Orange3 Text Mining Documentation

Release

Biolab

Aug 03, 2017

Contents

CHAPTER 1

Widgets

1.1 Corpus

Load a corpus of text documents, (optionally) tagged with categories.

1.1.1 Signals

Inputs:

- (None)

Outputs:

- **Corpus**

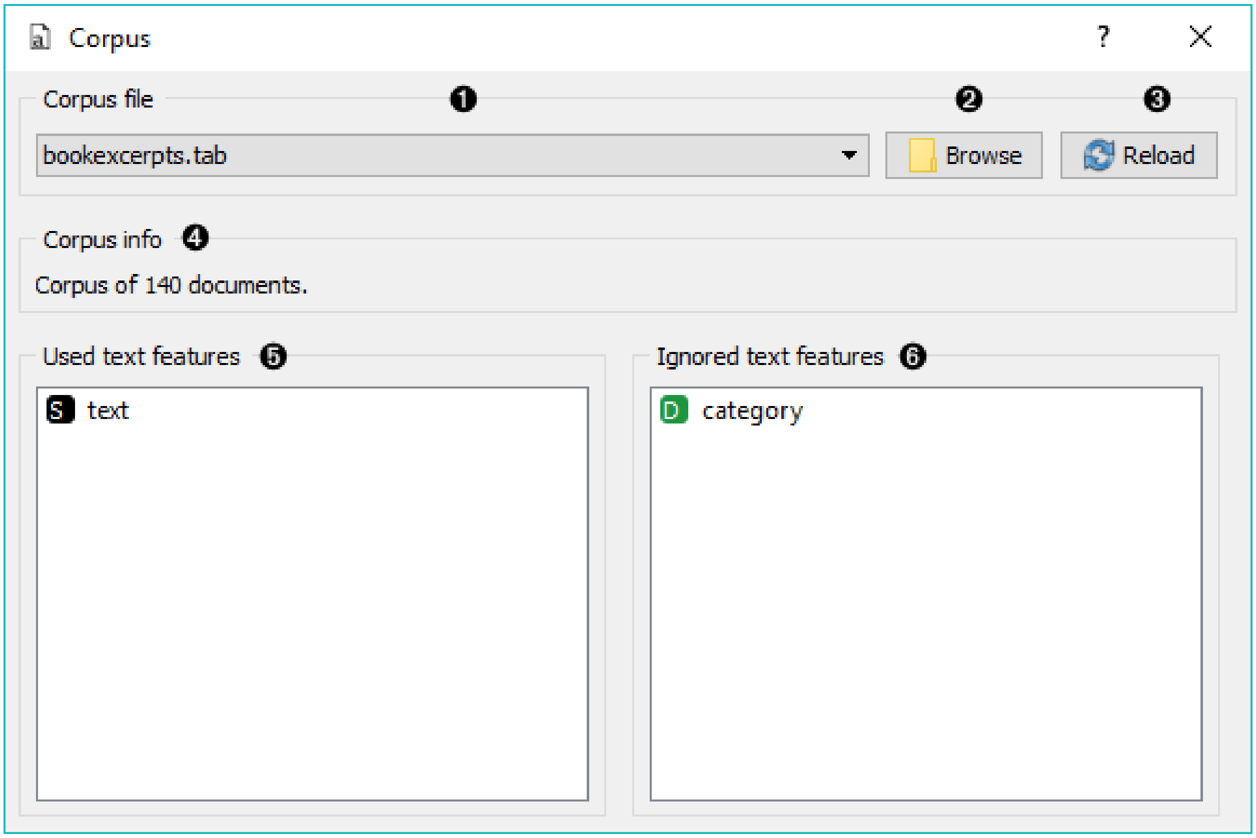
A *Corpus* instance.

1.1.2 Description

Corpus widget reads text corpora from files and sends a corpus instance to its output channel. History of the most recently opened files is maintained in the widget. The widget also includes a directory with sample corpora that come pre-installed with the add-on.

The widget reads data from Excel (**.xlsx**), comma-separated (**.csv**) and native tab-delimited (**.tab**) files.

1. Browse through previously opened data files, or load any of the sample ones.
2. Browse for a data file.



bookexcerpts.tab

Corpus of 140 documents.

S text

D category

3. Reloads currently selected data file.
4. Information on the loaded data set.
5. Features that will be used in text analysis.
6. Features that won't be used in text analysis and serve as labels or class.

You can drag and drop features between the two boxes and also change the order in which they appear.

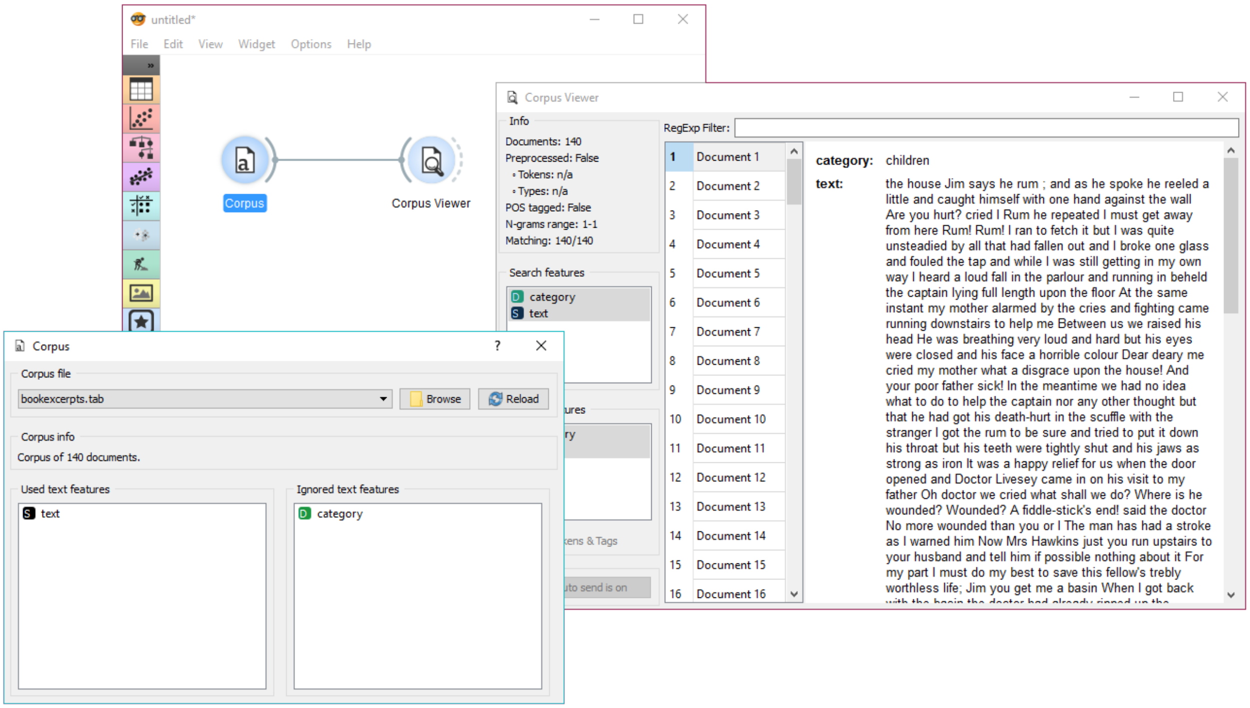
1.1.3 Example

The first example shows a very simple use of **Corpus** widget. Place **Corpus** onto canvas and connect it to *Corpus Viewer*. We've used *booxexcerpts.tab* data set, which comes with the add-on, and inspected it in **Corpus Viewer**.

The second example demonstrates how to quickly visualize your corpus with *Word Cloud*. We could connect **Word Cloud** directly to **Corpus**, but instead we decided to apply some preprocessing with *Preprocess Text*. We are again working with *book-excerpts.tab*. We've put all text to lowercase, tokenized (split) the text to words only, filtered out English stopwords and selected a 100 most frequent tokens.

1.2 Import Documents

Import text documents from folders.



Corpus

Corpus file

bookexcerpts.tab

Browse Reload

Corpus info

Corpus of 140 documents.

Used text features

S text

Ignored text features

D category

Corpus Viewer

Info

Documents: 140
Preprocessed: False
∘ Tokens: n/a
∘ Types: n/a
POS tagged: False
N-grams range: 1-1
Matching: 140/140

Search features

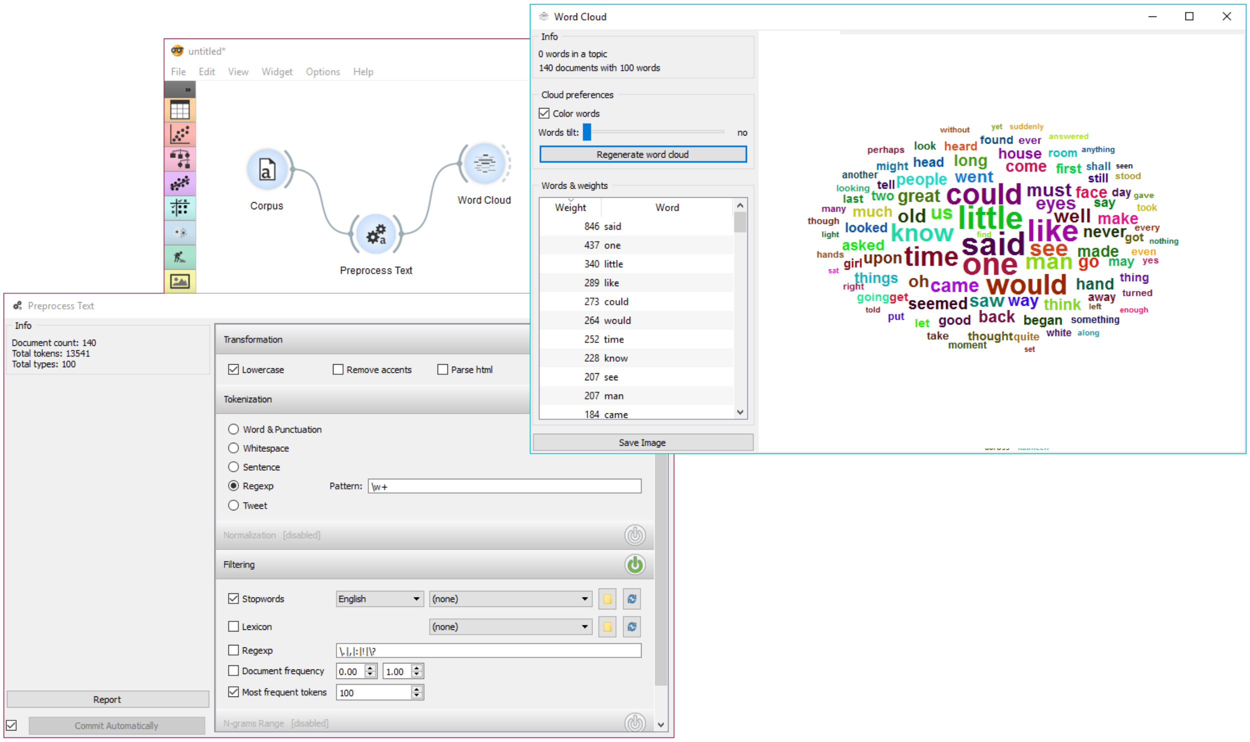
D category
S text

RegExp Filter:

1	Document 1
2	Document 2
3	Document 3
4	Document 4
5	Document 5
6	Document 6
7	Document 7
8	Document 8
9	Document 9
10	Document 10
11	Document 11
12	Document 12
13	Document 13
14	Document 14
15	Document 15
16	Document 16

category: children

text: the house Jim says he rum ; and as he spoke he reeled a little and caught himself with one hand against the wall Are you hurt? cried I Rum he repeated I must get away from here Rum! Rum! I ran to fetch it but I was quite unsteadied by all that had fallen out and I broke one glass and fouled the tap and while I was still getting in my own way I heard a loud fall in the parlour and running in beheld the captain lying full length upon the floor At the same instant my mother alarmed by the cries and fighting came running downstairs to help me Between us we raised his head He was breathing very loud and hard but his eyes were closed and his face a horrible colour Dear deary me cried my mother what a disgrace upon the house! And your poor father sick! In the meantime we had no idea what to do to help the captain nor any other thought but that he had got his death-hurt in the scuffle with the stranger I got the rum to be sure and tried to put it down his throat but his teeth were tightly shut and his jaws as strong as iron It was a happy relief for us when the door opened and Doctor Livesey came in on his visit to my father Oh doctor we cried what shall we do? Where is he wounded? Wounded? A fiddle-stick's end! said the doctor No more wounded than you or I The man has had a stroke as I warned him Now Mrs Hawkins just you run upstairs to your husband and tell him if possible nothing about it For my part I must do my best to save this fellow's trebly worthless life; Jim you get me a basin When I got back


untitled*
File Edit View Widget Options Help
Corpus
Preprocess Text
Word Cloud
said one little like could would time know see man came upon must eyes face well make never great old us much looked asked girl things oh hand thing seemed saw way think good back began let put take thought quite white moment people went come house room first shall still long head heard found ever might perhaps look tell two last another many though light hands right going get told say day gave took every got nothing even may yes made go away left turned enough something along without yet suddenly answered anything seen stood looking sat set find

Preprocess Text

Info

Document count: 140
Total tokens: 13541
Total types: 100

Transformation

☑ Lowercase ☐ Remove accents ☐ Parse html

Tokenization

○ Word & Punctuation
○ Whitespace
○ Sentence
● Regexp Pattern: \w+
○ Tweet

Normalization [disabled]

Filtering

☑ Stopwords English (none)
☐ Lexicon (none)
☐ Regexp \.|,|:|!|\?
☐ Document frequency 0.00 1.00
☑ Most frequent tokens 100

N-grams Range [disabled]

Report

☑ Commit Automatically

Word Cloud

Info

0 words in a topic
140 documents with 100 words

Cloud preferences

☑ Color words
Words tilt: no

Regenerate word cloud

Words & weights

Weight	Word
846	said
437	one
340	little
289	like
273	could
264	would
252	time
228	know
207	see
207	man
184	came

Save Image

1.2.1 Signals

Inputs:

- (None)

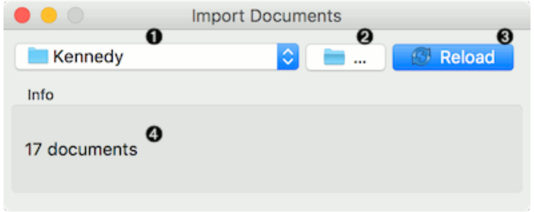
Outputs:

- **Corpus**

  A *Corpus* instance.

1.2.2 Description

Import Documents widget retrieves text files from folders and creates a corpus. The widget reads .txt, .docx, .odt, .pdf and .xml files. If a folder contains subfolders, they will be used as class labels.



1. Folder being loaded.
2. Load folder from a local machine.
3. Reload the data.
4. Number of documents retrieved.

If the widget cannot read the file for some reason, the file will be skipped. Files that were successfully retrieved will still be on the output.

1.2.3 Example

To retrieve the data, select the folder icon on the right side of the widget. Select the folder you wish to turn into coprus. Once the loading is finished, you will see how many documents the widget retrieved. To inspect them, connect the widget to *Corpus Viewer*. We've used a set of Kennedy's speeches in a plain text format.

Now let us try it with subfolders. We have placed Kennedy's speeches in two folders - pre-1962 and post-1962. If I load the parent folder, these two subfolders will be used as class labels. Check the output of the widget in a **Data Table**.

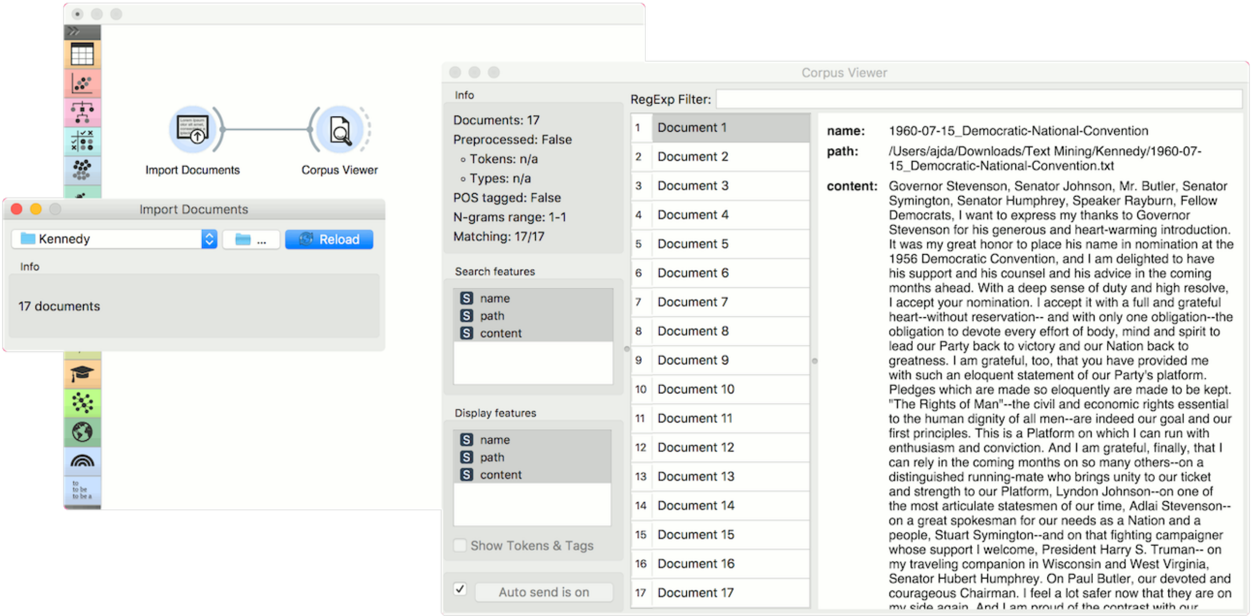
1.3 NY Times

Loads data from the New York Times' Article Search API.

1.3.1 Signals

Inputs:

- (None)



Import Documents

Kennedy ... Reload

Info

17 documents

Corpus Viewer

Info

Documents: 17
Preprocessed: False
∘ Tokens: n/a
∘ Types: n/a
POS tagged: False
N-grams range: 1-1
Matching: 17/17

Search features

- S name
- S path
- S content

Display features

- S name
- S path
- S content

□ Show Tokens & Tags

☑ Auto send is on

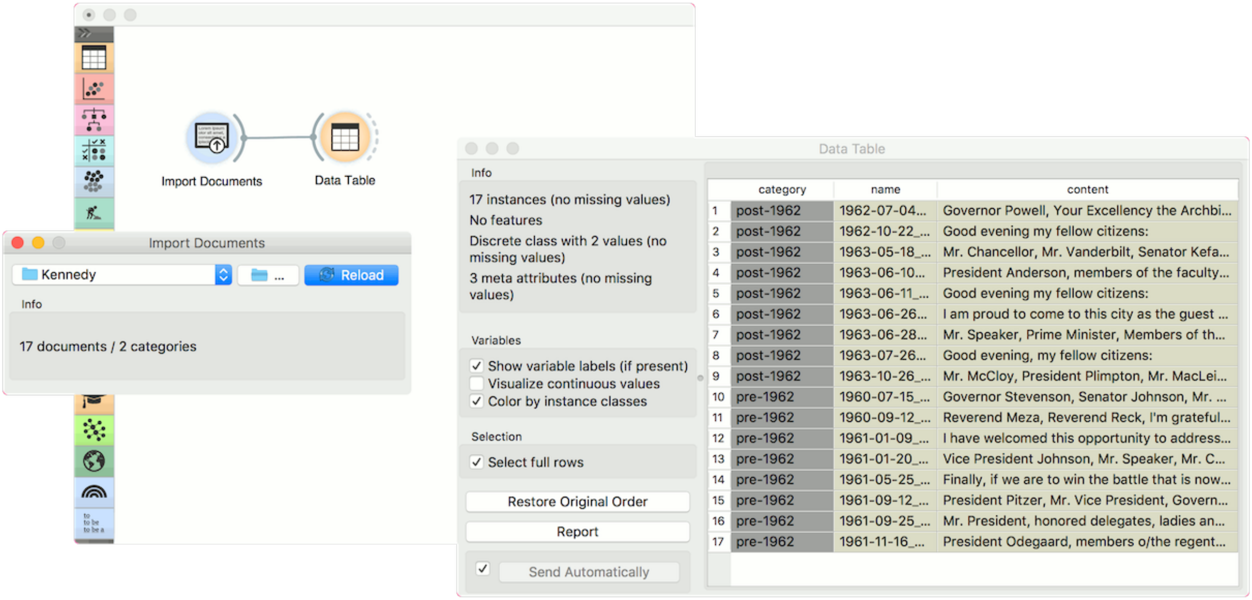
RegExp Filter:

1	Document 1
2	Document 2
3	Document 3
4	Document 4
5	Document 5
6	Document 6
7	Document 7
8	Document 8
9	Document 9
10	Document 10
11	Document 11
12	Document 12
13	Document 13
14	Document 14
15	Document 15
16	Document 16
17	Document 17

name: 1960-07-15_Democratic-National-Convention

path: /Users/ajda/Downloads/Text Mining/Kennedy/1960-07-15_Democratic-National-Convention.txt

content: Governor Stevenson, Senator Johnson, Mr. Butler, Senator Symington, Senator Humphrey, Speaker Rayburn, Fellow Democrats, I want to express my thanks to Governor Stevenson for his generous and heart-warming introduction. It was my great honor to place his name in nomination at the 1956 Democratic Convention, and I am delighted to have his support and his counsel and his advice in the coming months ahead. With a deep sense of duty and high resolve, I accept your nomination. I accept it with a full and grateful heart--without reservation-- and with only one obligation--the obligation to devote every effort of body, mind and spirit to lead our Party back to victory and our Nation back to greatness. I am grateful, too, that you have provided me with such an eloquent statement of our Party's platform. Pledges which are made so eloquently are made to be kept. "The Rights of Man"--the civil and economic rights essential to the human dignity of all men--are indeed our goal and our first principles. This is a Platform on which I can run with enthusiasm and conviction. And I am grateful, finally, that I can rely in the coming months on so many others--on a distinguished running-mate who brings unity to our ticket and strength to our Platform, Lyndon Johnson--on one of the most articulate statesmen of our time, Adlai Stevenson--on a great spokesman for our needs as a Nation and a people, Stuart Symington--and on that fighting campaigner whose support I welcome, President Harry S. Truman-- on my traveling companion in Wisconsin and West Virginia, Senator Hubert Humphrey. On Paul Butler, our devoted and courageous Chairman. I feel a lot safer now that they are on my side again. And I am proud of the contrast with our



Import Documents

Kennedy ... Reload

Info

17 documents / 2 categories

Data Table

Info

17 instances (no missing values)
No features
Discrete class with 2 values (no missing values)
3 meta attributes (no missing values)

Variables

☑ Show variable labels (if present)
□ Visualize continuous values
☑ Color by instance classes

Selection

☑ Select full rows

Restore Original Order

Report

☑ Send Automatically

	category	name	content
1	post-1962	1962-07-04...	Governor Powell, Your Excellency the Archbi...
2	post-1962	1962-10-22_...	Good evening my fellow citizens:
3	post-1962	1963-05-18_...	Mr. Chancellor, Mr. Vanderbilt, Senator Kefa...
4	post-1962	1963-06-10...	President Anderson, members of the faculty...
5	post-1962	1963-06-11_...	Good evening my fellow citizens:
6	post-1962	1963-06-26...	I am proud to come to this city as the guest ...
7	post-1962	1963-06-28...	Mr. Speaker, Prime Minister, Members of th...
8	post-1962	1963-07-26...	Good evening, my fellow citizens:
9	post-1962	1963-10-26_...	Mr. McCloy, President Plimpton, Mr. MacLei...
10	pre-1962	1960-07-15_...	Governor Stevenson, Senator Johnson, Mr. ...
11	pre-1962	1960-09-12_...	Reverend Meza, Reverend Reck, I'm grateful...
12	pre-1962	1961-01-09_...	I have welcomed this opportunity to address...
13	pre-1962	1961-01-20_...	Vice President Johnson, Mr. Speaker, Mr. C...
14	pre-1962	1961-05-25_...	Finally, if we are to win the battle that is now...
15	pre-1962	1961-09-12_...	President Pitzer, Mr. Vice President, Govern...
16	pre-1962	1961-09-25_...	Mr. President, honored delegates, ladies an...
17	pre-1962	1961-11-16_...	President Odegaard, members o/the regent...

