'prune above absolute'. This way we ignore words which only occur in 1 document.
- Now double click on Process Documents from Data, this is where we'll do the various preprocessing steps!

The first step in preprocessing text is tokenization. Tokenization is the splitting of the sentence into words, which are also called tokens.

- To do this, find the Tokenize operator and drag it into the screen.
- Connect *doc* on the left to the left *doc* of Tokenize.

The second step will be to lowercase everything. That way the computer will not see Dog and dog as two seperate words but as the same word.

- Find the Transform Cases operator, add it, and connect the right *doc* of Tokenize to the left *doc* of Transform Cases.

The next step in our preprocessing will be the removal of stop words. Stop words are words that do not contain a lot of information but are used in the English language very frequently like 'the', 'and', 'of', 'is' etc. We will remove those to help our algorithm separate positive from negative reviews, because we expect that these words will not be helpful in distinguishing between the two. Both positive and negative reviews will contain a lot of stop words.

- Find the Filter Stopwords (English) operator and add it and connect it to Transform Cases.

Now we will generate n-grams. N-grams are sequences of n items from the piece of text. For example, the unigrams (uni= one) of the sentence "the cat bit the dog" are the, cat, bit, the and dog. The bigrams (bi = two) are "the cat", "cat bit", "bit the", and "the dog".

- Find and add the Generate n-Grams (Terms) operator and connect it to Filter Stopwords.
- Click on Generate n-Grams and set 'max length' to 2 in the settings on the right.
- The final step is to connect Generate n-Grams to the *doc* on the right, so we can output the results. Your screen should look something like this:

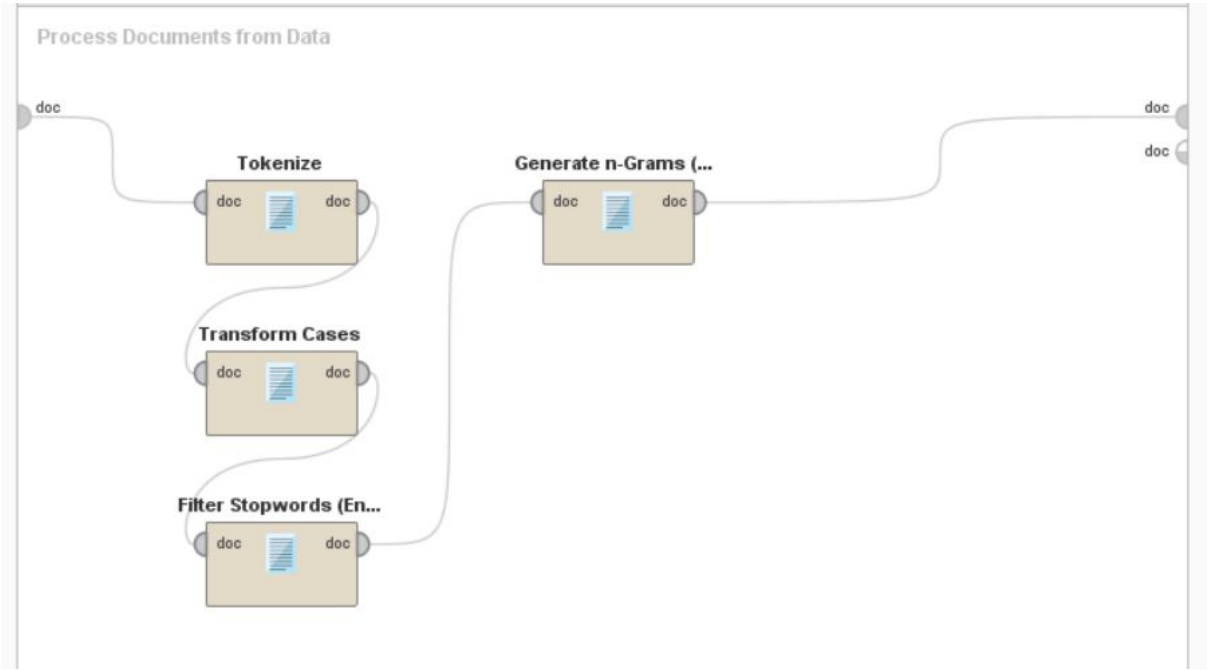


Figure 2: Preprocessing pipeline

Step 4: Looking at the most frequent words and creating a Word Cloud

First let's take a look at the most frequent words manually.

- Run your pipeline by pressing the Run button (the blue ▸ at the top). This might take a little bit of time.
- In the table that you get find the column total and click on it. This sorts the words by frequency and gives you the most frequent words.

Is this what you expected to see? Why? Why not?

Alright, let's create a word cloud of the top 20 n-grams. We will need to first sort the words from more frequent to least frequent and then select only the top 20:

- Add the Wordlist to Data operator to the word output of the Process Documents operator. See in the picture below how to connect the operators.
- Add the Sort operator to the Wordlist to Data
- Set the Sort operator to decreasing
- Set the attribute name to total
- Add the Filter Example Range after Sort Operator
- Set the Filter Example Range to first = 1 and last = 20, this way we only include the top 20 most occurring words
- Connect it to the results button on the right hand side

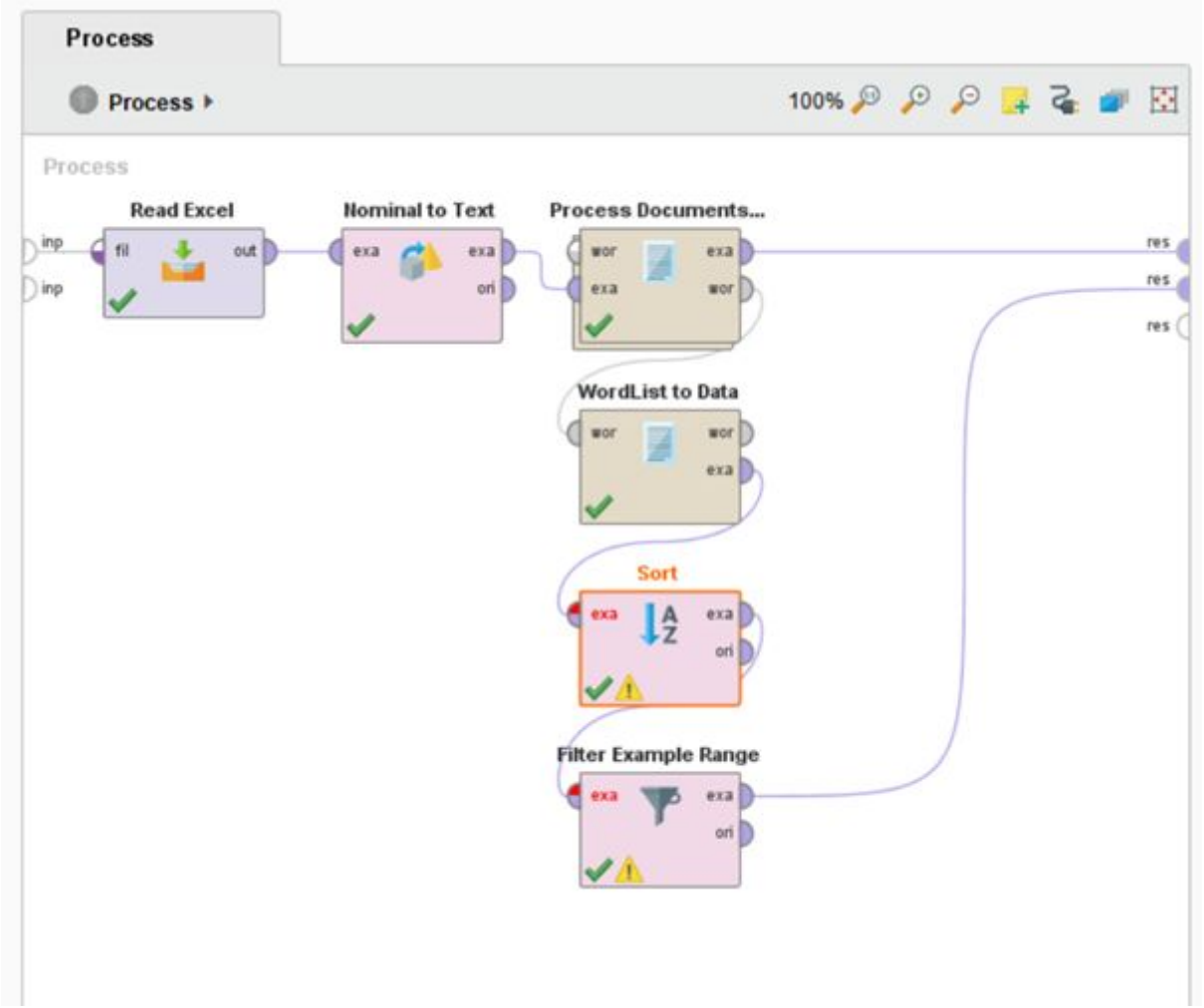


Figure 3: Making the word cloud

- Run your pipeline to go to the results

Now within the Results tabs we will make a wordcloud:

- In the Data tab (that you start in) make sure to go ExampleSet (Filter Example Range) instead
- Then go to Visualizations
- For plot type select Word cloud (the one the bottom right)
- For the value select word
- For the weight select total

Hmmm this does not look too great. There are many words here with only a few letters. Let's go back to the pipeline and add a filter so these words are filtered out.