Outputs:

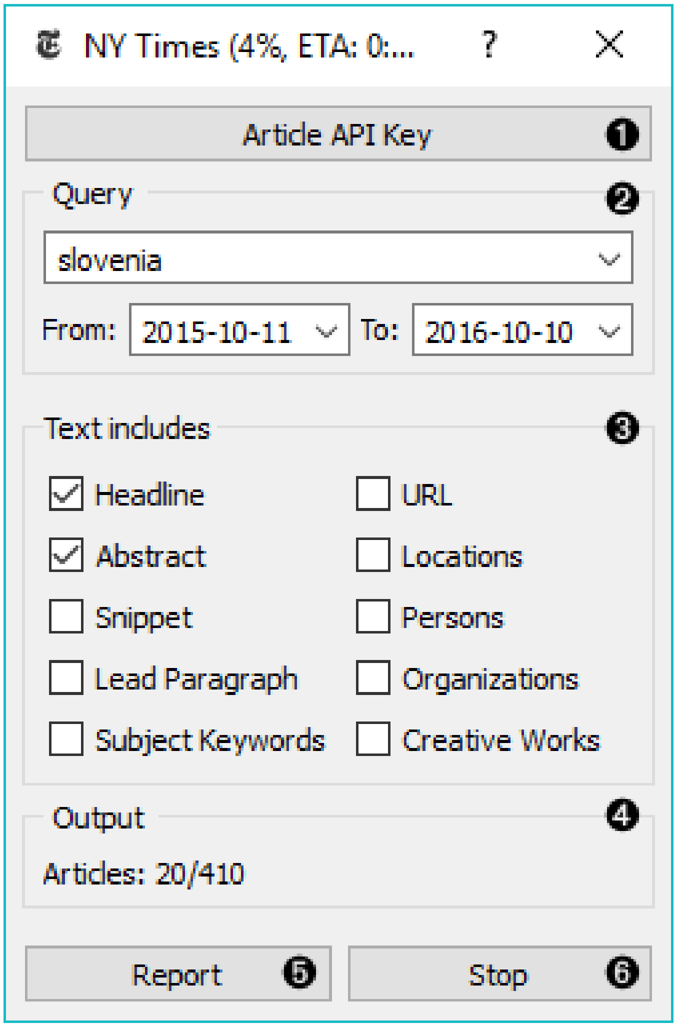
- **Corpus**

  A *Corpus* instance.

1.3.2 Description

NYTimes widget loads data from New York Times' Article Search API. You can query NYTimes articles from September 18, 1851 to today, but the API limit is set to allow retrieving only a 1000 documents per query. Define which features to use for text mining, *Headline* and *Abstract* being selected by default.

To use the widget, you must enter your own API key.



NY Times (4%, ETA: 0:...

Query: slovenia

From: 2015-10-11 To: 2016-10-10

Text includes

☑ Headline ☐ URL

☑ Abstract ☐ Locations

☐ Snippet ☐ Persons

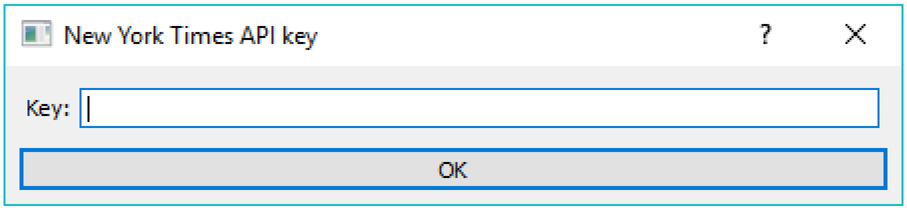
☐ Lead Paragraph ☐ Organizations

☐ Subject Keywords ☐ Creative Works

Output

Articles: 20/410

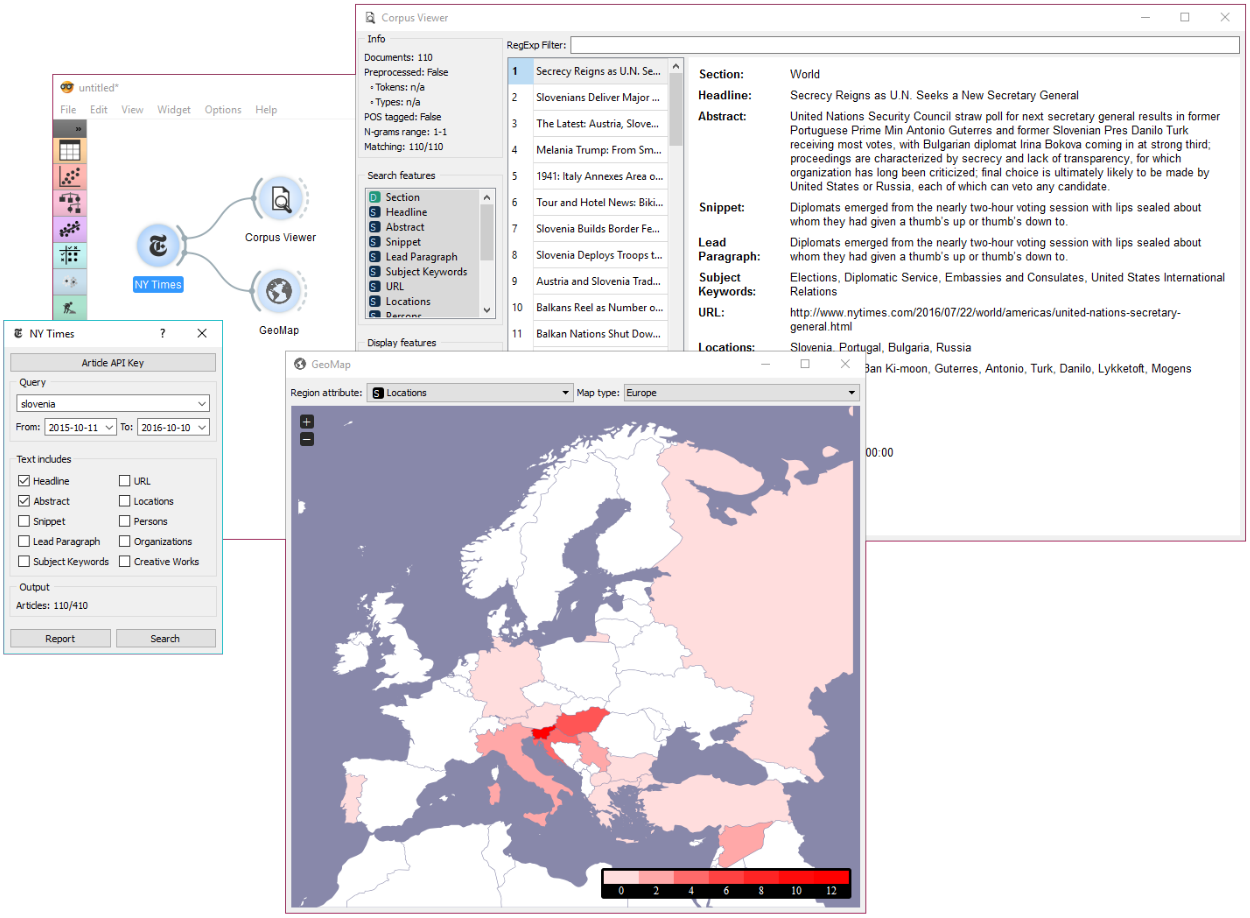
1. To begin your query, insert NY Times' Article Search API key. The key is securely saved in your system keyring service (like Credential Vault, Keychain, KWallet, etc.) and won't be deleted when clearing widget settings.
2. **Set query parameters:**
   - *Query*
   - Query time frame. The widget allows querying articles from September 18, 1851 onwards. Default is set to 1 year back from the current date.



3. Define which features to include as text features.
4. Information on the output.
5. Produce report.
6. Run or stop the query.

1.3.3 Example

NYTimes is a data retrieving widget, similar to *Twitter* and *Wikipedia*. As it can retrieve geolocations, that is geographical locations the article mentions, it is great in combination with *GeoMap* widget.



NY Times

Article API Key

Query: slovenia

From: 2015-10-11 To: 2016-10-10

Text includes

☑ Headline
☑ Abstract
□ Snippet
□ Lead Paragraph
□ Subject Keywords
□ URL
□ Locations
□ Persons
□ Organizations
□ Creative Works

Output

Articles: 110/410

Corpus Viewer

Info

Documents: 110
Preprocessed: False
◦ Tokens: n/a
◦ Types: n/a
POS tagged: False
N-grams range: 1-1
Matching: 110/110

Search features

D Section
S Headline
S Abstract
S Snippet
S Lead Paragraph
S Subject Keywords
S URL
S Locations
S Persons

Display features

RegExp Filter:

1 Secrecy Reigns as U.N. Se...
2 Slovenians Deliver Major ...
3 The Latest: Austria, Slove...
4 Melania Trump: From Sm...
5 1941: Italy Annexes Area o...
6 Tour and Hotel News: Biki...
7 Slovenia Builds Border Fe...
8 Slovenia Deploys Troops t...
9 Austria and Slovenia Trad...
10 Balkans Reel as Number o...
11 Balkan Nations Shut Dow...

Section: World

Headline: Secrecy Reigns as U.N. Seeks a New Secretary General

Abstract: United Nations Security Council straw poll for next secretary general results in former Portuguese Prime Min Antonio Guterres and former Slovenian Pres Danilo Turk receiving most votes, with Bulgarian diplomat Irina Bokova coming in at strong third; proceedings are characterized by secrecy and lack of transparency, for which organization has long been criticized; final choice is ultimately likely to be made by United States or Russia, each of which can veto any candidate.

Snippet: Diplomats emerged from the nearly two-hour voting session with lips sealed about whom they had given a thumb's up or thumb's down to.

Lead Paragraph: Diplomats emerged from the nearly two-hour voting session with lips sealed about whom they had given a thumb's up or thumb's down to.

Subject Keywords: Elections, Diplomatic Service, Embassies and Consulates, United States International Relations

URL: http://www.nytimes.com/2016/07/22/world/americas/united-nations-secretary-general.html

Locations: Slovenia, Portugal, Bulgaria, Russia

Ban Ki-moon, Guterres, Antonio, Turk, Danilo, Lykketoft, Mogens

00:00

GeoMap

Region attribute: S Locations Map type: Europe

First, let's query **NYTimes** for all articles on Slovenia. We can retrieve the articles found and view the results in *Corpus Viewer*. The widget displays all the retrieved features, but includes on selected features as text mining features.

Now, let's inspect the distribution of geolocations from the articles mentioning Slovenia. We can do this with *GeoMap*. Unsuprisignly, Croatia and Hungary appear the most often in articles on Slovenia (discounting Slovenia itself), with the rest of Europe being mentioned very often as well.

1.4 The Guardian

Fetching data from The Guardian Open Platform.

1.4.1 Signals

Inputs:

- (None)

Outputs:

- **Corpus**

  A *Corpus* instance.

1.4.2 Description

No description for this widget yet.

1.4.3 Examples

No examples for this widget yet.

1.5 Twitter

Fetching data from The Twitter Search API.

1.5.1 Signals

Inputs:

- (None)

Outputs:

- **Corpus**

  A *Corpus* instance.

1.5.2 Description

Twitter widget enables querying tweets through Twitter API. You can query by content, author or both and accummulate results should you wish to create a larger data set. The widget only supports REST API and allows queries for up to two weeks back.

1. To begin your queries, insert Twitter key and secret. They are securely saved in your system keyring service (like Credential Vault, Keychain, KWallet, etc.) and won't be deleted when clearing widget settings. You must first create a Twitter app to get API keys.
2. **Set query parameters:**
   - *Query word list*: list desired queries, one per line. Queries are automatically joined by OR.
   - *Search by*: specify whether you want to search by content, author or both. If searching by author, you must enter proper Twitter handle (without @) in the query list.
   - *Allow retweets*: if 'Allow retweets' is checked, retweeted tweets will also appear on the output. This might duplicate some results.
   - *Date*: set the query time frame. Twitter only allows retrieving tweets from up to two weeks back.
   - *Language*: set the language of retrieved tweets. Any will retrieve tweets in any language.
   - *Max tweets*: set the top limit of retrieved tweets. If box is not ticked, no upper bound will be set - widget will retrieve all available tweets.
   - *Accumulate results*: if 'Accumulate results' is ticked, widget will append new queries to the previous ones. Enter new queries, run *Search* and new results will be appended to the previous ones.
3. Define which features to include as text features.
4. Information on the number of tweets on the output.
5. Produce report.
6. Run query.

1.5.3 Examples

First, let's try a simple query. We will search for tweets containing either 'data mining' or 'machine learning' in the content and allow retweets. We will further limit our search to only a 100 tweets in English.

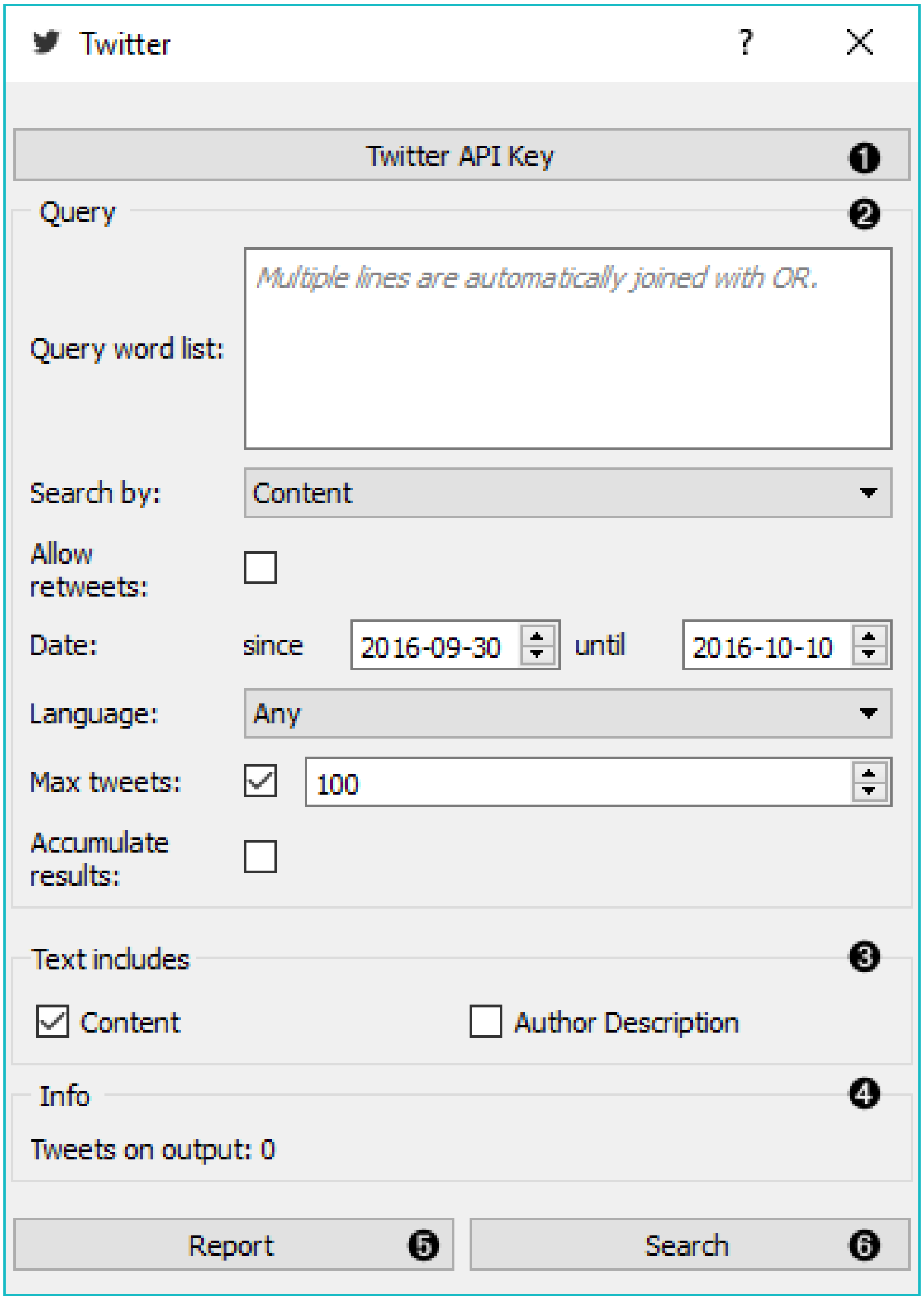
First, we're checking the output in *Corpus Viewer* to get the initial idea about our results. Then we're preprocessing the tweets with lowercase, url removal, tweet tokenizer and removal of stopword and punctuation. The best way to see the results is with *Word Cloud*. This will display the most popular words in field of data mining and machine learning in the past two weeks.

Our next example is a bit more complex. We're querying tweets from Hillary Clinton and Donald Trump from the presidential campaign 2016.

Then we've used *Preprocess Text* to get suitable tokens on our output. We've connected **Preprocess Text** to *Bag of Words* in order to create a table with words as features and their counts as values. A quick check in **Word Cloud** gives us an idea about the results.

Now we would like to predict the author of the tweet. With **Select Columns** we're setting 'Author' as our target variable. Then we connect **Select Columns** to **Test & Score**. We'll be using **Logistic Regression** as our learner, which we also connect to **Test & Score**.

We can observe the results of our author predictions directly in the widget. AUC score is quite ok. Seems like we can to some extent predict who is the author of the tweet based on the tweet content.



Query word list: Multiple lines are automatically joined with OR.

Search by: Content

Allow retweets: □

Date: since 2016-09-30 until 2016-10-10

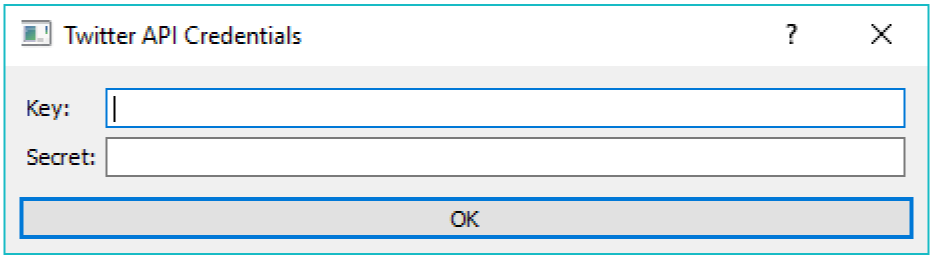
Language: Any

Max tweets: ☑ 100

Accumulate results: □

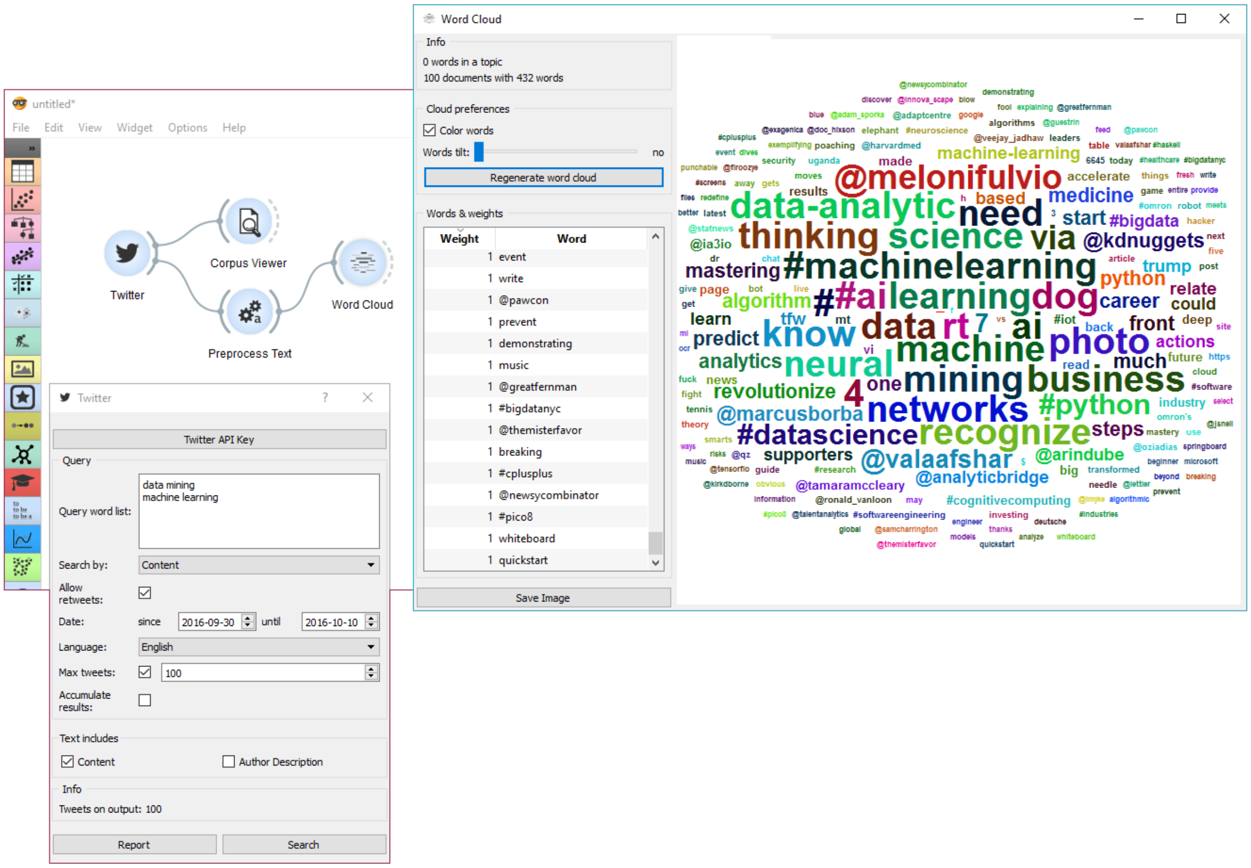
☑ Content □ Author Description

Tweets on output: 0



Key:

Secret:



Twitter

Twitter API Key

Query

Query word list: data mining
machine learning

Search by: Content

Allow retweets: ☑

Date: since 2016-09-30 until 2016-10-10

Language: English

Max tweets: ☑ 100

Accumulate results: □

Text includes

☑ Content □ Author Description

Info

Tweets on output: 100

Report | Search

Word Cloud

Info

0 words in a topic
100 documents with 432 words

Cloud preferences

☑ Color words

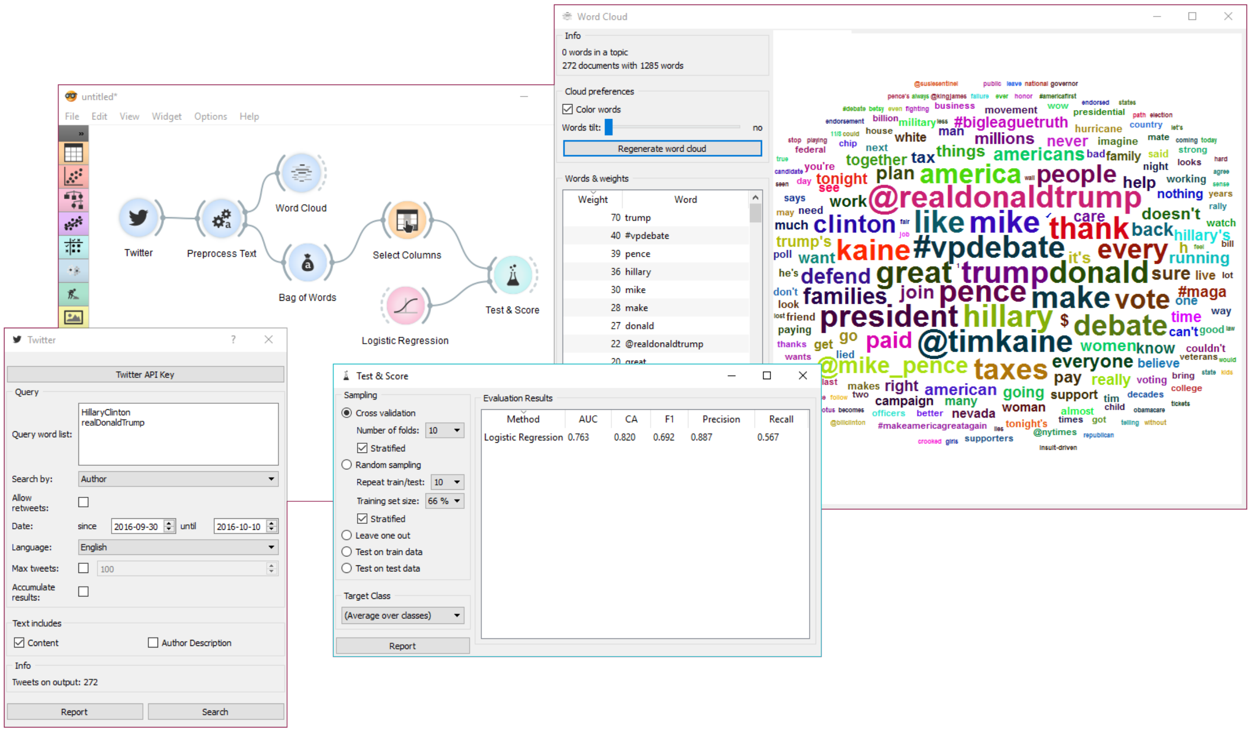
Words tilt: no

Regenerate word cloud

Words & weights

Weight	Word
1	event
1	write
1	@pawcon
1	prevent
1	demonstrating
1	music
1	@greatfernman
1	#bigdatanyc
1	@themisterfavor
1	breaking
1	#cplusplus
1	@newsycombinator
1	#pico8
1	whiteboard
1	quickstart

Save Image



Twitter

Twitter API Key

Query

Query word list: HillaryClinton
realDonaldTrump

Search by: Author

Allow retweets: □

Date: since 2016-09-30 until 2016-10-10

Language: English

Max tweets: □ 100

Accumulate results: □

Text includes

☑ Content □ Author Description

Info

Tweets on output: 272

Report | Search

Word Cloud

Info

0 words in a topic
272 documents with 1285 words

Cloud preferences

☑ Color words

Words tilt: no

Regenerate word cloud

Words & weights

Weight	Word
70	trump
40	#vpdebate
39	pence
36	hillary
30	mike
28	make
27	donald
22	@realdonaldtrump
20	great

Test & Score

Sampling

● Cross validation

Number of folds: 10

☑ Stratified

○ Random sampling

Repeat train/test: 10

Training set size: 66 %

☑ Stratified

○ Leave one out

○ Test on train data

○ Test on test data

Target Class

(Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
Logistic Regression	0.763	0.820	0.692	0.887	0.567

1.6 Wikipedia

Fetching data from MediaWiki RESTful web service API.