- Go back to the pipeline with the Design button
- Click on Process Documents from Data
- Add a Filter Tokens (by Length) operator before Generate n-grams
- Set the min chars (minimum characters or letters of a word) to 3 and the maximum to 999
- Run again and make a new wordcloud
- ***Does the word cloud match your expectations now?***

Figure 4: Settings for a word cloud and an example

Extra challenge*: If you want, you can change the last parameter in the Filter Example Range to add more or less words for instance the top 10 or top 100*

Step 5: Doing sentiment analysis on your data

Before we can start doing sentiment analysis we should remove a few steps from the pipeline:

- Delete the Wordlist to Data, Sort and Filter Example Range operators

So remember that sentiment is how positive or negative something is. We are now going to use the IMDB movie reviews that are labelled for sentiment (tagged with positive or negative) to make a sentiment classifier that can automatically categorize movie reviews into positive and negative categories. To do this we are going to use machine learning.

First of all we need to tell the computer what we want to predict.

- The Set Role operator will let us tell the computer which values to use as labels of the sentences. Add a Set Role operator between Read Excel and Nominal to Text. Click on the Set Role operator to select it. We want it to use the label column as the label so for Attribute name we set label. For target role we choose label (because this is the function of this attribute).

Now we will change our preprocessing pipeline.

- Double click on Process Documents from Data
- We do not want to filter out words of less than 2 letters anymore because we want to allow the algorithm to select for itself if these words are important or not so let's remove the Filter Tokens (by Length) operator
- To speed up the training of the algorithm, we will also remove the Generate N-grams operator.

It should now look like this:

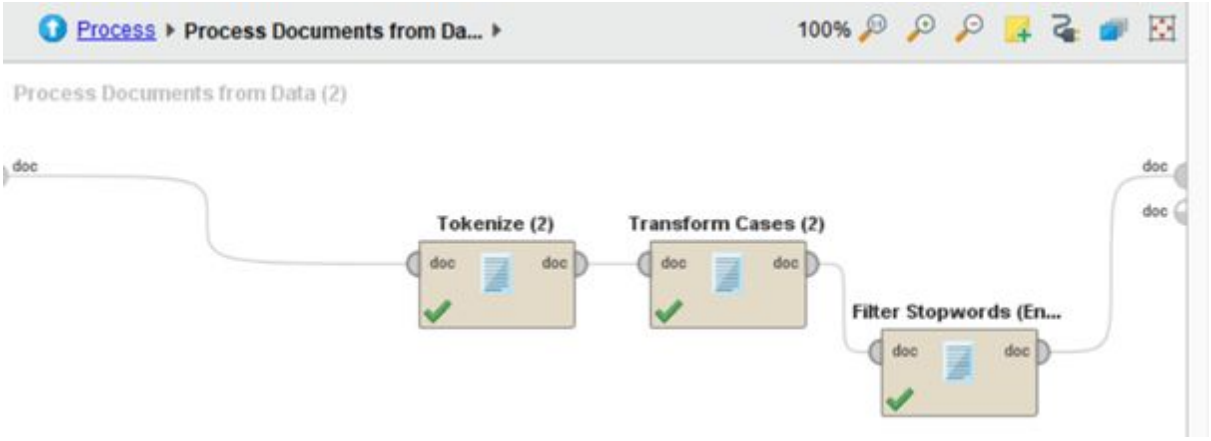


Figure 5: How the preprocessing pipeline should look

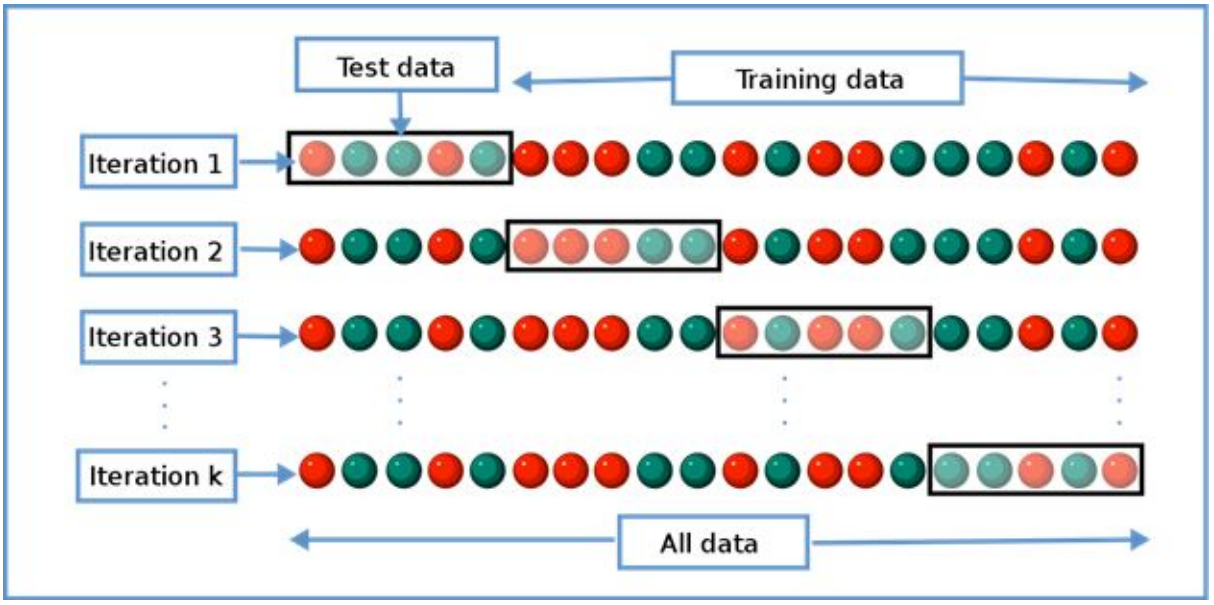
- Go back to the overall pipeline screen.
- Now just click on the Process Documents from Data to adjust its settings. Tick create word vectors, tick keep text and tick add meta information.

Unfortunately, there is not enough time to explain what all these features mean but they allow the algorithm to better make use of the data.

Time to start training an algorithm.

When training an algorithm, data is always split into a training set and a testing set. First you train a model with the training set by giving it examples for each of the categories. Then you want to test how well your model performs and for this you use the testing set. You give the model only the texts (and not the labels) and let it make predictions. How close the predictions are to the true labels shows how good the model is.

Cross validation is when you do this multiple times. You split the data into training and testing in different ways. Each run or iteration you train a model and test it. The average of the testing of the different runs give you a better idea of the performance of the model than just one run.



- We will add a Cross Validation operator to the pipeline
- Set the number of folds to 5. This means you split the data into training and testing sets 5x.
- Now double click on the Cross Validation operator to design the training process.

You can see that this screen is split into two. The left side is for training and the right side is for testing.

First we need to select the type of model that we will use to predict the sentiment. We will use an algorithm called SVM (Support Vector Machine).

- On the **left** side, add the Support Vector Machine operator.
- You do not need to change any of the parameters.
- On the **right** side, add an Apply Model operator. Make sure to connect all the operators correctly like in the picture below. This operator will actually run the model on the test data to see how it performs.
- After the Apply Model operator, also add a Performance operator on the **right** side to measure the performance of the model. You do not need to change any of the parameters.

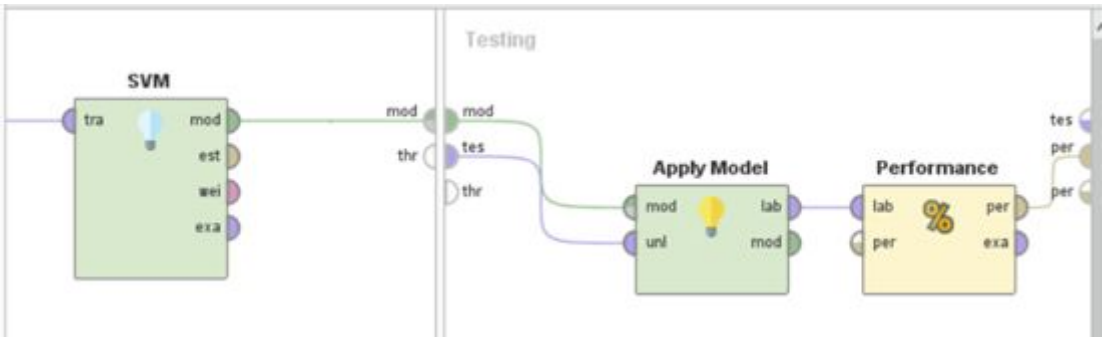


Figure 6: How to connect the steps within the Cross Validation step

- OK. Done. Go back the overview screen.
- Finally, connect the output of the Cross Validation operator to the right side of the screen (see figure below for how they should be connected).
- ***Press Run.***