1.6.1 Signals

Inputs:

- (None)

Outputs:

- **Corpus**

  A *Corpus* instance.

1.6.2 Description

Wikipedia widget is used to retrieve texts from Wikipedia API and it is useful mostly for teaching and demonstration.

1. **Query parameters:**
   - Query word list, where each query is listed in a new line.
   - Language of the query. English is set by default.
   - Number of articles to retrieve per query (range 1-25). Please note that querying is done recursively and that disambiguations are also retrieved, sometimes resulting in a larger number of queries than set on the slider.
2. Select which features to include as text features.
3. Information on the output.
4. Produce a report.
5. Run query.

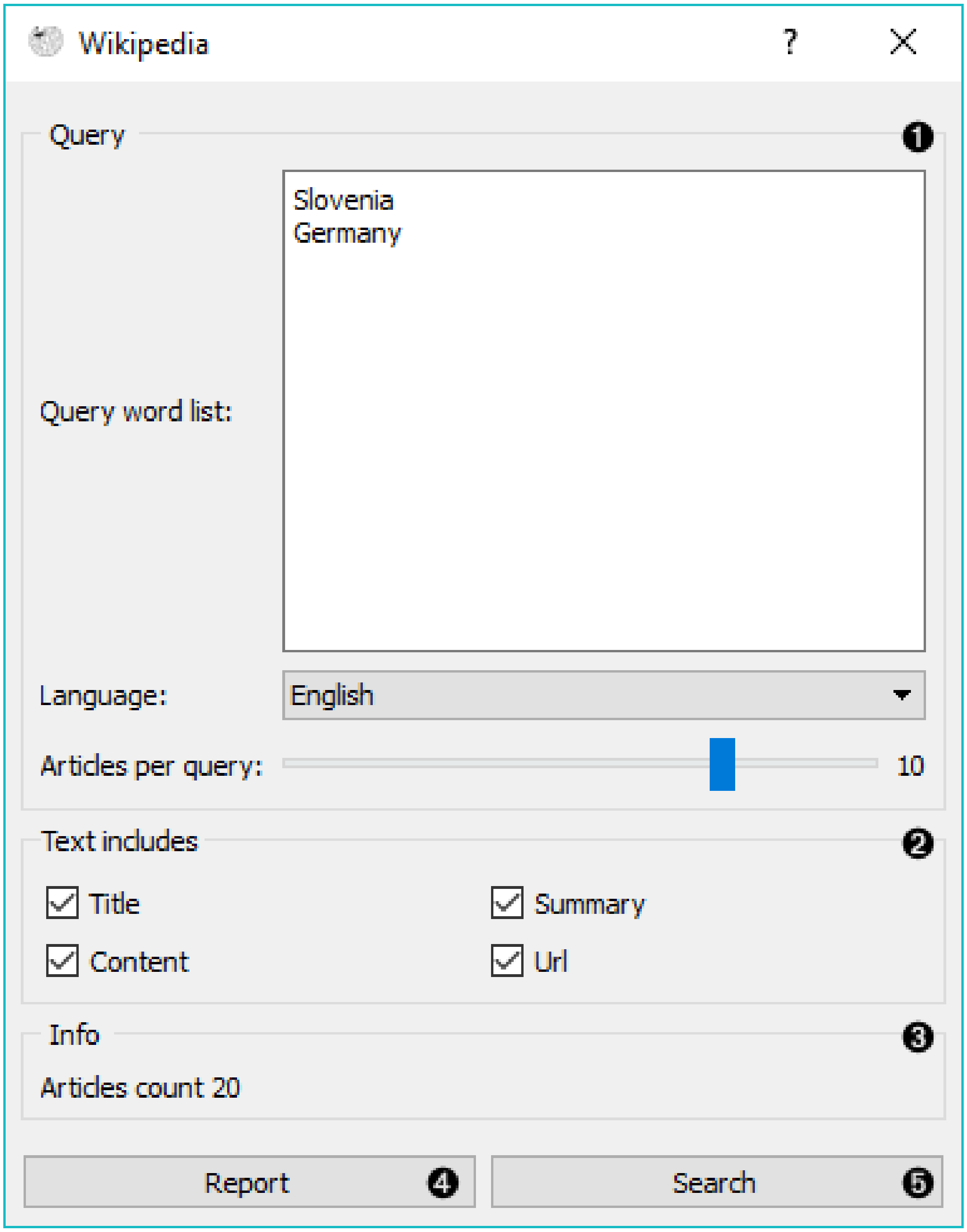
1.6.3 Example

This is a simple example, where we use **Wikipedia** and retrieve the articles on 'Slovenia' and 'Germany'. Then we simply apply default preprocessing with *Preprocess Text* and observe the most frequent words in those articles with *Word Cloud*.

Wikipedia works just like any other corpus widget (*NY Times*, *Twitter*) and can be used accordingly.

1.7 Pubmed

Fetch data from PubMed journals.



Query word list: Slovenia
Germany

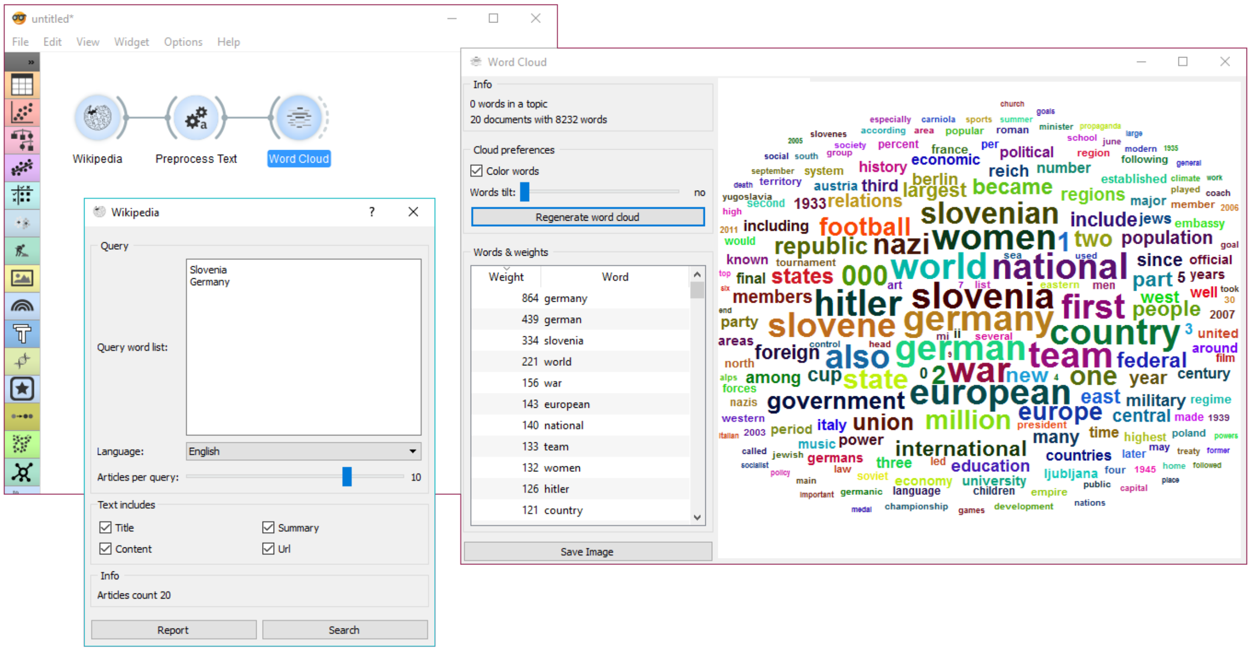
Language: English

Articles per query: 10

☑ Title ☑ Summary

☑ Content ☑ Url

Articles count 20



Wikipedia

Query

Query word list:

Slovenia
Germany

Language: English

Articles per query: 10

Text includes

☑ Title ☑ Summary

☑ Content ☑ Url

Info

Articles count 20

Word Cloud

Info

0 words in a topic

20 documents with 8232 words

Cloud preferences

☑ Color words

Words tilt: no

Words & weights

Weight	Word
864	germany
439	german
334	slovenia
221	world
156	war
143	european
140	national
133	team
132	women
126	hitler
121	country

1.7.1 Signals

Inputs:

- (None)

Outputs:

- **Corpus**

  A *Corpus* instance.

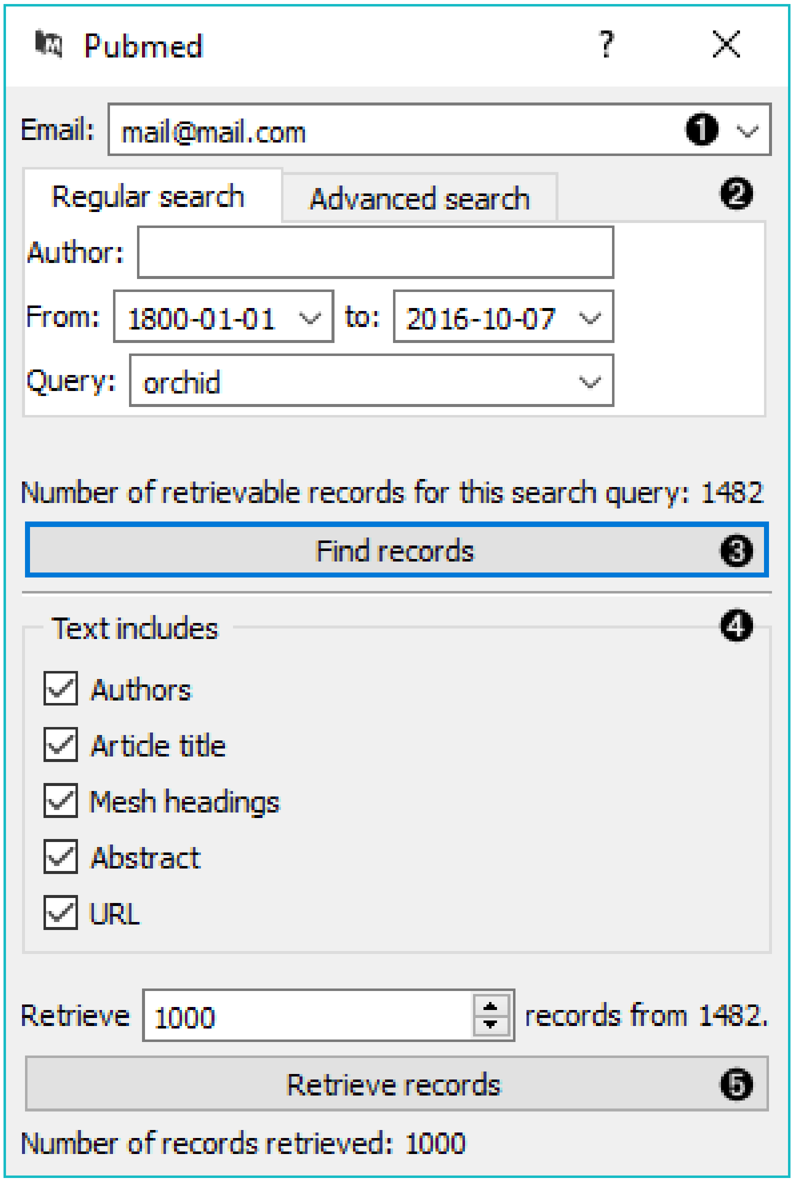
1.7.2 Description

PubMed comprises more than 26 million citations for biomedical literature from MEDLINE, life science journals, and online books. The widget allows you to query and retrieve these entries. You can use regular search or construct advanced queries.

1. Enter a valid e-mail to retrieve queries.
2. ***Regular search***:

   - *Author*: queries entries from a specific author. Leave empty to query by all authors.
   - *From*: define the time frame of publication.
   - *Query*: enter the query.

   Advanced search: enables you to construct complex queries. See PubMed's website to learn how to construct such queries. You can also copy-paste constructed queries from the website.
3. *Find records* finds available data from PubMed matching the query. Number of records found will be displayed above the button.



Pubmed

Email: mail@mail.com

Author:

From: 1800-01-01 to: 2016-10-07

Query: orchid

Number of retrievable records for this search query: 1482

Text includes

☑ Authors

☑ Article title

☑ Mesh headings

☑ Abstract

☑ URL

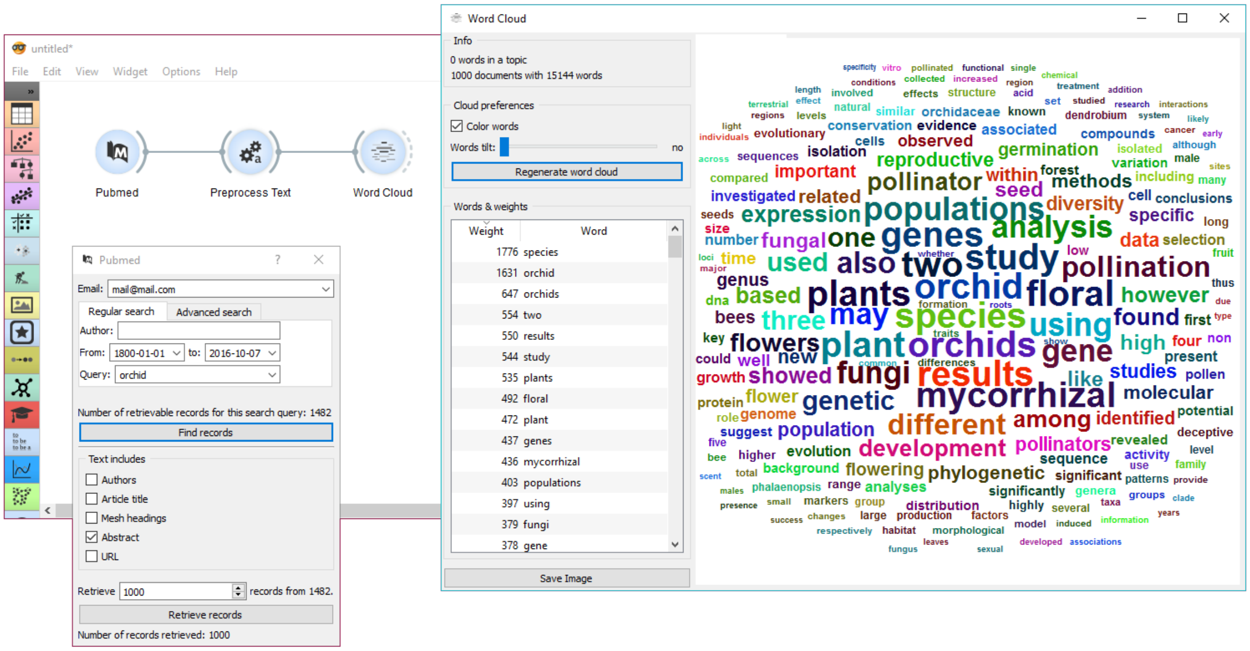
Retrieve 1000 records from 1482.

Number of records retrieved: 1000

4. Define the output. All checked features will be on the output of the widget.
5. Set the number of record you wish to retrieve. Press *Retrieve records* to get results of your query on the output. Below the button is an information on the number of records on the output.

1.7.3 Example

PubMed can be used just like any other data widget. In this example we've queried the database for records on orchids. We retrieved 1000 records and kept only 'abstract' in our meta features to limit the construction of tokens only to this feature.



Pubmed

Email: mail@mail.com

Regular search | Advanced search

Author:

From: 1800-01-01 to: 2016-10-07

Query: orchid

Number of retrievable records for this search query: 1482

Find records

Text includes

☐ Authors
☐ Article title
☐ Mesh headings
☑ Abstract
☐ URL

Retrieve 1000 records from 1482.

Retrieve records

Number of records retrieved: 1000

Word Cloud

Info

0 words in a topic
1000 documents with 15144 words

Cloud preferences

☑ Color words

Words tilt: no

Regenerate word cloud

Words & weights

Weight	Word
1776	species
1631	orchid
647	orchids
554	two
550	results
544	study
535	plants
492	floral
472	plant
437	genes
436	mycorrhizal
403	populations
397	using
379	fungi
378	gene

Save Image

We used *Preprocess Text* to remove stopword and words shorter than 3 characters (regexp `\b\w{1,2}\b`). This will perhaps get rid of some important words denoting chemicals, so we need to be careful with what we filter out. For the sake of quick inspection we only retained longer words, which are displayed by frequency in *Word Cloud*.

1.8 Corpus Viewer

Displays corpus content.

1.8.1 Signals

Inputs:

- **Data**

  Data instance.

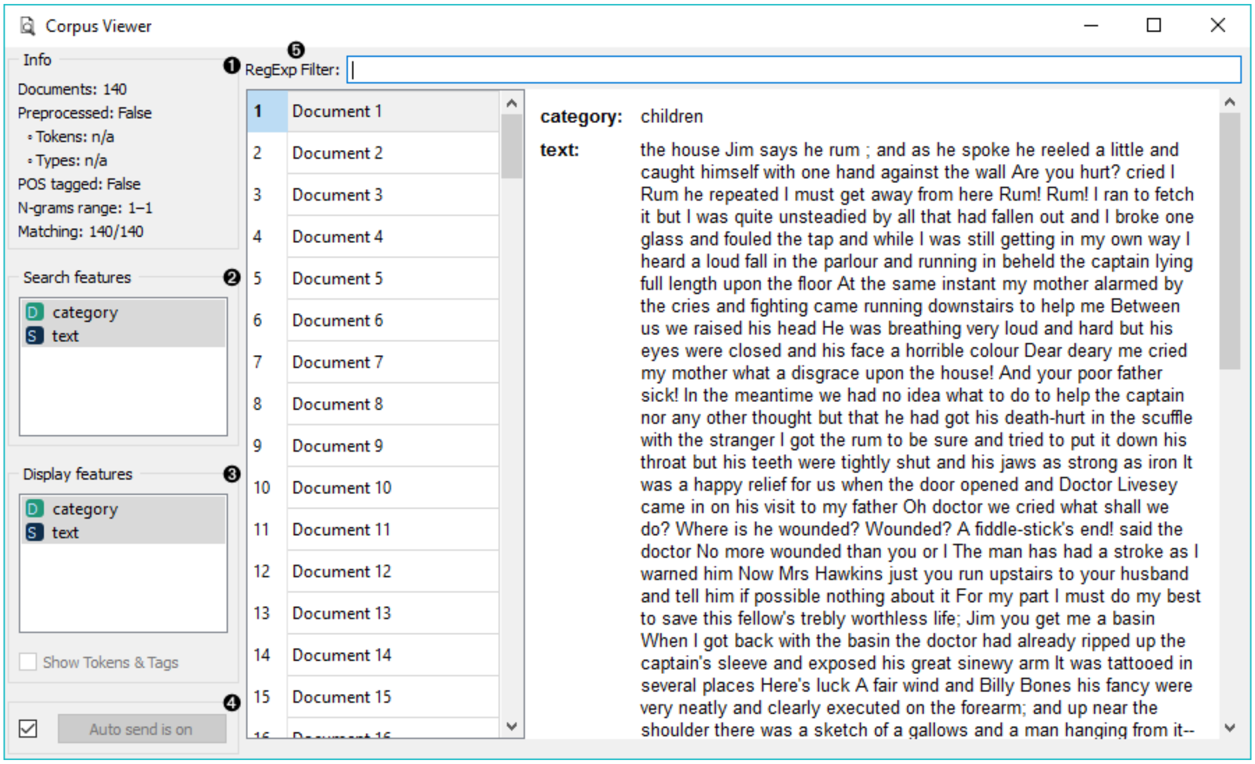
Outputs:

- **Corpus**

  A *Corpus* instance.

1.8.2 Description

Corpus Viewer is primarily meant for viewing text files (instances of *Corpus*), but it can also display other data files from **File** widget. **Corpus Viewer** will always output an instance of corpus. If *RegExp* filtering is used, the widget will output only matching documents.



Documents: 140
Preprocessed: False
◦ Tokens: n/a
◦ Types: n/a
POS tagged: False
N-grams range: 1–1
Matching: 140/140

Search features: D category, S text

Display features: D category, S text

☐ Show Tokens & Tags

☑ Auto send is on

1 Document 1
2 Document 2
3 Document 3
4 Document 4
5 Document 5
6 Document 6
7 Document 7
8 Document 8
9 Document 9
10 Document 10
11 Document 11
12 Document 12
13 Document 13
14 Document 14
15 Document 15
16 Document 16

category: children

text: the house Jim says he rum ; and as he spoke he reeled a little and caught himself with one hand against the wall Are you hurt? cried I Rum he repeated I must get away from here Rum! Rum! I ran to fetch it but I was quite unsteadied by all that had fallen out and I broke one glass and fouled the tap and while I was still getting in my own way I heard a loud fall in the parlour and running in beheld the captain lying full length upon the floor At the same instant my mother alarmed by the cries and fighting came running downstairs to help me Between us we raised his head He was breathing very loud and hard but his eyes were closed and his face a horrible colour Dear deary me cried my mother what a disgrace upon the house! And your poor father sick! In the meantime we had no idea what to do to help the captain nor any other thought but that he had got his death-hurt in the scuffle with the stranger I got the rum to be sure and tried to put it down his throat but his teeth were tightly shut and his jaws as strong as iron It was a happy relief for us when the door opened and Doctor Livesey came in on his visit to my father Oh doctor we cried what shall we do? Where is he wounded? Wounded? A fiddle-stick's end! said the doctor No more wounded than you or I The man has had a stroke as I warned him Now Mrs Hawkins just you run upstairs to your husband and tell him if possible nothing about it For my part I must do my best to save this fellow's trebly worthless life; Jim you get me a basin When I got back with the basin the doctor had already ripped up the captain's sleeve and exposed his great sinewy arm It was tattooed in several places Here's luck A fair wind and Billy Bones his fancy were very neatly and clearly executed on the forearm; and up near the shoulder there was a sketch of a gallows and a man hanging from it--

1. ***Information***:

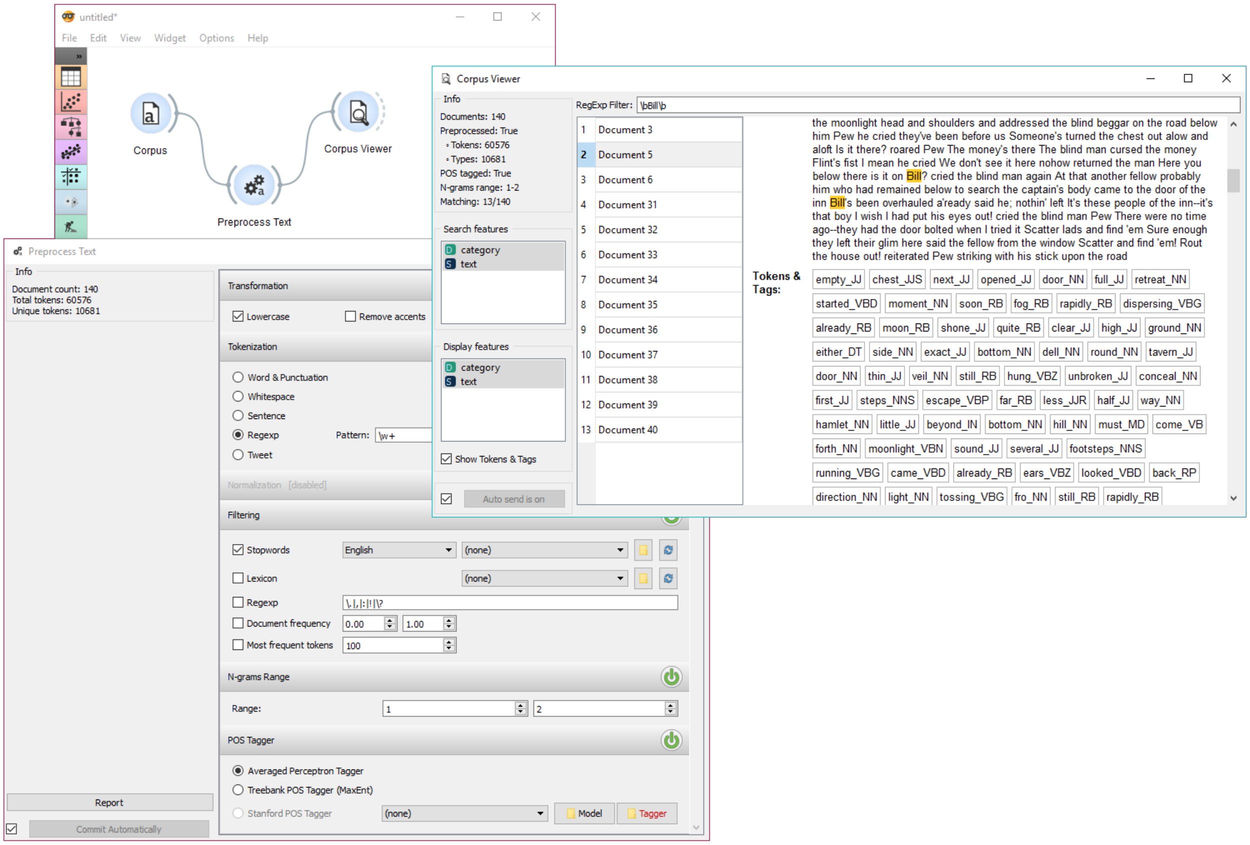
   - *Documents*: number of documents on the input
   - *Preprocessed*: if preprocessor is used, the result is True, else False. Reports also on the number of tokens and types (unique tokens).
   - *POS tagged*: if POS tags are on the input, the result is True, else False.
   - *N-grams range*: if N-grams are set in *Preprocess Text*, results are reported, default is 1-1 (one-grams).
   - *Matching*: number of documents matching the *RegExp Filter*. All documents are output by default.

2. *RegExp Filter*: Python regular expression for filtering documents. By default no documents are filtered (entire corpus is on the output).
3. *Search Features*: features by which the RegExp Filter is filtering. Use Ctrl (Cmd) to select multiple features.
4. *Display Features*: features that are displayed in the viewer. Use Ctrl (Cmd) to select multiple features.
5. *Show Tokens & Tags*: if tokens and POS tag are present on the input, you can check this box to display them.
6. If *Auto commit is on*, changes are communicated automatically. Alternatively press *Commit*.

1.8.3 Example

Corpus Viewer can be used for displaying all or some documents in corpus. In this example, we will first load *book-excerpts.tab*, that already comes with the add-on, into *Corpus* widget. Then we will preprocess the text into words, filter out the stopwords, create bi-grams and add POS tags (more on preprocessing in *Preprocess Text*). Now we want to see the results of preprocessing. In *Corpus Viewer* we can see, how many unique tokens we got and what they are (tick *Show Tokens & Tags*). Since we used also POS tagger to show part-of-speech labels, they will be displayed alongside tokens underneath the text.

Now we will filter out just the documents talking about a character Bill. We use regular expression *\bBill\b* to find the documents containing only the word Bill. You can output matching or non-matching documents, view them in another *Corpus Viewer* or further analyse them.



Corpus Viewer

Info
Documents: 140
Preprocessed: True
- Tokens: 60576
- Types: 10681
POS tagged: True
N-grams range: 1-2
Matching: 13/140

Search features: D category, S text

Display features: D category, S text

☑ Show Tokens & Tags

☑ Auto send is on

RegExp Filter: \bBill\b

1 Document 3
2 Document 5
3 Document 6
4 Document 31
5 Document 32
6 Document 33
7 Document 34
8 Document 35
9 Document 36
10 Document 37
11 Document 38
12 Document 39
13 Document 40

the moonlight head and shoulders and addressed the blind beggar on the road below him Pew he cried they've been before us Someone's turned the chest out alow and aloft Is it there? roared Pew The money's there The blind man cursed the money Flint's fist I mean he cried We don't see it here nohow returned the man Here you below there is it on Bill? cried the blind man again At that another fellow probably him who had remained below to search the captain's body came to the door of the inn Bill's been overhauled a'ready said he; nothin' left It's these people of the inn--it's that boy I wish I had put his eyes out! cried the blind man Pew There were no time ago--they had the door bolted when I tried it Scatter lads and find 'em Sure enough they left their glim here said the fellow from the window Scatter and find 'em! Rout the house out! reiterated Pew striking with his stick upon the road

Tokens & Tags:
empty_JJ chest_JJS next_JJ opened_JJ door_NN full_JJ retreat_NN
started_VBD moment_NN soon_RB fog_RB rapidly_RB dispersing_VBG
already_RB moon_RB shone_JJ quite_RB clear_JJ high_JJ ground_NN
either_DT side_NN exact_JJ bottom_NN dell_NN round_NN tavern_JJ
door_NN thin_JJ veil_NN still_RB hung_VBZ unbroken_JJ conceal_NN
first_JJ steps_NNS escape_VBP far_RB less_JJR half_JJ way_NN
hamlet_NN little_JJ beyond_IN bottom_NN hill_NN must_MD come_VB
forth_NN moonlight_VBN sound_JJ several_JJ footsteps_NNS
running_VBG came_VBD already_RB ears_VBZ looked_VBD back_RP
direction_NN light_NN tossing_VBG fro_NN still_RB rapidly_RB

Preprocess Text

Info
Document count: 140
Total tokens: 60576
Unique tokens: 10681

Transformation
☑ Lowercase ☐ Remove accents

Tokenization
○ Word & Punctuation
○ Whitespace
○ Sentence
● Regexp Pattern: \w+
○ Tweet

Normalization [disabled]

Filtering
☑ Stopwords English (none)
☐ Lexicon (none)
☐ Regexp \.|,|:|!|\?
☐ Document frequency 0.00 1.00
☐ Most frequent tokens 100

N-grams Range
Range: 1 2

POS Tagger
● Averaged Perceptron Tagger
○ Treebank POS Tagger (MaxEnt)
○ Stanford POS Tagger (none) Model Tagger

Report

☑ Commit Automatically

1.9 Preprocess Text

Preprocesses corpus with selected methods.

1.9.1 Signals

Inputs:

- **Corpus**

  Corpus instance.

Outputs:

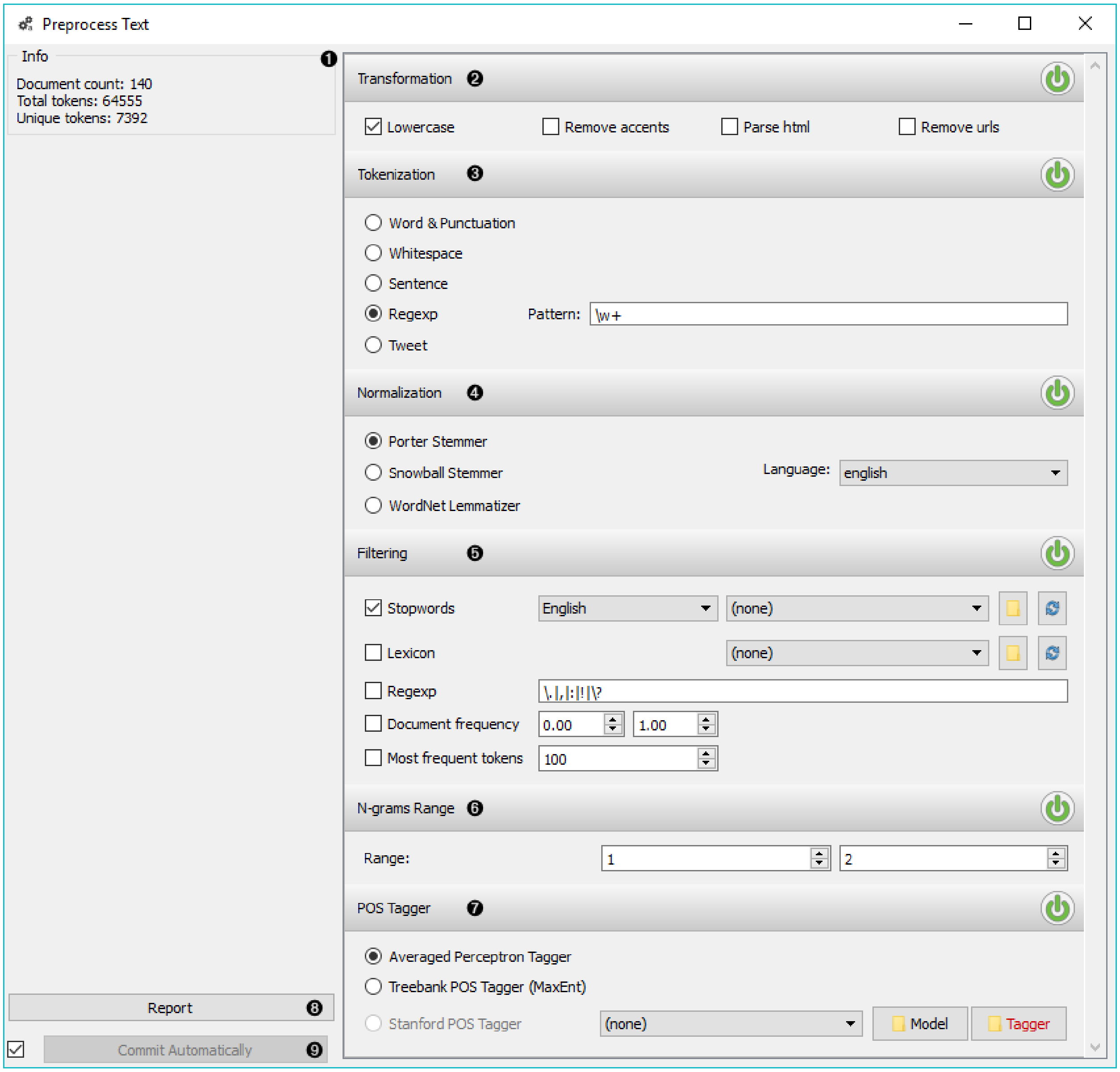
- **Corpus**

  Preprocessed corpus.

1.9.2 Description

Preprocess Text splits your text into smaller units (tokens), filters them, runs normalization (stemming, lemmatization), creates n-grams and tags tokens with part-of-speech labels. Steps in the analysis are applied sequentially and can be turned on or off.

1. **Information on preprocessed data**. *Document count* reports on the number of documents on the input. *Total tokens* counts all the tokens in corpus. *Unique tokens* excludes duplicate tokens and reports only on unique tokens in the corpus.
2. **Transformation transforms input data. It applies lowercase transformation by default.**
   - *Lowercase* will turn all text to lowercase.
   - ***Remove accents* will remove all diacritics/accents in text.** naïve → naive
   - ***Parse html* will detect html tags and parse out text only.** <a href...>Some text</a> → Some text
   - ***Remove urls* will remove urls from text.** This is a http://orange.biolab.si/ url. → This is a url.
3. **Tokenization is the method of breaking the text into smaller components (words, sentences, bigrams).**
   - ***Word & Punctuation* will split the text by words and keep punctuation symbols.** This example. → (This), (example), (.)
   - ***Whitespace* will split the text by whitespace only.** This example. → (This), (example.)
   - ***Sentence* will split the text by fullstop, retaining only full sentences.** This example. Another example. → (This example.), (Another example.)
   - Regexp will split the text by provided regex. It splits by words only by default (omits punctuation).
   - ***Tweet* will split the text by pre-trained Twitter model, which keeps hashtags, emoticons and other special symbol** This example. :-) #simple → (This), (example), (.), (:-)), (#simple)
4. **Normalization applies stemming and lemmatization to words. (I've always loved cats. → I have alway love cat.) For langu**
   - Porter Stemmer applies the original Porter stemmer.
   - Snowball Stemmer applies an improved version of Porter stemmer (Porter2). Set the language for normalization, default is English.
   - WordNet Lemmatizer applies a networks of cognitive synonyms to tokens based on a large lexical database of English.
5. **Filtering removes or keeps a selection of words.**
   - *Stopwords* removes stopwords from text (e.g. removes 'and', 'or', 'in'...). Select the language to filter by, English is set as default. You can also load your own list of stopwords provided in a simple *.txt file with one stopword per line.



Preprocess Text

Info 1

Document count: 140
Total tokens: 64555
Unique tokens: 7392

Transformation 2

☑ Lowercase ☐ Remove accents ☐ Parse html ☐ Remove urls

Tokenization 3

○ Word & Punctuation
○ Whitespace
○ Sentence
● Regexp Pattern: \w+
○ Tweet

Normalization 4

● Porter Stemmer
○ Snowball Stemmer Language: english
○ WordNet Lemmatizer

Filtering 5

☑ Stopwords English (none)
☐ Lexicon (none)
☐ Regexp \.|,|:|!|\?
☐ Document frequency 0.00 1.00
☐ Most frequent tokens 100

N-grams Range 6

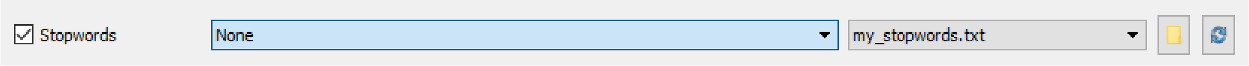
Range: 1 2

POS Tagger 7

● Averaged Perceptron Tagger
○ Treebank POS Tagger (MaxEnt)
○ Stanford POS Tagger (none) Model Tagger

Report 8

☑ Commit Automatically 9


☑ Stopwords None my_stopwords.txt

Click 'browse' icon to select the file containing stopwords. If the file was properly loaded, its name will be displayed next to pre-loaded stopwords. Change 'English' to 'None' if you wish to filter out only the provided stopwords. Click 'reload' icon to reload the list of stopwords.

- *Lexicon* keeps only words provided in the file. Load a *.txt file with one word per line to use as lexicon. Click 'reload' icon to reload the lexicon.
- *Regexp* removes words that match the regular expression. Default is set to remove punctuation.
- *Document frequency* keeps tokens that appear in not less than and not more than the specified number / percentage of documents. If you provide integers as parameters, it keeps only tokens that appear in the specified number of documents. E.g. DF = (3, 5) keeps only tokens that appear in 3 or more and 5 or less documents. If you provide floats as parameters, it keeps only tokens that appear in the specified percentage of documents. E.g. DF = (0.3, 0.5) keeps only tokens that appear in 30% to 50% of documents. Default returns all tokens.
- *Most frequent tokens* keeps only the specified number of most frequent tokens. Default is a 100 most frequent tokens.

6. **N-grams Range** creates n-grams from tokens. Numbers specify the range of n-grams. Default returns one-grams and two-grams.
7. **POS Tagger runs part-of-speech tagging on tokens.**
   - Averaged Perceptron Tagger runs POS tagging with Matthew Honnibal's averaged perceptron tagger.
   - Treebank POS Tagger (MaxEnt) runs POS tagging with a trained Penn Treebank model.
   - Stanford POS Tagger runs a log-linear part-of-speech tagger designed by Toutanova et al. Please download it from the provided website and load it in Orange.
8. Produce a report.
9. If *Commit Automatically* is on, changes are communicated automatically. Alternatively press *Commit*.

Note: Preprocess Text applies preprocessing steps in the order they are listed. This means it will first transform the text, then apply tokenization, POS tags, normalization, filtering and finally constructs n-grams based on given tokens. This is especially important for WordNet Lemmatizer since it requires POS tags for proper normalization.

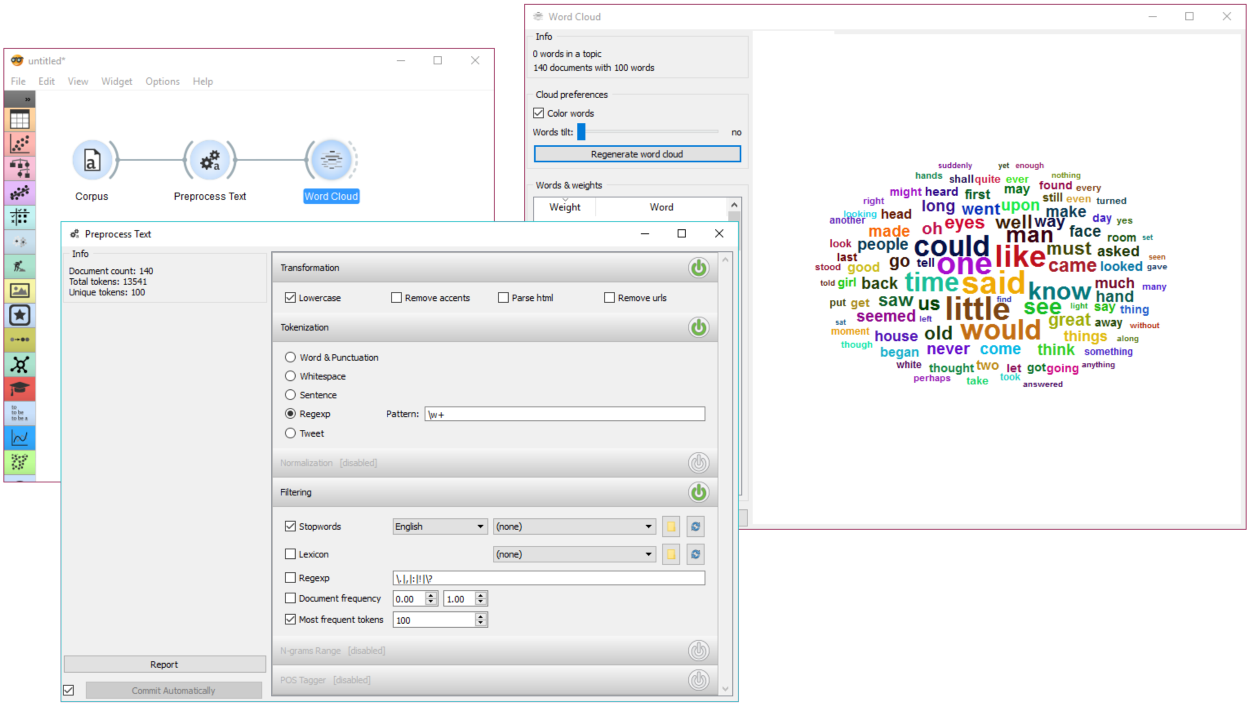
1.9.3 Useful Regular Expressions

Here are some useful regular expressions for quick filtering:

`\bword\b`	matches exact word
`\w+`	matches only words, no punctuation
`\b(B|b)\w+\b`	matches words beginning with the letter b
`\w{4,}`	matches words that are longer than 4 characters
`\b\w+(Y|y)\b`	matches words ending with the letter y

1.9.4 Examples

In the first example we will observe the effects of preprocessing on our text. We are working with *book-excerpts.tab* that we've loaded with *Corpus* widget. We have connected **Preprocess Text** to **Corpus** and retained default preprocessing methods (lowercase, per-word tokenization and stopword removal). The only additional parameter we've added as outputting only the first 100 most frequent tokens. Then we connected **Preprocess Text** with *Word Cloud* to observe words that are the most frequent in our text. Play around with different parameters, to see how they transform the output.


untitled*
File
Edit
View
Widget
Options
Help
Corpus
Preprocess Text
Word Cloud
suddenly yet enough
hands shall quite ever nothing
might heard first may found every
right long went upon still even turned
looking head make day yes
another eyes well way
made oh man face room set
look people could must asked
last like seen
stood good go tell one came looked gave
told girl back time said know much many
put get saw us find hand
little see light say thing
sat seemed left great away without
moment house old would things along
though began never come think something
white thought two let got going anything
perhaps take took answered

Word Cloud

Info
0 words in a topic
140 documents with 100 words

Cloud preferences
☑ Color words
Words tilt: no
Regenerate word cloud

Words & weights

Weight	Word

Preprocess Text

Info
Document count: 140
Total tokens: 13541
Unique tokens: 100

Report
☑ Commit Automatically

Transformation
☑ Lowercase □ Remove accents □ Parse html □ Remove urls

Tokenization
○ Word & Punctuation
○ Whitespace
○ Sentence
● Regexp Pattern: \w+
○ Tweet

Normalization [disabled]

Filtering
☑ Stopwords English (none)
□ Lexicon (none)
□ Regexp \.|,|:|!|\?
□ Document frequency 0.00 1.00
☑ Most frequent tokens 100

N-grams Range [disabled]

POS Tagger [disabled]

The second example is slightly more complex. We first acquired our data with *Twitter* widget. We quired the internet for tweets from users @HillaryClinton and @realDonaldTrump and got their tweets from the past two weeks, 242 in total.

In **Preprocess Text** there's *Tweet* tokenization available, which retains hashtags, emojis, mentions and so on. However, this tokenizer doesn't get rid of punctuation, thus we expanded the Regexp filtering with symbols that we wanted to get rid of. We ended up with word-only tokens, which we displayed in *Word Cloud*. Then we created a schema for predicting author based on tweet content, which is explained in more details in the documentation for *Twitter* widget.

1.10 Bag of Words

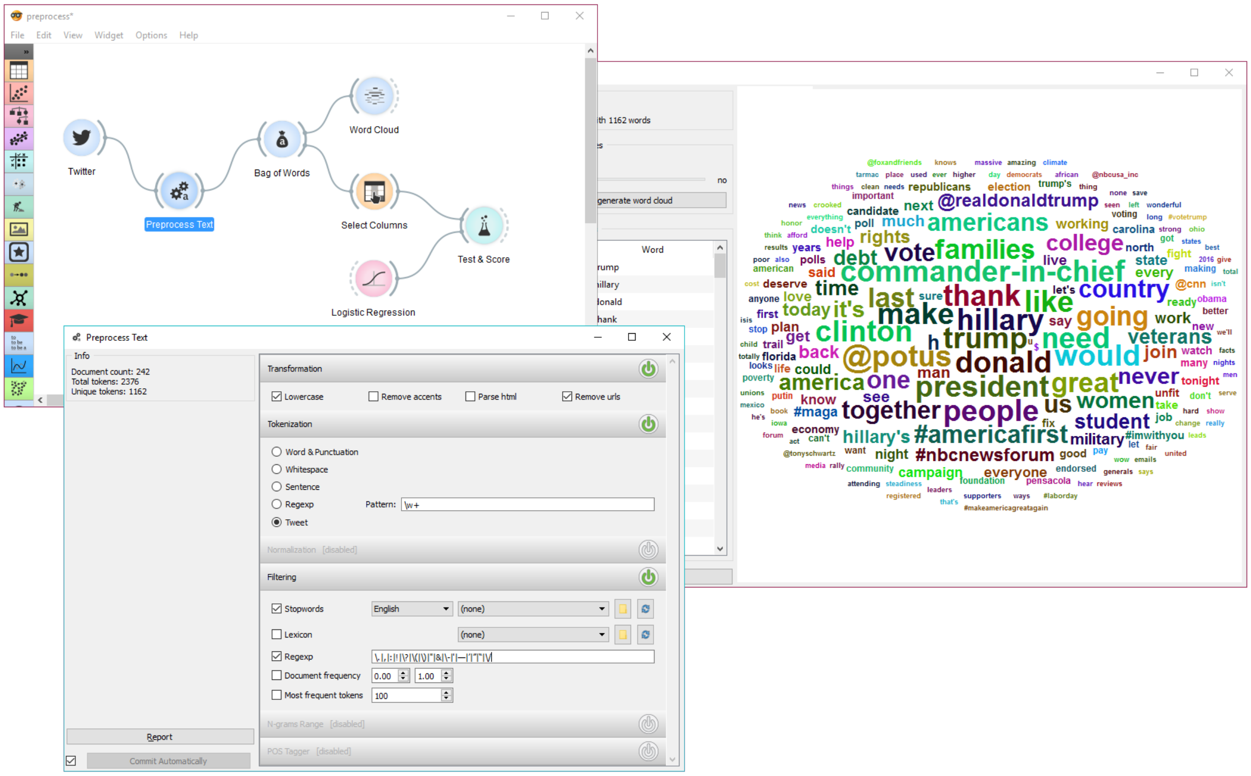
Generates a bag of words from the input corpus.