This is what the whole pipeline should look like:

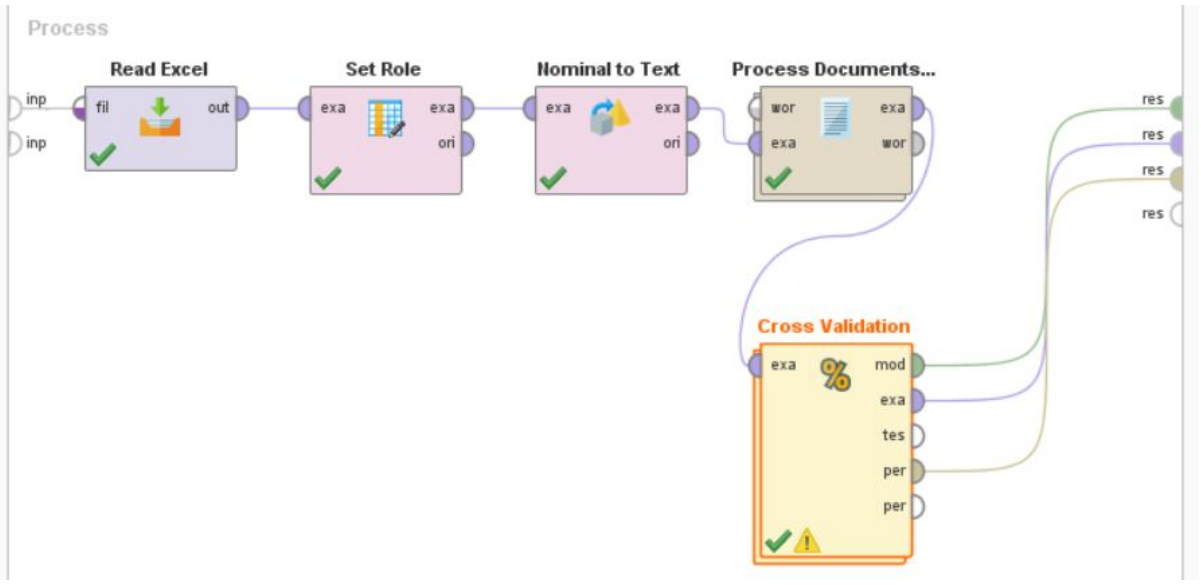


Figure 7: The whole pipeline for training a sentiment classifier

Well done! You have trained your first machine learning algorithm on text. Now let's look what has happened

Looking at the output

First of all, let's look at how well the model did during testing.

- Click on Performance Vector in the top tabs on the screen.
- In bold in the left you should see the accuracy. This is how much it got correct. How did your model do?

Now let's look at which words our model used to predict positive and negative reviews.

- Go to the Kernel Model (SVM) tab on the top
- Click on Weight Table on the left
- Now you can see the weights for each word
- Press Weight to see the words with the highest negative and highest positive weight

A high negative weight means a word is important for predicting the negative category (when a sentence is negative). A high positive weight means that a word is important for predicting the positive category (when a sentence is positive).

If the weight of a word is 0 it has no influence at all on the prediction.

What are the 10 most important words for predicting positive sentences? Was this what you expected?

What are the 10 most important words for predicting negative sentences? Was this what you expected?

Great job! You completed the tutorial. If you have extra time left, you can do some of the optional challenges on the next page.

OPTIONAL CHALLENGES

Option 1: Have a look at the most frequent words in only the positive tweets. Do the same for the negative tweets.

The most frequent words in the positive tweets are not the same as the words that were important for deciding that it was positive. If a word is used very frequently in both the positive and negative tweets, it does not help in deciding which to pick. However, it is still a very frequent word and can tell us something about what the positive or negative tweets are about.

Hint: you will need the Filter Example operator.

Option 2: Document clustering by similarity

Complete this tutorial on document clustering based on similarity:

https://academy.rapidminer.com/learn/course/text-and-web-mining-with-rapidminer/text-and-web-mining/comparison-classification-and-clustering?page=2