1.10.1 Signals

Inputs:

- **Corpus**

  Corpus instance.


preprocess*
File Edit View Widget Options Help
Twitter
Preprocess Text
Bag of Words
Word Cloud
Select Columns
Test & Score
Logistic Regression
ith 1162 words
es
no
generate word cloud
Word
rump
illary
lonald
hank
@foxandfriends knows massive amazing climate
tarmac place used ever higher day democrats african @nbcusa_inc
things clean needs republicans election trump's thing
important none save
news crooked next @realdonaldtrump seen left wonderful
candidate voting
everything long #votetrump
honor poll much americans working carolina strong ohio
think afford doesn't rights got
results years help college states best
north
poor also polls debt vote families live fight 2016 give
american state
said commander-in-chief every making total
cost deserve time @cnn isn't
anyone love last sure thank like let's country
ready obama
first today it's make hillary say going work better
isis
stop plan clinton new
get h trump u $ need veterans we'll
child trail
totally florida back @potus donald would join watch facts
looks life could many nights
man
poverty america one never tonight men
president great
unions putin know see unfit don't serve
mexico women take
book #maga together people us hard show
he's
iowa fix student job change really
economy
forum act can't hillary's #americafirst military #imwithyou leads
let fair
@tonyschwartz want night #nbcnewsforum good pay united
wow emails
media rally community campaign everyone endorsed generals says
attending steadiness foundation pensacola hear reviews
leaders
registered supporters ways #laborday
that's
#makeamericagreatagain

Preprocess Text

Info

Document count: 242
Total tokens: 2376
Unique tokens: 1162

Transformation

☑ Lowercase □ Remove accents □ Parse html ☑ Remove urls

Tokenization

○ Word & Punctuation
○ Whitespace
○ Sentence
○ Regexp Pattern: \w+
● Tweet

Normalization [disabled]

Filtering

☑ Stopwords English (none)
□ Lexicon (none)
☑ Regexp \.|,|:|!|\?|\(|\)|"|&|\-|'|—|'|"|"|\/
□ Document frequency 0.00 1.00
□ Most frequent tokens 100

N-grams Range [disabled]

POS Tagger [disabled]

Report

☑ Commit Automatically

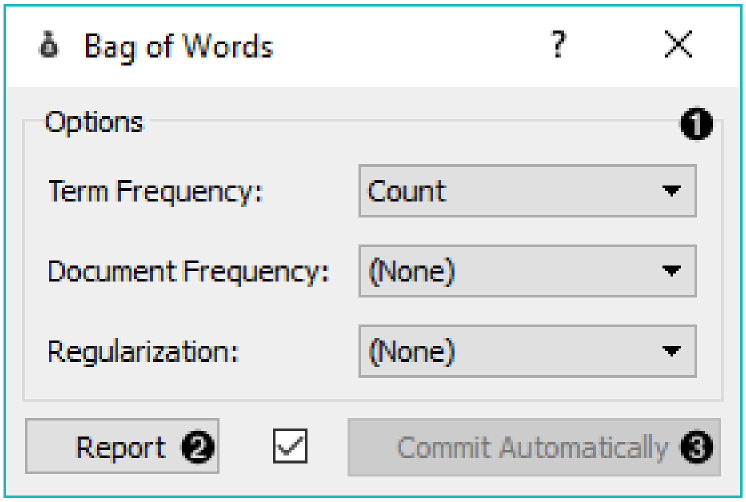
Outputs:

- **Corpus**

  Corpus with bag of words.

1.10.2 Description

Bag of Words model creates a corpus with word counts for each data instance (document). The count can be either absolute, binary (contains or does not contain) or sublinear (logarithm of the term frequency). Bag of words model is required in combination with *Word Enrichment* and could be used for predictive modelling.



Term Frequency: Count

Document Frequency: (None)

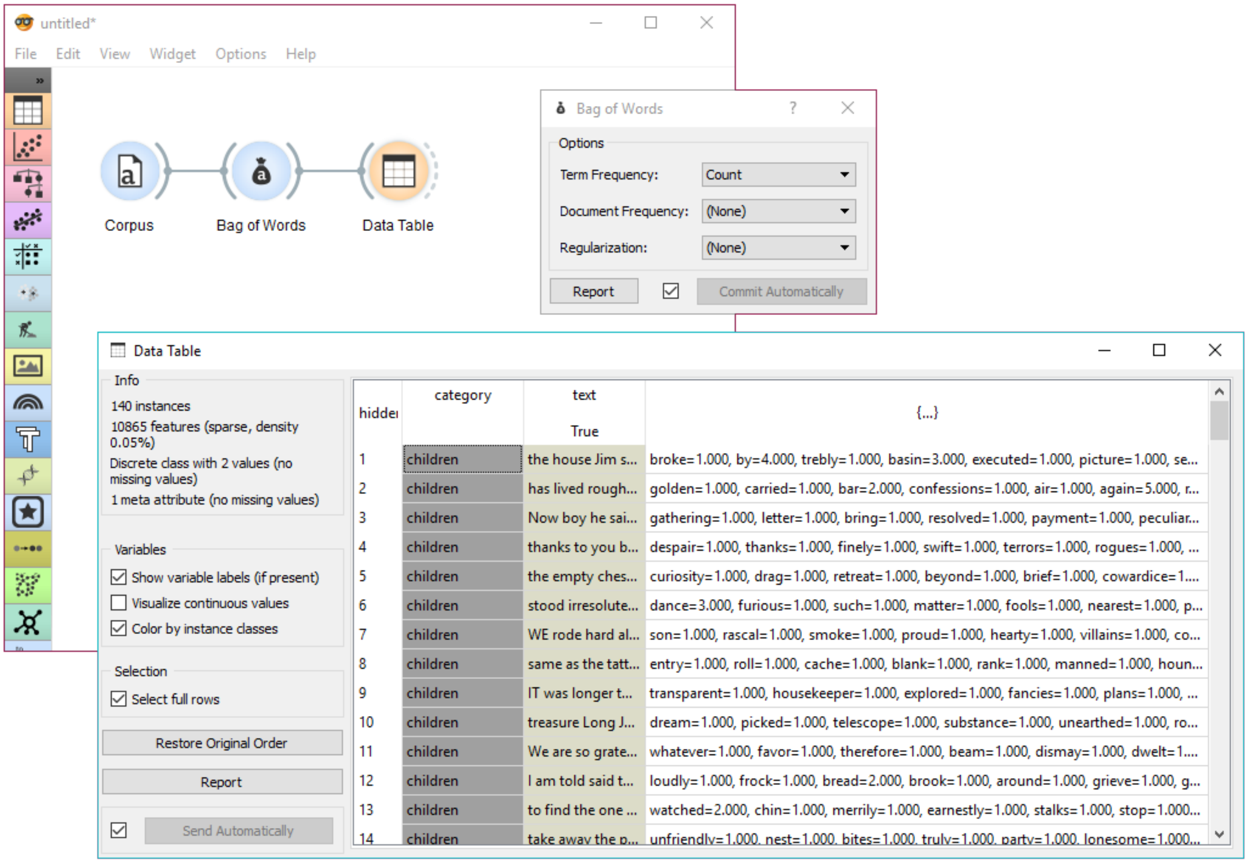
Regularization: (None)

☑ Commit Automatically

1. **Parameters for bag of words model:**
   - **Term Frequency:**
     - Count: number of occurences of a word in a document
     - Binary: word appears or does not appear in the document
     - Sublinear: logarithm of term frequency (count)
   - **Document Frequency:**
     - (None)
     - IDF: inverse document frequency
     - Smooth IDF: adds one to document frequencies to prevent zero division.
   - **Regulariation:**
     - (None)
     - L1 (Sum of elements): normalizes vector length to sum of elements
     - L2 (Euclidean): normalizes vector length to sum of squares
2. Produce a report.
3. If *Commit Automatically* is on, changes are communicated automatically. Alternatively press *Commit*.

1.10.3 Example

In the first example we will simply check how the bag of words model looks like. Load *book-excerpts.tab* with *Corpus* widget and connect it to **Bag of Words**. Here we kept the defaults - a simple count of term frequencies. Check what the **Bag of Words** outputs with **Data Table**. The final column in white represents term frequencies for each document.



Options

Term Frequency: Count

Document Frequency: (None)

Regularization: (None)

☑ Commit Automatically

Info

140 instances
10865 features (sparse, density 0.05%)
Discrete class with 2 values (no missing values)
1 meta attribute (no missing values)

Variables

☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

Selection

☑ Select full rows

☑ Send Automatically

hidden	category	text True	{...}
1	children	the house Jim s...	broke=1.000, by=4.000, trebly=1.000, basin=3.000, executed=1.000, picture=1.000, se...
2	children	has lived rough...	golden=1.000, carried=1.000, bar=2.000, confessions=1.000, air=1.000, again=5.000, r...
3	children	Now boy he sai...	gathering=1.000, letter=1.000, bring=1.000, resolved=1.000, payment=1.000, peculiar...
4	children	thanks to you b...	despair=1.000, thanks=1.000, finely=1.000, swift=1.000, terrors=1.000, rogues=1.000, ...
5	children	the empty ches...	curiosity=1.000, drag=1.000, retreat=1.000, beyond=1.000, brief=1.000, cowardice=1....
6	children	stood irresolute...	dance=3.000, furious=1.000, such=1.000, matter=1.000, fools=1.000, nearest=1.000, p...
7	children	WE rode hard al...	son=1.000, rascal=1.000, smoke=1.000, proud=1.000, hearty=1.000, villains=1.000, co...
8	children	same as the tatt...	entry=1.000, roll=1.000, cache=1.000, blank=1.000, rank=1.000, manned=1.000, houn...
9	children	IT was longer t...	transparent=1.000, housekeeper=1.000, explored=1.000, fancies=1.000, plans=1.000, ...
10	children	treasure Long J...	dream=1.000, picked=1.000, telescope=1.000, substance=1.000, unearthed=1.000, ro...
11	children	We are so grate...	whatever=1.000, favor=1.000, therefore=1.000, beam=1.000, dismay=1.000, dwelt=1....
12	children	I am told said t...	loudly=1.000, frock=1.000, bread=2.000, brook=1.000, around=1.000, grieve=1.000, g...
13	children	to find the one ...	watched=2.000, chin=1.000, merrily=1.000, earnestly=1.000, stalks=1.000, stop=1.000...
14	children	take away the p...	unfriendly=1.000, nest=1.000, bites=1.000, truly=1.000, party=1.000, lonesome=1.000...

In the second example we will try to predict document category. We are still using the *book-excerpts.tab* data set, which we sent through *Preprocess Text* with default parameters. Then we connected **Preprocess Text** to **Bag of Words** to obtain term frequencies by which we will compute the model.

Connect **Bag of Words** to **Test & Score** for predictive modelling. Connect **SVM** or any other classifier to **Test & Score** as well (both on the left side). **Test & Score** will now compute performance scores for each learner on the input. Here we got quite impressive results with SVM. Now we can check, where the model made a mistake.

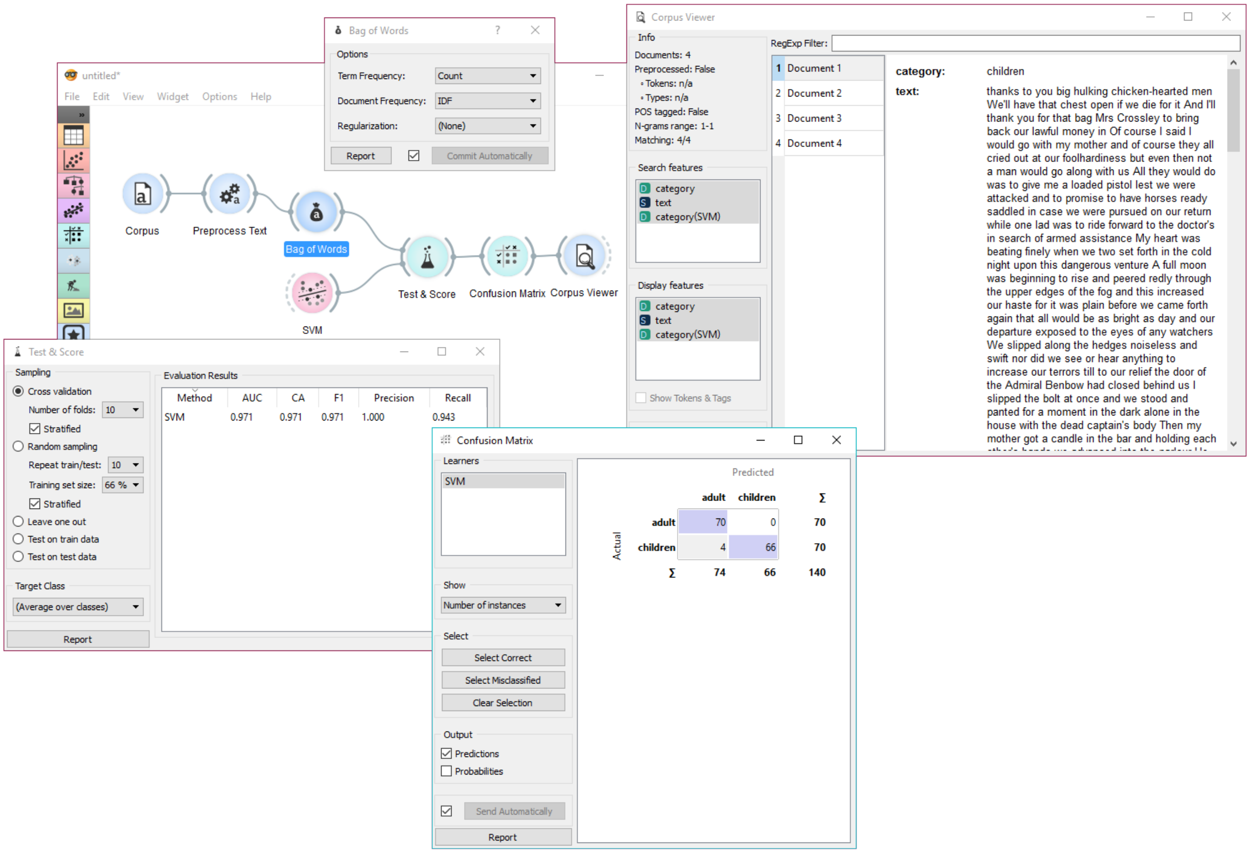
Add **Confusion Matrix** to **Test & Score**. Confusion matrix displays correctly and incorrectly classified documents. *Select Misclassified* will output misclassified documents, which we can further inspect with *Corpus Viewer*.

1.11 Topic Modelling

Topic modelling with Latent Diriclet Allocation, Latent Semantic Indexing or Hierarchical Dirichlet Process.

1.11.1 Signals

Inputs:



Bag of Words

Options

Term Frequency: Count

Document Frequency: IDF

Regularization: (None)

Report ☑ Commit Automatically

Corpus Viewer

Info

Documents: 4
Preprocessed: False
◦ Tokens: n/a
◦ Types: n/a
POS tagged: False
N-grams range: 1-1
Matching: 4/4

Search features: D category, S text, D category(SVM)

Display features: D category, S text, D category(SVM)

□ Show Tokens & Tags

RegExp Filter:

1 Document 1
2 Document 2
3 Document 3
4 Document 4

category: children

text: thanks to you big hulking chicken-hearted men We'll have that chest open if we die for it And I'll thank you for that bag Mrs Crossley to bring back our lawful money in Of course I said I would go with my mother and of course they all cried out at our foolhardiness but even then not a man would go along with us All they would do was to give me a loaded pistol lest we were attacked and to promise to have horses ready saddled in case we were pursued on our return while one lad was to ride forward to the doctor's in search of armed assistance My heart was beating finely when we two set forth in the cold night upon this dangerous venture A full moon was beginning to rise and peered redly through the upper edges of the fog and this increased our haste for it was plain before we came forth again that all would be as bright as day and our departure exposed to the eyes of any watchers We slipped along the hedges noiseless and swift nor did we see or hear anything to increase our terrors till to our relief the door of the Admiral Benbow had closed behind us I slipped the bolt at once and we stood and panted for a moment in the dark alone in the house with the dead captain's body Then my mother got a candle in the bar and holding each

Test & Score

Sampling

● Cross validation
Number of folds: 10
☑ Stratified
○ Random sampling
Repeat train/test: 10
Training set size: 66 %
☑ Stratified
○ Leave one out
○ Test on train data
○ Test on test data

Target Class: (Average over classes)

Report

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
SVM	0.971	0.971	0.971	1.000	0.943

Confusion Matrix

Learners: SVM

Show: Number of instances

Select: Select Correct | Select Misclassified | Clear Selection

Output
☑ Predictions
□ Probabilities

☑ Send Automatically

Report

Actual \ Predicted	adult	children	Σ
adult	70	0	70
children	4	66	70
Σ	74	66	140

- **Corpus**

  Corpus instance.

Outputs:

- **Data**

  Data with topic weights appended.

- **Topics**

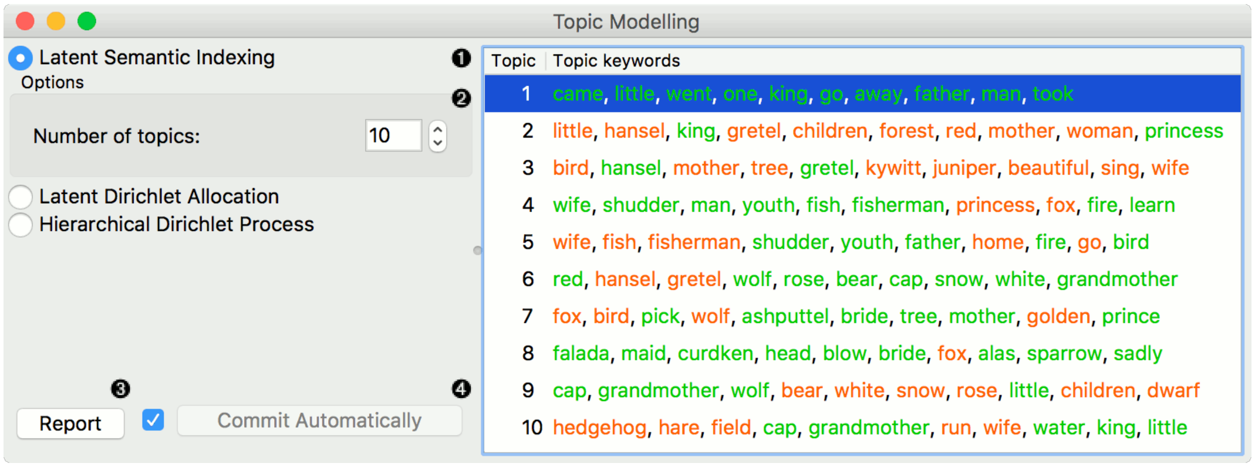
  Selected topics with word weights.

- **All Topics**

  Topic weights by tokens.

1.11.2 Description

Topic Modelling discovers abstract topics in a corpus based on clusters of words found in each document and their respective frequency. A document typically contains multiple topics in different proportions, thus the widget also reports on the topic weight per document.



● Latent Semantic Indexing

Options

Number of topics: 10

○ Latent Dirichlet Allocation

○ Hierarchical Dirichlet Process

☑ Commit Automatically

Topic	Topic keywords
1	came, little, went, one, king, go, away, father, man, took
2	little, hansel, king, gretel, children, forest, red, mother, woman, princess
3	bird, hansel, mother, tree, gretel, kywitt, juniper, beautiful, sing, wife
4	wife, shudder, man, youth, fish, fisherman, princess, fox, fire, learn
5	wife, fish, fisherman, shudder, youth, father, home, fire, go, bird
6	red, hansel, gretel, wolf, rose, bear, cap, snow, white, grandmother
7	fox, bird, pick, wolf, ashputtel, bride, tree, mother, golden, prince
8	falada, maid, curdken, head, blow, bride, fox, alas, sparrow, sadly
9	cap, grandmother, wolf, bear, white, snow, rose, little, children, dwarf
10	hedgehog, hare, field, cap, grandmother, run, wife, water, king, little

1. **Topic modelling algorithm:**

   - Latent Semantic Indexing
   - Latent Dirichlet Allocation
   - Hierarchical Dirichlet Process

2. **Parameters for the algorithm. LSI and LDA accept only the number of topics modelled, with the default set to 10. HDP, h**

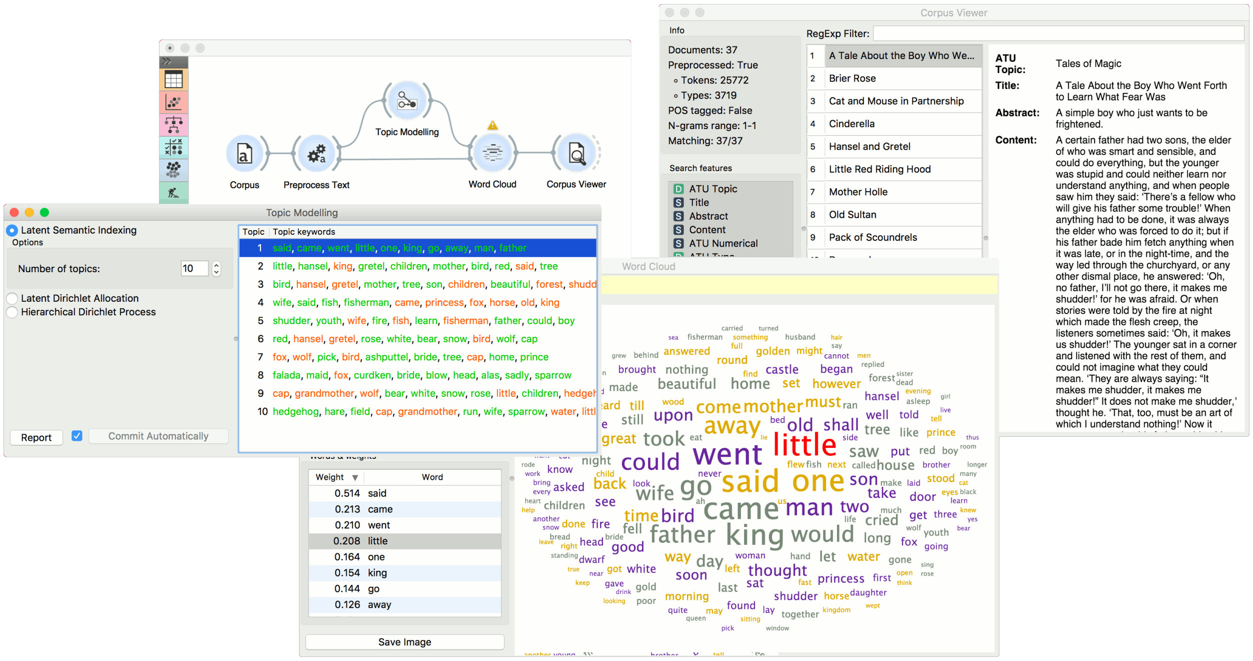
   - First level concentration (γ): distribution at the first (corpus) level of Dirichlet Process
   - Second level concentration (α): distribution at the second (document) level of Dirichlet Process
   - The topic Dirichlet (α): concentration parameter used for the topic draws
   - Top level truncation (T): corpus-level truncation (no of topics)
   - Second level truncation (K): document-level truncation (no of topics)
   - Learning rate (κ): step size

  - Slow down parameter (τ)

3. Produce a report.
4. If *Commit Automatically* is on, changes are communicated automatically. Alternatively press *Commit*.

1.11.3 Example

In the first example, we present a simple use of the **Topic Modelling** widget. First we load *grimm-tales-selected.tab* data set and use *Preprocess Text* to tokenize by words only and remove stopwords. Then we connect **Preprocess Text** to **Topic Modelling**, where we use a simple *Latent Semantic Indexing* to find 10 topics in the text.



Topic Modelling

● Latent Semantic Indexing

Options

Number of topics: 10

○ Latent Dirichlet Allocation

○ Hierarchical Dirichlet Process

Report ☑ Commit Automatically

Topic	Topic keywords
1	said, came, went, little, one, king, go, away, man, father
2	little, hansel, king, gretel, children, mother, bird, red, said, tree
3	bird, hansel, gretel, mother, tree, son, children, beautiful, forest, shudd
4	wife, said, fish, fisherman, came, princess, fox, horse, old, king
5	shudder, youth, wife, fire, fish, learn, fisherman, father, could, boy
6	red, hansel, gretel, rose, white, bear, snow, bird, wolf, cap
7	fox, wolf, pick, bird, ashputtel, bride, tree, cap, home, prince
8	falada, maid, fox, curdken, bride, blow, head, alas, sadly, sparrow
9	cap, grandmother, wolf, bear, white, snow, rose, little, children, hedgeh
10	hedgehog, hare, field, cap, grandmother, run, wife, sparrow, water, littl

Weight ▼	Word
0.514	said
0.213	came
0.210	went
0.208	little
0.164	one
0.154	king
0.144	go
0.126	away

Corpus Viewer

Info

Documents: 37
Preprocessed: True
∘ Tokens: 25772
∘ Types: 3719
POS tagged: False
N-grams range: 1-1
Matching: 37/37

Search features

D ATU Topic
S Title
S Abstract
S Content
S ATU Numerical

RegExp Filter:

1	A Tale About the Boy Who We...
2	Brier Rose
3	Cat and Mouse in Partnership
4	Cinderella
5	Hansel and Gretel
6	Little Red Riding Hood
7	Mother Holle
8	Old Sultan
9	Pack of Scoundrels

ATU Topic: Tales of Magic

Title: A Tale About the Boy Who Went Forth to Learn What Fear Was

Abstract: A simple boy who just wants to be frightened.

Content: A certain father had two sons, the elder of who was smart and sensible, and could do everything, but the younger was stupid and could neither learn nor understand anything, and when people saw him they said: ‘There’s a fellow who will give his father some trouble!’ When anything had to be done, it was always the elder who was forced to do it; but if his father bade him fetch anything when it was late, or in the night-time, and the way led through the churchyard, or any other dismal place, he answered: ‘Oh, no father, I’ll not go there, it makes me shudder!’ for he was afraid. Or when stories were told by the fire at night which made the flesh creep, the listeners sometimes said: ‘Oh, it makes us shudder!’ The younger sat in a corner and listened with the rest of them, and could not imagine what they could mean. ‘They are always saying: “It makes me shudder, it makes me shudder!” It does not make me shudder,’ thought he. ‘That, too, must be an art of which I understand nothing!’ Now it

LSI provides both positive and negative weights per topic. A positive weight means the word is highly representative of a topic, while a negative weight means the word is highly unrepresentative of a topic (the less it occurs in a text, the more likely the topic). Positive words are colored green and negative words are colored red.

We then select the first topic and display the most frequent words in the topic in *Word Cloud*. We also connected **Preprocess Text** to **Word Cloud** in order to be able to output selected documents. Now we can select a specific word in the word cloud, say *little*. It will be colored red and also highlighted in the word list on the left.

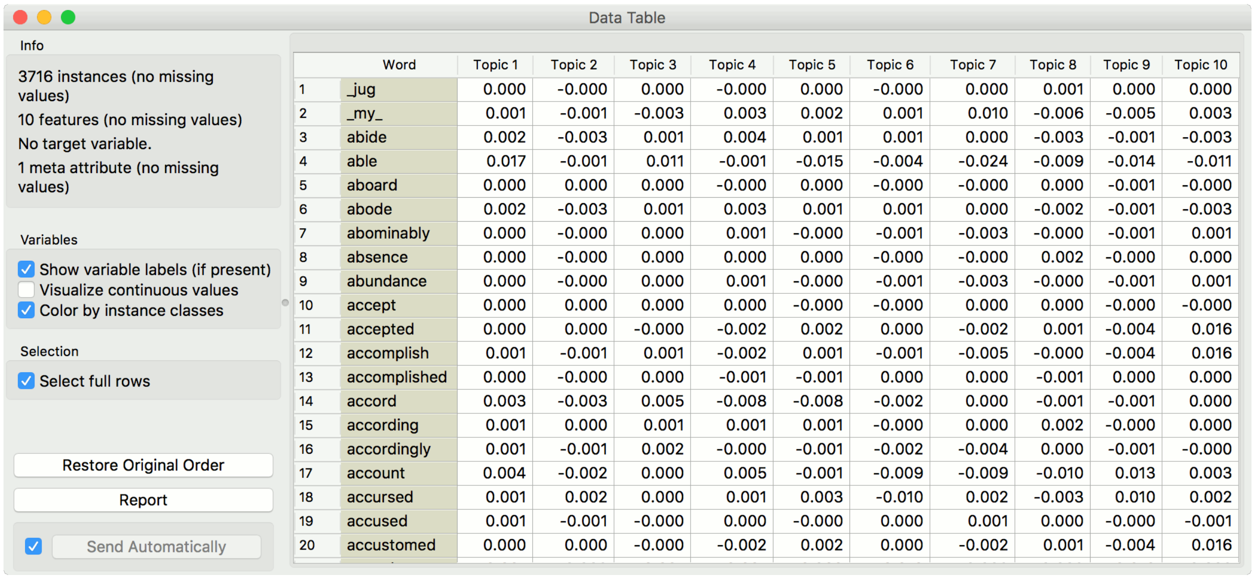
Now we can observe all the documents containing the word *little* in *Corpus Viewer*.

In the second example, we will look at the correlation between topics and words/documents. Connect **Topic Modelling** to **Heat Map**. Ensure the link is set to *All Topics - Data*. **Topic Modelling** will output a matrix of topic weights by words from text (more precisely, tokens).

We can observe the output in a **Data Table**. Tokens are in rows and retrieved topics in colums. Values represent how much a word is represented in a topic.

To visualize this matrix, open **Heat Map**. Select *Merge by k-means* and *Cluster - Rows* to merge similar rows into one and sort them by similarity, which makes the visualization more compact.

In the upper part of the visualization, we have words that highly define topics 1-3 and in the lower part those that define topics 5 and 10.



Data Table

Info

3716 instances (no missing values)

10 features (no missing values)

No target variable.

1 meta attribute (no missing values)

Variables

☑ Show variable labels (if present)

☐ Visualize continuous values

☑ Color by instance classes

Selection

☑ Select full rows

	Word	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
1	_jug	0.000	-0.000	0.000	-0.000	0.000	-0.000	0.000	0.001	0.000	0.000
2	_my_	0.001	-0.001	-0.003	0.003	0.002	0.001	0.010	-0.006	-0.005	0.003
3	abide	0.002	-0.003	0.001	0.004	0.001	0.001	0.000	-0.003	-0.001	-0.003
4	able	0.017	-0.001	0.011	-0.001	-0.015	-0.004	-0.024	-0.009	-0.014	-0.011
5	aboard	0.000	0.000	0.000	-0.000	0.000	-0.000	-0.000	0.000	-0.001	-0.000
6	abode	0.002	-0.003	0.001	0.003	0.001	0.001	0.000	-0.002	-0.001	-0.003
7	abominably	0.000	-0.000	0.000	0.001	-0.000	-0.001	-0.003	-0.000	-0.001	0.001
8	absence	0.000	-0.000	0.000	0.000	0.000	-0.000	-0.000	0.002	-0.000	0.000
9	abundance	0.000	-0.000	0.000	0.001	-0.000	-0.001	-0.003	-0.000	-0.001	0.001
10	accept	0.000	0.000	0.000	0.000	-0.000	-0.000	0.000	0.000	-0.000	-0.000
11	accepted	0.000	0.000	-0.000	-0.002	0.002	0.000	-0.002	0.001	-0.004	0.016
12	accomplish	0.001	-0.001	0.001	-0.002	0.001	-0.001	-0.005	-0.000	-0.004	0.016
13	accomplished	0.000	-0.000	0.000	-0.001	-0.001	0.000	0.000	-0.001	0.000	0.000
14	accord	0.003	-0.003	0.005	-0.008	-0.008	-0.002	0.000	-0.001	-0.001	0.000
15	according	0.001	0.000	0.001	0.001	0.001	-0.000	0.000	0.002	-0.000	0.000
16	accordingly	0.001	-0.001	0.002	-0.000	-0.001	-0.002	-0.004	0.000	-0.001	-0.000
17	account	0.004	-0.002	0.000	0.005	-0.001	-0.009	-0.009	-0.010	0.013	0.003
18	accursed	0.001	0.002	0.000	0.001	0.003	-0.010	0.002	-0.003	0.010	0.002
19	accused	0.001	-0.001	-0.000	0.000	-0.000	0.000	0.001	0.000	-0.000	-0.001
20	accustomed	0.000	0.000	-0.000	-0.002	0.002	0.000	-0.002	0.001	-0.004	0.016

We can similarly observe topic representation across documents. We connect another **Heat Map** to **Topic Modelling** and set link to *Corpus* - *Data*. We set *Merge* and *Cluster* as above.

In this visualization we see how much is a topic represented in a document. Looks like Topic 1 is represented almost across the entire corpus, while other topics are more specific. To observe a specific set of document, select either a clustering node or a row in the visualization. Then pass the data to *Corpus Viewer*.

1.12 Word Enrichment

Word enrichment analysis for selected documents.

1.12.1 Signals

Inputs:

- **Data**

  Corpus instance.

- **Selected Data**

  Selected instances from corpus.

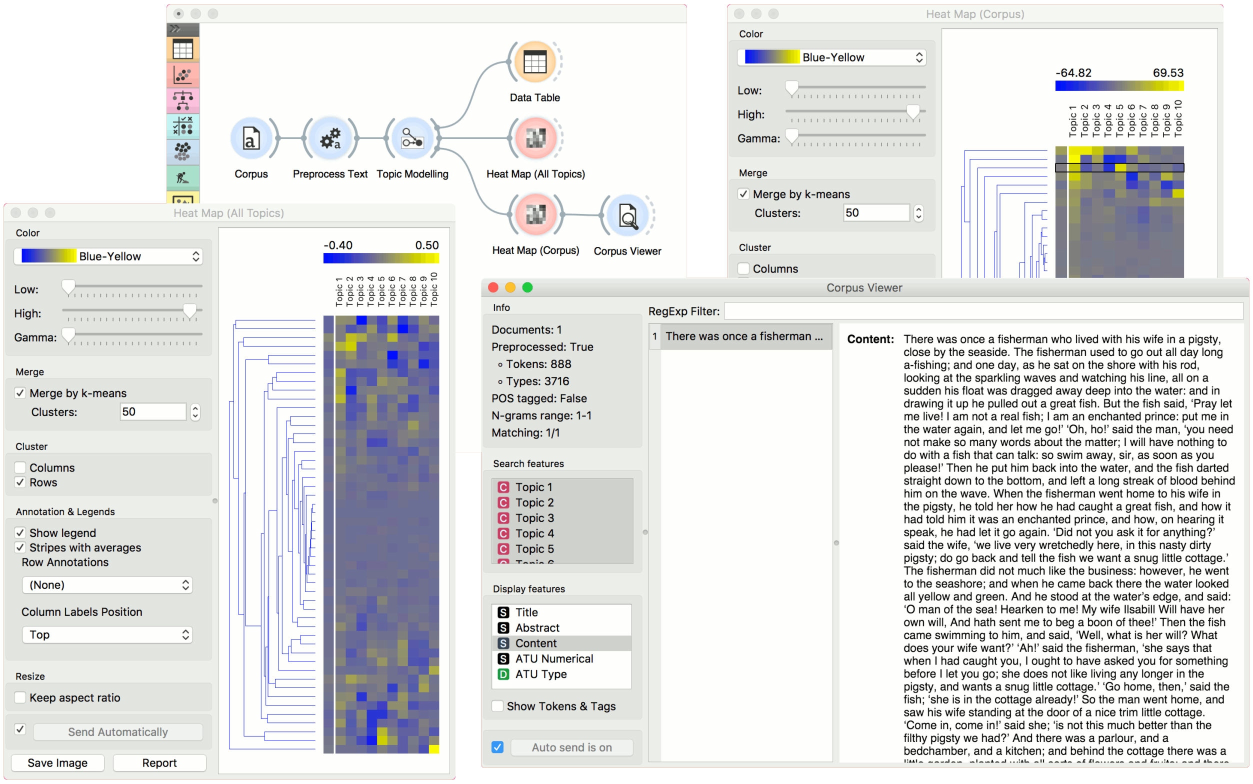
Outputs:

- (None)

1.12.2 Description

Word Enrichment displays a list of words with lower p-values (higher significance) for a selected subset compared to the entire corpus. Lower p-value indicates a higher likelihood that the word is significant for the selected subset (not randomly occurring in a text). FDR (False Discovery Rate) is linked to p-value and reports on the expected percent of false predictions in the set of predictions, meaning it account for false positives in list of low p-values.

1. **Information on the input.**



Heat Map (All Topics)

Color

Blue-Yellow

Low:

High:

Gamma:

Merge

☑ Merge by k-means

Clusters: 50

Cluster

☐ Columns

☑ Rows

Annotation & Legends

☑ Show legend

☑ Stripes with averages

Row Annotations

(None)

Column Labels Position

Top

Resize

☐ Keep aspect ratio

Heat Map (Corpus)

Color

Blue-Yellow

Low:

High:

Gamma:

Merge

☑ Merge by k-means

Clusters: 50

Cluster

☐ Columns

Corpus Viewer

Info

Documents: 1

Preprocessed: True

- Tokens: 888
- Types: 3716

POS tagged: False

N-grams range: 1-1

Matching: 1/1

Search features

- C Topic 1
- C Topic 2
- C Topic 3
- C Topic 4
- C Topic 5

Display features

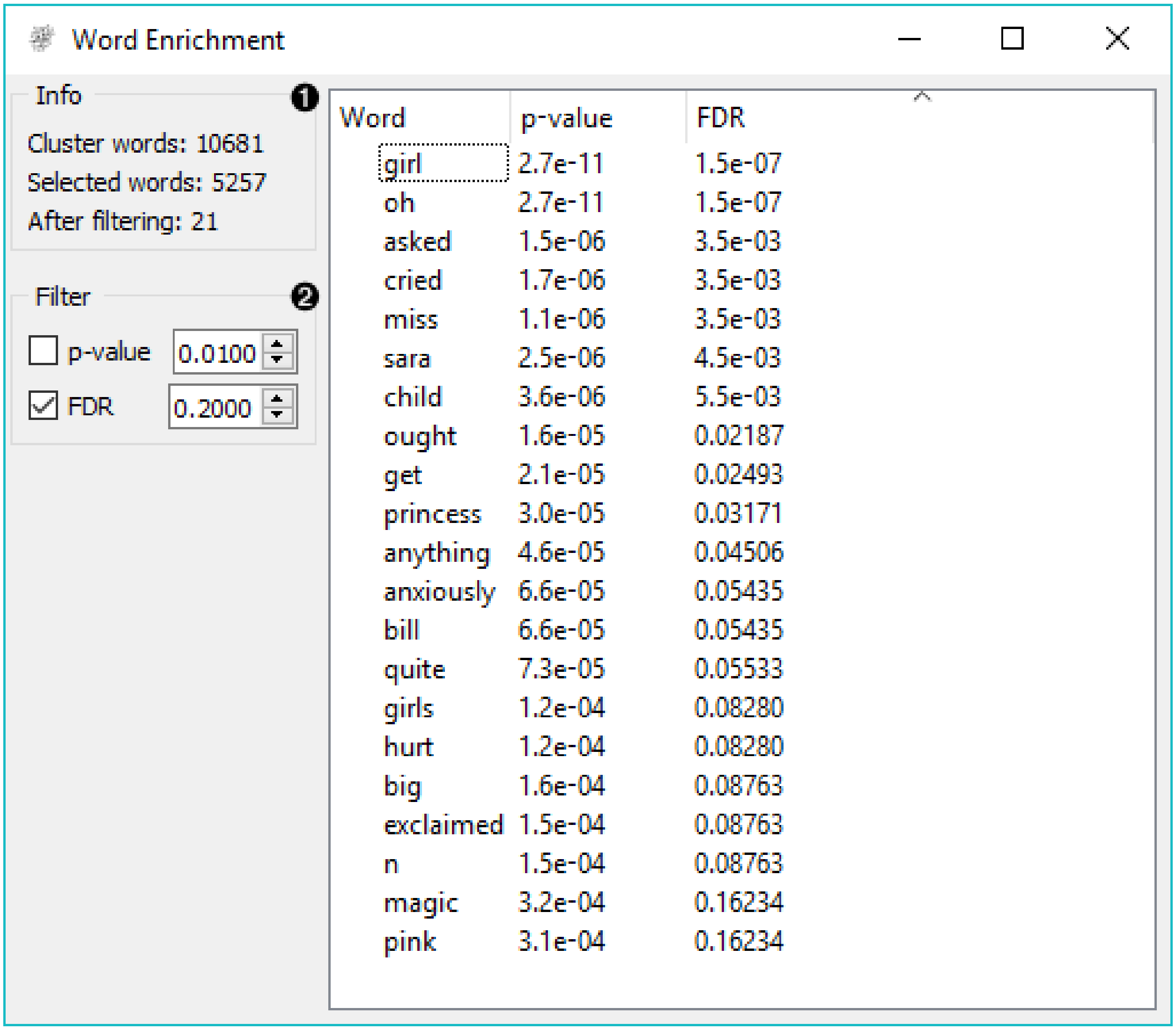
- S Title
- S Abstract
- S Content
- S ATU Numerical
- D ATU Type

☐ Show Tokens & Tags

RegExp Filter:

1 There was once a fisherman ...

Content: There was once a fisherman who lived with his wife in a pigsty, close by the seaside. The fisherman used to go out all day long a-fishing; and one day, as he sat on the shore with his rod, looking at the sparkling waves and watching his line, all on a sudden his float was dragged away deep into the water: and in drawing it up he pulled out a great fish. But the fish said, ‘Pray let me live! I am not a real fish; I am an enchanted prince: put me in the water again, and let me go!’ ‘Oh, ho!’ said the man, ‘you need not make so many words about the matter; I will have nothing to do with a fish that can talk: so swim away, sir, as soon as you please!’ Then he put him back into the water, and the fish darted straight down to the bottom, and left a long streak of blood behind him on the wave. When the fisherman went home to his wife in the pigsty, he told her how he had caught a great fish, and how it had told him it was an enchanted prince, and how, on hearing it speak, he had let it go again. ‘Did not you ask it for anything?’ said the wife, ‘we live very wretchedly here, in this nasty dirty pigsty; do go back and tell the fish we want a snug little cottage.’ The fisherman did not much like the business: however, he went to the seashore; and when he came back there the water looked all yellow and green. And he stood at the water’s edge, and said: ‘O man of the sea! Hearken to me! My wife Ilsabill Will have her own will, And hath sent me to beg a boon of thee!’ Then the fish came swimming to him, and said, ‘Well, what is her will? What does your wife want?’ ‘Ah!’ said the fisherman, ‘she says that when I had caught you, I ought to have asked you for something before I let you go; she does not like living any longer in the pigsty, and wants a snug little cottage.’ ‘Go home, then,’ said the fish; ‘she is in the cottage already!’ So the man went home, and saw his wife standing at the door of a nice trim little cottage. ‘Come in, come in!’ said she; ‘is not this much better than the filthy pigsty we had?’ And there was a parlour, and a bedchamber, and a kitchen; and behind the cottage there was a



Info

Cluster words: 10681

Selected words: 5257

After filtering: 21

Filter

□ p-value 0.0100

☑ FDR 0.2000

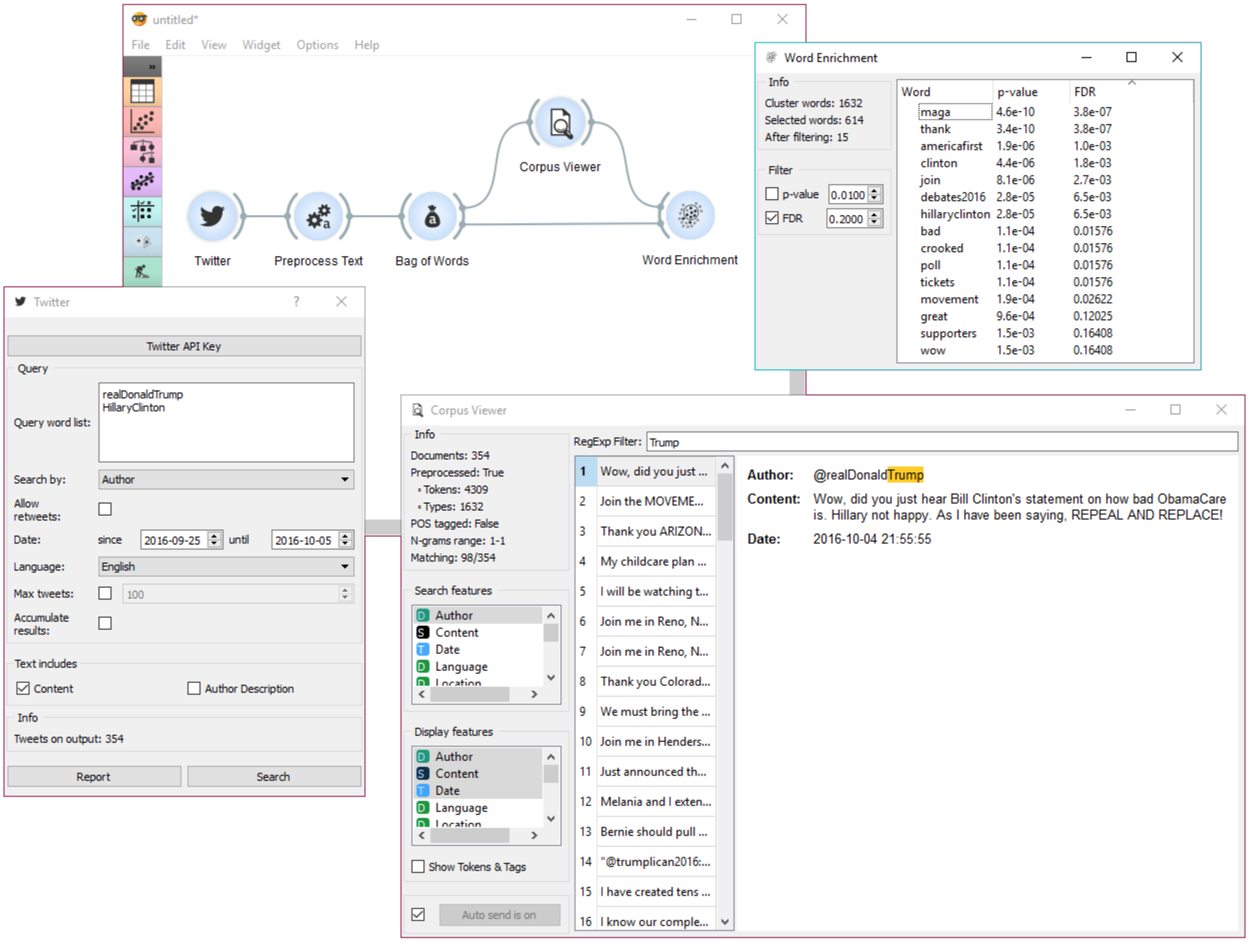
Word	p-value	FDR
girl	2.7e-11	1.5e-07
oh	2.7e-11	1.5e-07
asked	1.5e-06	3.5e-03
cried	1.7e-06	3.5e-03
miss	1.1e-06	3.5e-03
sara	2.5e-06	4.5e-03
child	3.6e-06	5.5e-03
ought	1.6e-05	0.02187
get	2.1e-05	0.02493
princess	3.0e-05	0.03171
anything	4.6e-05	0.04506
anxiously	6.6e-05	0.05435
bill	6.6e-05	0.05435
quite	7.3e-05	0.05533
girls	1.2e-04	0.08280
hurt	1.2e-04	0.08280
big	1.6e-04	0.08763
exclaimed	1.5e-04	0.08763
n	1.5e-04	0.08763
magic	3.2e-04	0.16234
pink	3.1e-04	0.16234

- Cluster words are all the tokens from the corpus.
- Selected words are all the tokens from the selected subset.
- After filtering reports on the enriched words found in the subset.

2. **Filter enables you to filter by:**
   - p-value
   - false discovery rate (FDR)

1.12.3 Example

In the example below, we're retrieved recent tweets from the 2016 presidential candidates, Donald Trump and Hillary Clinton. Then we've preprocessed the tweets to get only words as tokens and to remove the stopwords. We've connected the preprocessed corpus to *Bag of Words* to get a table with word counts for our corpus.



Twitter

Twitter API Key

Query

Query word list: realDonaldTrump
HillaryClinton

Search by: Author

Allow retweets: □

Date: since 2016-09-25 until 2016-10-05

Language: English

Max tweets: □ 100

Accumulate results: □

Text includes

☑ Content □ Author Description

Info

Tweets on output: 354

Report Search

Word Enrichment

Info

Cluster words: 1632
Selected words: 614
After filtering: 15

Filter

□ p-value 0.0100
☑ FDR 0.2000

Word	p-value	FDR
maga	4.6e-10	3.8e-07
thank	3.4e-10	3.8e-07
americafirst	1.9e-06	1.0e-03
clinton	4.4e-06	1.8e-03
join	8.1e-06	2.7e-03
debates2016	2.8e-05	6.5e-03
hillaryclinton	2.8e-05	6.5e-03
bad	1.1e-04	0.01576
crooked	1.1e-04	0.01576
poll	1.1e-04	0.01576
tickets	1.1e-04	0.01576
movement	1.9e-04	0.02622
great	9.6e-04	0.12025
supporters	1.5e-03	0.16408
wow	1.5e-03	0.16408

Corpus Viewer

Info

Documents: 354
Preprocessed: True
- Tokens: 4309
- Types: 1632
POS tagged: False
N-grams range: 1-1
Matching: 98/354

Search features

D Author
S Content
T Date
D Language
D Location

Display features

D Author
S Content
T Date
D Language
D Location

□ Show Tokens & Tags

☑ Auto send is on

RegExp Filter: Trump

1 Wow, did you just ...
2 Join the MOVEME...
3 Thank you ARIZON...
4 My childcare plan ...
5 I will be watching t...
6 Join me in Reno, N...
7 Join me in Reno, N...
8 Thank you Colorad...
9 We must bring the ...
10 Join me in Henders...
11 Just announced th...
12 Melania and I exten...
13 Bernie should pull ...
14 "@trumplican2016:...
15 I have created tens ...
16 I know our comple...

Author: @realDonaldTrump

Content: Wow, did you just hear Bill Clinton's statement on how bad ObamaCare is. Hillary not happy. As I have been saying, REPEAL AND REPLACE!

Date: 2016-10-04 21:55:55

Then we've connected *Corpus Viewer* to **Bag of Words** and selected only those tweets that were published by Donald Trump. See how we marked only the *Author* as our *Search feature* to retrieve those tweets.

Word Enrichment accepts two inputs - the entire corpus to serve as a reference and a selected subset from the corpus to do the enrichment on. First connect **Corpus Viewer** to **Word Enrichment** (input Matching Docs → Selected Data) and then connect **Bag of Words** to it (input Corpus → Data). In the **Word Enrichment** widget we can see the list of words that are more significant for Donald Trump than they are for Hillary Clinton.

1.13 Word Cloud

Generates a word cloud from corpus.

1.13.1 Signals

Inputs:

- **Topic**

  Selected topic.

- **Corpus**

  A *Corpus* instance.

Outputs:

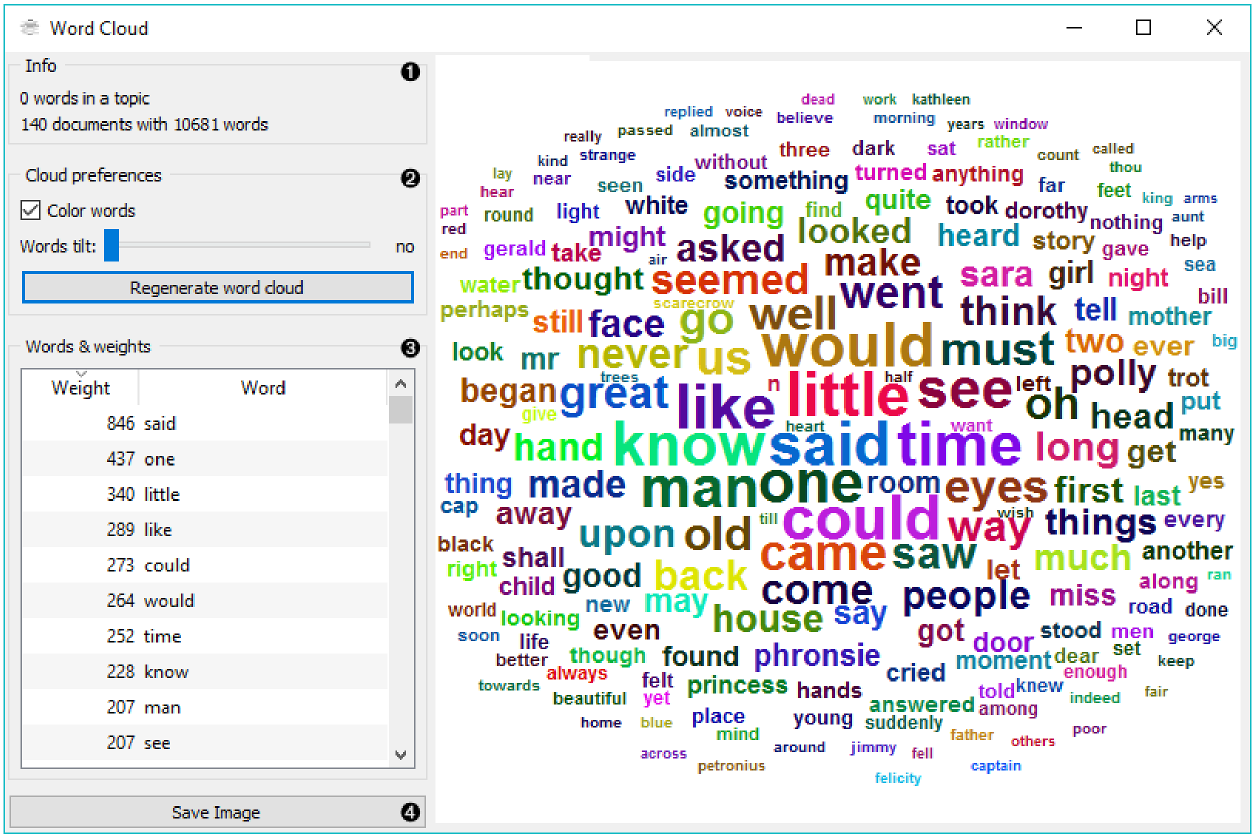
- **Corpus**

  Documents that match the selection.

1.13.2 Description

Word Cloud displays tokens in the corpus, their size denoting the frequency of the word in corpus. Words are listed by their frequency (weight) in the widget. The widget outputs documents, containing selected tokens from the word cloud.

1. **Information on the input.**
    - number of words (tokens) in a topic
    - number of documents and tokens in the corpus
2. **Adjust the plot.**
    - If *Color words* is ticked, words will be assigned a random color. If unchecked, the words will be black.
    - *Word tilt* adjust the tilt of words. The current state of tilt is displayed next to the slider ('no' is the default).
    - *Regenerate word cloud* plot the cloud anew.
3. Words & weights displays a sorted list of words (tokens) by their frequency in the corpus or topic. Clicking on a word will select that same word in the cloud and output matching documents. Use *Ctrl* to select more than one word. Documents matching ANY of the selected words will be on the output (logical OR).
4. *Save Image* saves the image to your computer in a .svg or .png format.



Word Cloud

❶ Info

0 words in a topic

140 documents with 10681 words

❷ Cloud preferences

☑ Color words

Words tilt: no

Regenerate word cloud

❸ Words & weights

Weight	Word
846	said
437	one
340	little
289	like
273	could
264	would
252	time
228	know
207	man
207	see

❹ Save Image

1.13.3 Example

Word Cloud is an excellent widget for displaying the current state of the corpus and for monitoring the effects of preprocessing.

Use *Corpus* to load the data. Connect *Preprocess Text* to it and set your parameters. We've used defaults here, just to see the difference between the default preprocessing in the **Word Cloud** widget and the **Preprocess Text** widget.

We can see from the two widgets, that **Preprocess Text** displays only words, while default preprocessing in the **Word Cloud** tokenizes by word and punctuation.

1.14 GeoMap

Displays geographic distribution of data.

1.14.1 Signals

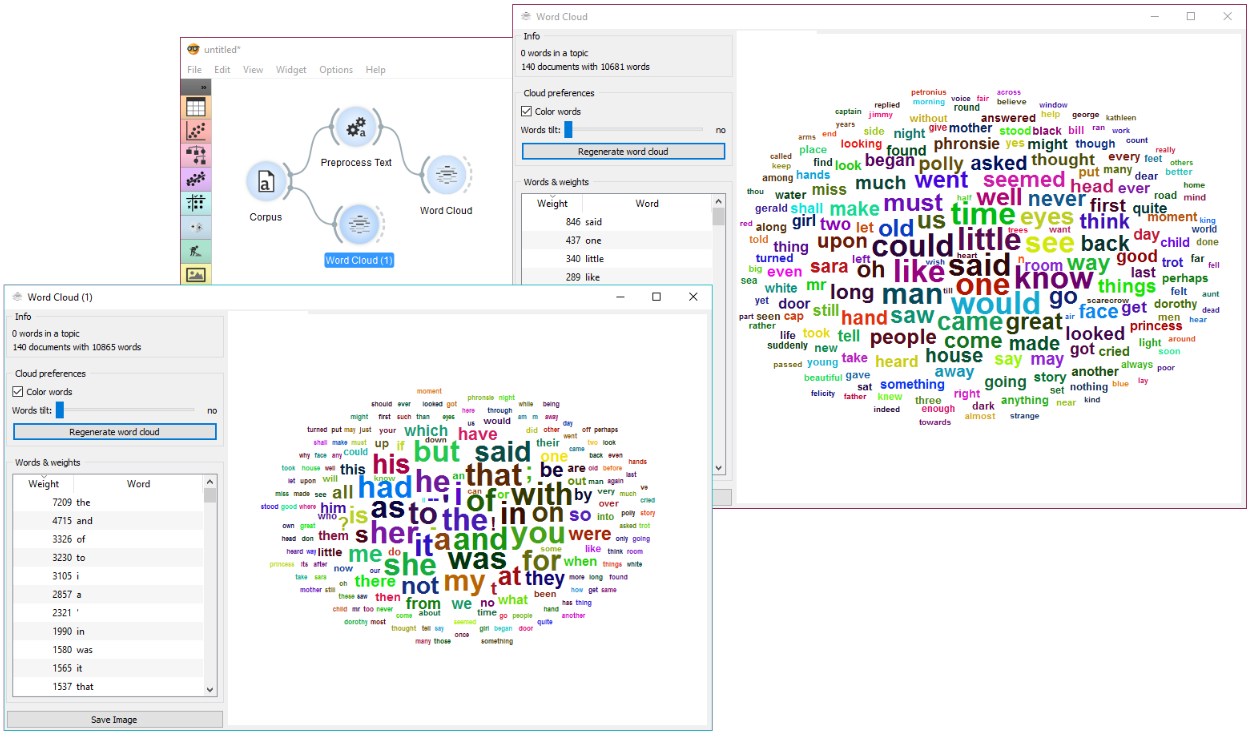
Inputs:

- **Data**

  Data set.

Outputs:

- **Corpus**



Word Cloud

Info
0 words in a topic
140 documents with 10681 words

Cloud preferences
☑ Color words
Words tilt: no
Regenerate word cloud

Words & weights

Weight	Word
846	said
437	one
340	little
289	like

Word Cloud (1)

Info
0 words in a topic
140 documents with 10865 words

Cloud preferences
☑ Color words
Words tilt: no
Regenerate word cloud

Words & weights

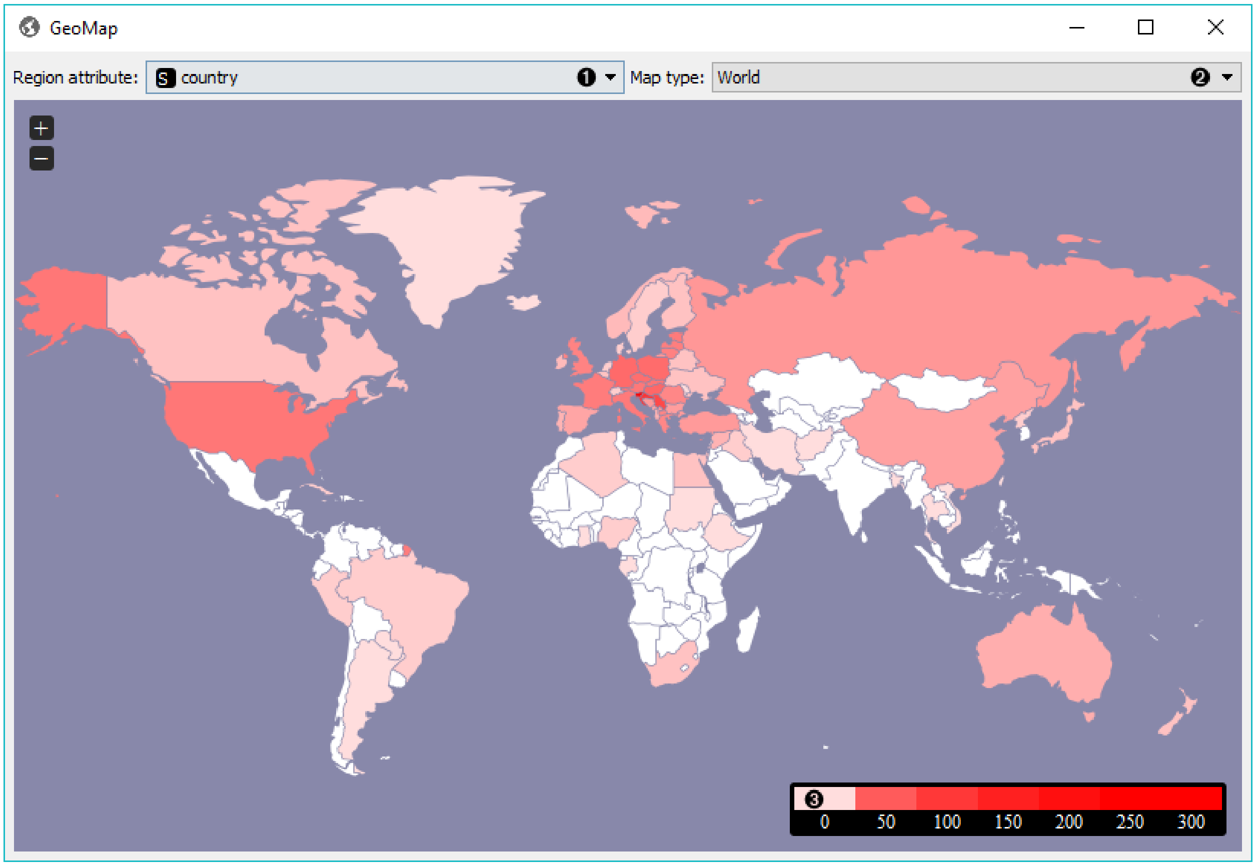
Weight	Word
7209	the
4715	and
3326	of
3230	to
3105	i
2857	a
2321	'
1990	in
1580	was
1565	it
1537	that

Save Image

A *Corpus* instance.

1.14.2 Description

GeoMap widget shows geolocations from textual (string) data. It finds mentions of geographic names (countries and capitals) and displays distributions (frequency of mentiones) of these names on a map. It works with any Orange widget that outputs a data table and that contains at least one string attribute. The widget outputs selected data instances, that is all documents containing mentions of a selected country (or countries).

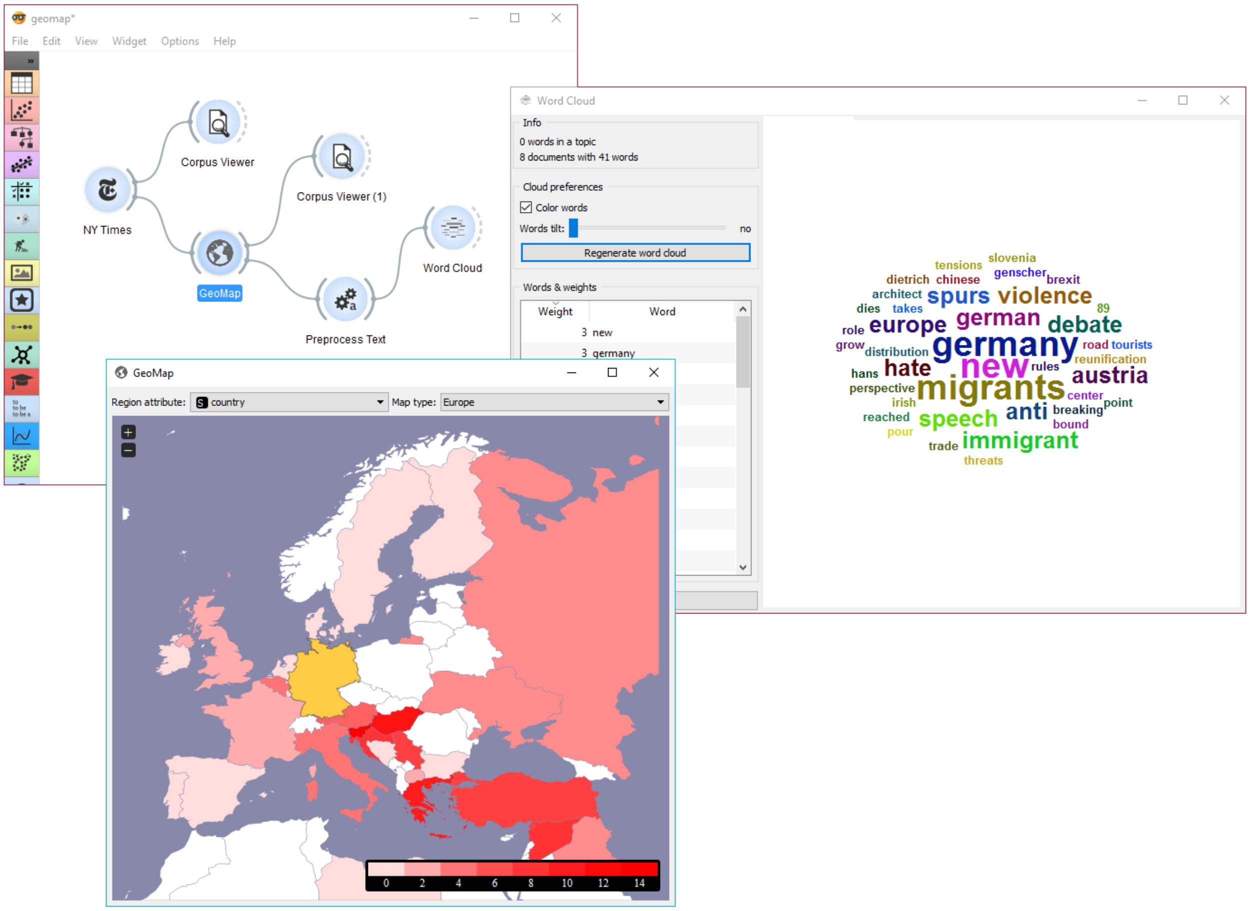


1. Select the meta attribute you want to search geolocations by. The widget will find all mentions of geolocations in a text and display distributions on a map.
2. Select the type of map you wish to display. The options are *World*, *Europe* and *USA*. You can zoom in and out of the map by pressing + and - buttons on a map or by mouse scroll.
3. The legend for the geographic distribution of data. Countries with the boldest color are most often mentioned in the selected region attribute (highest frequency).

To select documents mentioning a specific country, click on a country and the widget will output matching documents. To select more than one country hold Ctrl/Cmd upon selection.

1.14.3 Example

GeoMap widget can be used for simply visualizing distributions of geolocations or for a more complex interactive data analysis. Here, we've queried *NY Times* for articles on Slovenia for the time period of the last year (2015-2016). First we checked the results with *Corpus Viewer*.


geomap*
File Edit View Widget Options Help
Corpus Viewer
NY Times
Corpus Viewer (1)
GeoMap
Word Cloud
Preprocess Text
Word Cloud
GeoMap
0 2 4 6 8 10 12 14
tensions slovenia dietrich chinese genscher brexit architect spurs violence dies takes 89 role europe german debate grow distribution germany road tourists hans hate new rules reunification austria perspective migrants center irish reached speech anti breaking point pour bound trade immigrant threats

Info

0 words in a topic
8 documents with 41 words

Cloud preferences

☑ Color words

Words tilt: no

Regenerate word cloud

Words & weights

Weight	Word
3	new
3	germany

Region attribute: country Map type: Europe

Then we sent the data to **GeoMap** to see distributiosn of geolocations by *country* attribute. The attribute already contains country tags for each article, which is why **NY Times** is great in combinations with **GeoMap**. We selected Germany, which sends all the documents tagged with Germany to the output. Remember, we queried **NY Times** for articles on Slovenia.

We can again inspect the output with **Corpus Viewer**. But there's a more interesting way of visualizing the data. We've sent selected documents to *Preprocess Text*, where we've tokenized text to words and removed stopwords.

Finally, we can inspect the top words appearing in last year's documents on Slovenia and mentioning also Germany with *Word Cloud*.

1.15 Concordance

Display the context of the word.

1.15.1 Signals

Inputs:

- **Corpus**

  A *Corpus* instance.

Outputs:

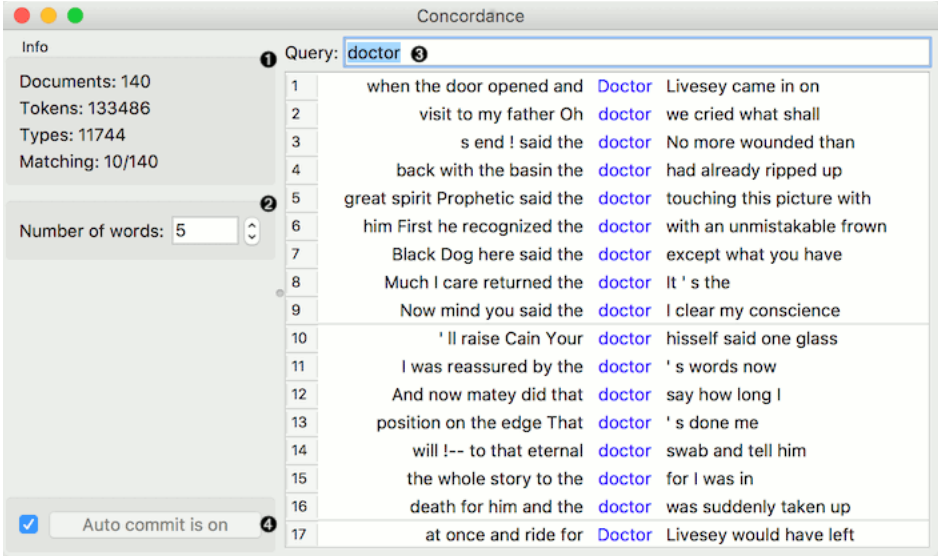
- **Selected Documents**

  A *Corpus* instance.

1.15.2 Description

Concordance finds the queried word in a text and displays the context in which this word is used. It can output selected documents for further analysis.

1. ***Information*:**
    - *Documents*: number of documents on the input.
    - *Tokens*: number of tokens on the input.
    - *Types*: number of unique tokens on the input.
    - *Matching*: number of documents containing the queried word.
2. *Number of words*: the number of words displayed on each side of the queried word.
3. Queried word.
4. If *Auto commit is on*, selected documents are communicated automatically. Alternatively press *Commit*.



Info

Documents: 140
Tokens: 133486
Types: 11744
Matching: 10/140

Number of words: 5

☑ Auto commit is on

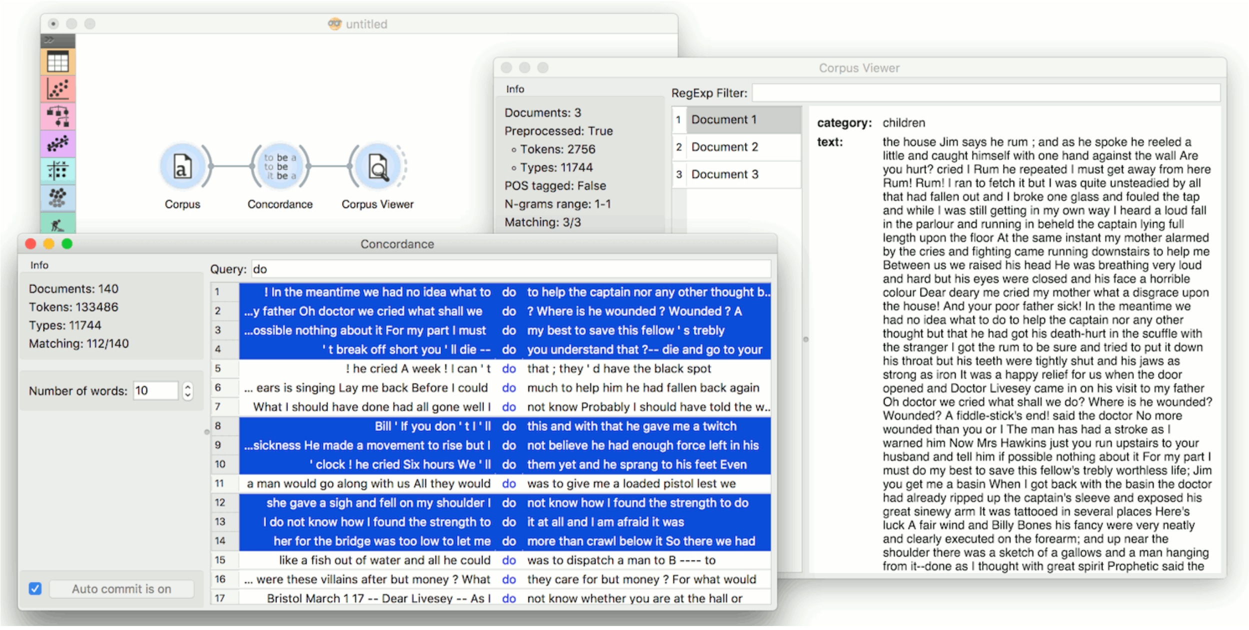
Query: doctor

1	when the door opened and	Doctor	Livesey came in on
2	visit to my father Oh	doctor	we cried what shall
3	s end ! said the	doctor	No more wounded than
4	back with the basin the	doctor	had already ripped up
5	great spirit Prophetic said the	doctor	touching this picture with
6	him First he recognized the	doctor	with an unmistakable frown
7	Black Dog here said the	doctor	except what you have
8	Much I care returned the	doctor	It ' s the
9	Now mind you said the	doctor	I clear my conscience
10	' ll raise Cain Your	doctor	hisself said one glass
11	I was reassured by the	doctor	' s words now
12	And now matey did that	doctor	say how long I
13	position on the edge That	doctor	' s done me
14	will !-- to that eternal	doctor	swab and tell him
15	the whole story to the	doctor	for I was in
16	death for him and the	doctor	was suddenly taken up
17	at once and ride for	Doctor	Livesey would have left

1.15.3 Example

Concordance can be used for displaying word contexts in a corpus. First, we load *book-excerpts.tab* in *Corpus*. Then we connect **Corpus** to **Concordances** and search for concordances of a word "doctor". The widget displays all documents containing the word "doctor" together with their surrounding (contextual) words. Note that the widget finds only exact matches of a word.

Now we can select those documents that contain interesting contexts and output them to *Corpus Viewer* to inspect them further.



Corpus Viewer

Info

Documents: 3
Preprocessed: True
∘ Tokens: 2756
∘ Types: 11744
POS tagged: False
N-grams range: 1-1
Matching: 3/3

RegExp Filter:

1	Document 1
2	Document 2
3	Document 3

category: children

text: the house Jim says he rum ; and as he spoke he reeled a little and caught himself with one hand against the wall Are you hurt? cried I Rum he repeated I must get away from here Rum! Rum! I ran to fetch it but I was quite unsteadied by all that had fallen out and I broke one glass and fouled the tap and while I was still getting in my own way I heard a loud fall in the parlour and running in beheld the captain lying full length upon the floor At the same instant my mother alarmed by the cries and fighting came running downstairs to help me Between us we raised his head He was breathing very loud and hard but his eyes were closed and his face a horrible colour Dear deary me cried my mother what a disgrace upon the house! And your poor father sick! In the meantime we had no idea what to do to help the captain nor any other thought but that he had got his death-hurt in the scuffle with the stranger I got the rum to be sure and tried to put it down his throat but his teeth were tightly shut and his jaws as strong as iron It was a happy relief for us when the door opened and Doctor Livesey came in on his visit to my father Oh doctor we cried what shall we do? Where is he wounded? Wounded? A fiddle-stick's end! said the doctor No more wounded than you or I The man has had a stroke as I warned him Now Mrs Hawkins just you run upstairs to your husband and tell him if possible nothing about it For my part I must do my best to save this fellow's trebly worthless life; Jim you get me a basin When I got back with the basin the doctor had already ripped up the captain's sleeve and exposed his great sinewy arm It was tattooed in several places Here's luck A fair wind and Billy Bones his fancy were very neatly and clearly executed on the forearm; and up near the shoulder there was a sketch of a gallows and a man hanging from it--done as I thought with great spirit Prophetic said the

Concordance

Info

Documents: 140
Tokens: 133486
Types: 11744
Matching: 112/140

Number of words: 10

☑ Auto commit is on

Query: do

1	! In the meantime we had no idea what to	do	to help the captain nor any other thought b..
2	...y father Oh doctor we cried what shall we	do	? Where is he wounded ? Wounded ? A
3	...ossible nothing about it For my part I must	do	my best to save this fellow ' s trebly
4	' t break off short you ' ll die --	do	you understand that ?-- die and go to your
5	! he cried A week ! I can ' t	do	that ; they ' d have the black spot
6	... ears is singing Lay me back Before I could	do	much to help him he had fallen back again
7	What I should have done had all gone well I	do	not know Probably I should have told the w..
8	Bill ' If you don ' t I ' ll	do	this and with that he gave me a twitch
9	...sickness He made a movement to rise but I	do	not believe he had enough force left in his
10	' clock ! he cried Six hours We ' ll	do	them yet and he sprang to his feet Even
11	a man would go along with us All they would	do	was to give me a loaded pistol lest we
12	she gave a sigh and fell on my shoulder I	do	not know how I found the strength to do
13	I do not know how I found the strength to	do	it at all and I am afraid it was
14	her for the bridge was too low to let me	do	more than crawl below it So there we had
15	like a fish out of water and all he could	do	was to dispatch a man to B ---- to
16	... were these villains after but money ? What	do	they care for but money ? For what would
17	Bristol March 1 17 -- Dear Livesey -- As I	do	not know whether you are at the hall or

1.16 Sentiment Analysis

Predict sentiment from text.

1.16.1 Signals

Inputs:

- **Corpus**

  A *Corpus* instance.

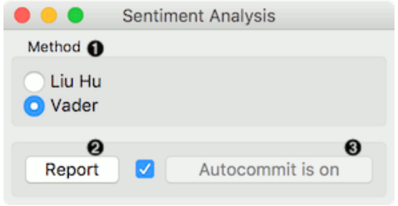
Outputs:

- **Corpus**

  A *Corpus* instance.

1.16.2 Description

Sentiment Analysis predicts sentiment for each document in a corpus. It uses Liu Hu and Vader sentiment modules from NLTK. Both of them are lexicon-based.



Sentiment Analysis

Method ❶

○ Liu Hu

● Vader

❷ Report ☑ Autocommit is on ❸

1. ***Method*:**

   - *Liu Hu*: lexicon-based sentiment analysis
   - *Vader*: lexicon- and rule-based sentiment analysis

2. Produce a report.
3. If *Auto commit is on*, sentiment-tagged corpus is communicated automatically. Alternatively press *Commit*.

1.16.3 Example

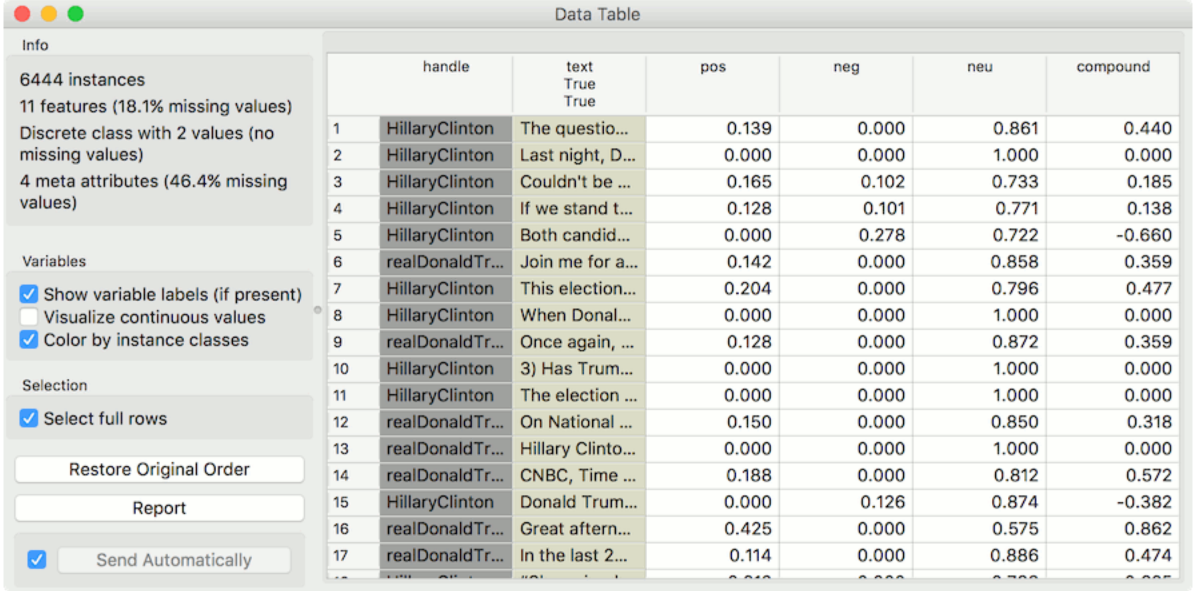
Sentiment Analysis can be used for constructing additional features with sentiment prediction from corpus. First, we load *Election-2016-tweets.tab* in *Corpus*. Then we connect **Corpus** to **Sentiment Analysis**. The widget will append 4 new features for Vader method: positive score, negative score, neutral score and compound (combined score).

We can observe new features in a **Data Table**, where we sorted the *compound* by score. Compound represents the total sentiment of a tweet, where -1 is the most negative and 1 the most positive.

Now let us visualize the data. We have some features we are currently not interested in, so we will remove them with **Select Columns**.

Then we will make our corpus a little smaller, so it will be easier to visualize. Pass the data to **Data Sampler** and retain a random 10% of the tweets.

Now pass the filtered corpus to **Heat Map**. Use *Merge by k-means* to merge tweets with the same polarity into one line. Then use *Cluster* by *rows* to create a clustered visualization where similar tweets are grouped together. Click on a cluster to select a group of tweets - we selected the negative cluster.



Data Table

Info

6444 instances
11 features (18.1% missing values)
Discrete class with 2 values (no missing values)
4 meta attributes (46.4% missing values)

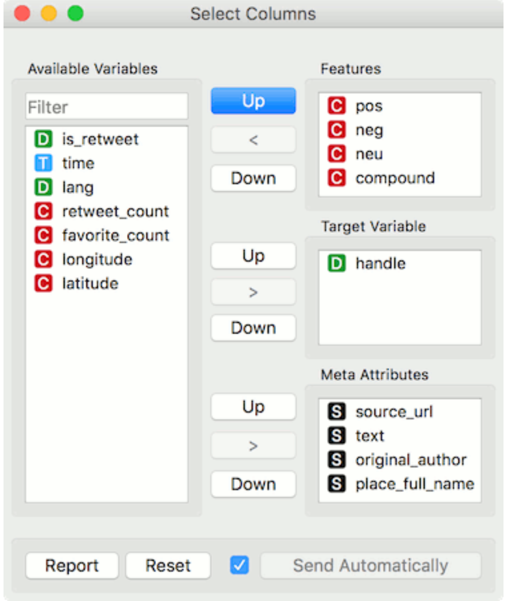
Variables

☑ Show variable labels (if present)
☐ Visualize continuous values
☑ Color by instance classes

Selection

☑ Select full rows

	handle	text True True	pos	neg	neu	compound
1	HillaryClinton	The questio...	0.139	0.000	0.861	0.440
2	HillaryClinton	Last night, D...	0.000	0.000	1.000	0.000
3	HillaryClinton	Couldn't be ...	0.165	0.102	0.733	0.185
4	HillaryClinton	If we stand t...	0.128	0.101	0.771	0.138
5	HillaryClinton	Both candid...	0.000	0.278	0.722	-0.660
6	realDonaldTr...	Join me for a...	0.142	0.000	0.858	0.359
7	HillaryClinton	This election...	0.204	0.000	0.796	0.477
8	HillaryClinton	When Donal...	0.000	0.000	1.000	0.000
9	realDonaldTr...	Once again, ...	0.128	0.000	0.872	0.359
10	HillaryClinton	3) Has Trum...	0.000	0.000	1.000	0.000
11	HillaryClinton	The election ...	0.000	0.000	1.000	0.000
12	realDonaldTr...	On National ...	0.150	0.000	0.850	0.318
13	realDonaldTr...	Hillary Clinto...	0.000	0.000	1.000	0.000
14	realDonaldTr...	CNBC, Time ...	0.188	0.000	0.812	0.572
15	HillaryClinton	Donald Trum...	0.000	0.126	0.874	-0.382
16	realDonaldTr...	Great aftern...	0.425	0.000	0.575	0.862
17	realDonaldTr...	In the last 2...	0.114	0.000	0.886	0.474



Select Columns

Available Variables

Filter

- D is_retweet
- T time
- D lang
- C retweet_count
- C favorite_count
- C longitude
- C latitude

Features

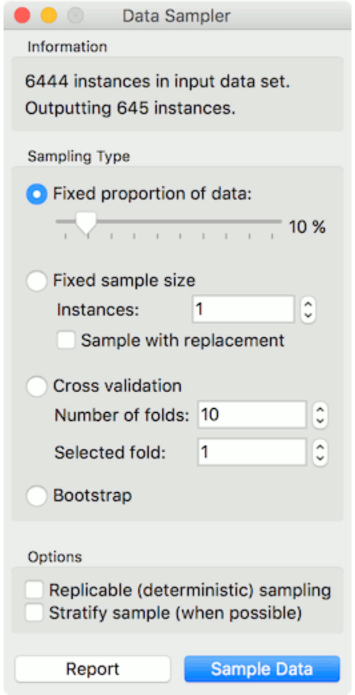
- C pos
- C neg
- C neu
- C compound

Target Variable

- D handle

Meta Attributes

- S source_url
- S text
- S original_author
- S place_full_name



Data Sampler

Information

6444 instances in input data set.
Outputting 645 instances.

Sampling Type

● Fixed proportion of data: 10 %

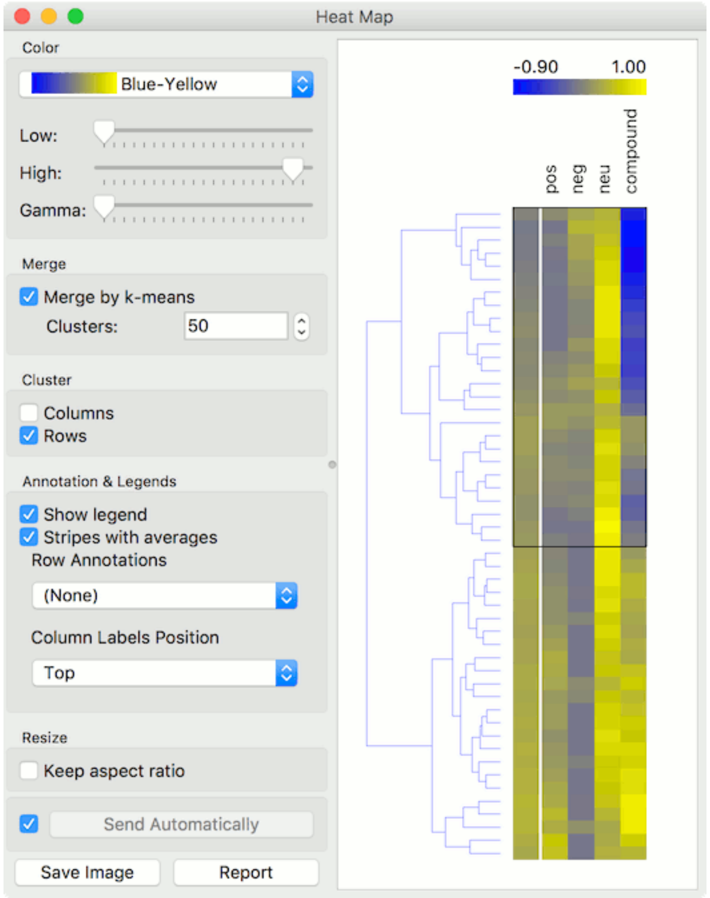
○ Fixed sample size
Instances: 1
□ Sample with replacement

○ Cross validation
Number of folds: 10
Selected fold: 1

○ Bootstrap

Options

□ Replicable (deterministic) sampling
□ Stratify sample (when possible)



Heat Map

Color

Blue-Yellow

Low:
High:
Gamma:

Merge

☑ Merge by k-means
Clusters: 50

Cluster

□ Columns
☑ Rows

Annotation & Legends

☑ Show legend
☑ Stripes with averages
Row Annotations
(None)

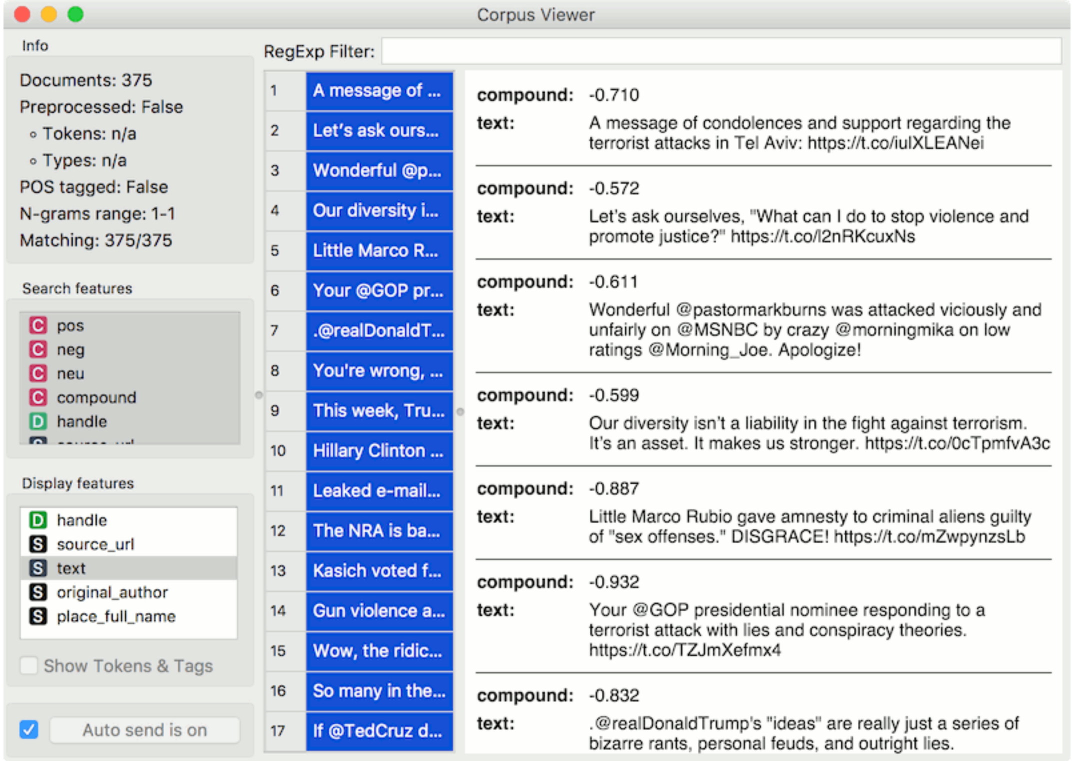
Column Labels Position
Top

Resize

□ Keep aspect ratio

☑ Send Automatically

To observe the selected subset, pass the tweets to *Corpus Viewer*.



Documents: 375
Preprocessed: False
- Tokens: n/a
- Types: n/a
POS tagged: False
N-grams range: 1-1
Matching: 375/375

☑ Auto send is on

compound: -0.710
text: A message of condolences and support regarding the terrorist attacks in Tel Aviv: https://t.co/iulXLEANei

compound: -0.572
text: Let's ask ourselves, "What can I do to stop violence and promote justice?" https://t.co/l2nRKcuxNs

compound: -0.611
text: Wonderful @pastormarkburns was attacked viciously and unfairly on @MSNBC by crazy @morningmika on low ratings @Morning_Joe. Apologize!

compound: -0.599
text: Our diversity isn't a liability in the fight against terrorism. It's an asset. It makes us stronger. https://t.co/0cTpmfvA3c

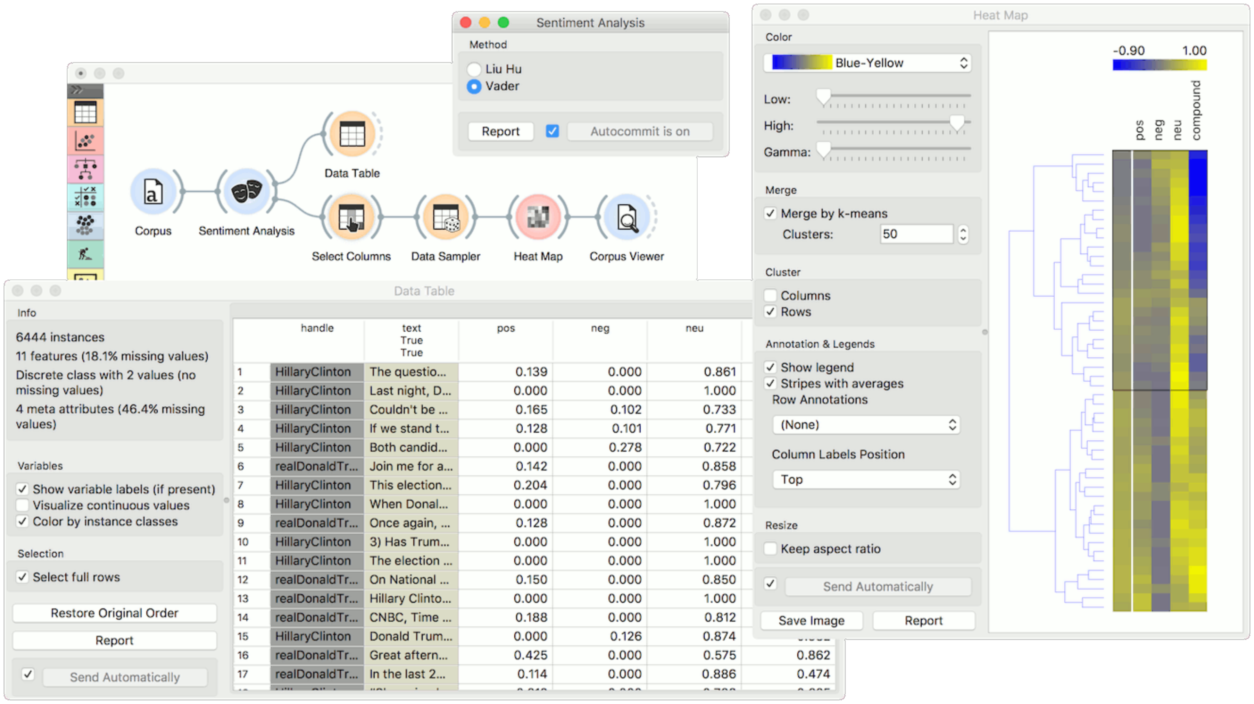
compound: -0.887
text: Little Marco Rubio gave amnesty to criminal aliens guilty of "sex offenses." DISGRACE! https://t.co/mZwpynzsLb

compound: -0.932
text: Your @GOP presidential nominee responding to a terrorist attack with lies and conspiracy theories. https://t.co/TZJmXefmx4

compound: -0.832
text: .@realDonaldTrump's "ideas" are really just a series of bizarre rants, personal feuds, and outright lies.

1.16.4 References

Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.



Sentiment Analysis

Method

○ Liu Hu
● Vader

Report ☑ Autocommit is on

Data Table

Info

6444 instances
11 features (18.1% missing values)
Discrete class with 2 values (no missing values)
4 meta attributes (46.4% missing values)

Variables

☑ Show variable labels (if present)
□ Visualize continuous values
☑ Color by instance classes

Selection

☑ Select full rows

Restore Original Order

Report

☑ Send Automatically

	handle	text True True	pos	neg	neu	
1	HillaryClinton	The questio...	0.139	0.000	0.861	
2	HillaryClinton	Last night, D...	0.000	0.000	1.000	
3	HillaryClinton	Couldn't be ...	0.165	0.102	0.733	
4	HillaryClinton	If we stand t...	0.128	0.101	0.771	
5	HillaryClinton	Both candid...	0.000	0.278	0.722	
6	realDonaldTr...	Join me for a...	0.142	0.000	0.858	
7	HillaryClinton	This election...	0.204	0.000	0.796	
8	HillaryClinton	When Donal...	0.000	0.000	1.000	
9	realDonaldTr...	Once again, ...	0.128	0.000	0.872	
10	HillaryClinton	3) Has Trum...	0.000	0.000	1.000	
11	HillaryClinton	The election ...	0.000	0.000	1.000	
12	realDonaldTr...	On National ...	0.150	0.000	0.850	
13	realDonaldTr...	Hillary Clinto...	0.000	0.000	1.000	
14	realDonaldTr...	CNBC, Time ...	0.188	0.000	0.812	
15	HillaryClinton	Donald Trum...	0.000	0.126	0.874	
16	realDonaldTr...	Great aftern...	0.425	0.000	0.575	0.862
17	realDonaldTr...	In the last 2...	0.114	0.000	0.886	0.474

Heat Map

Color

Blue-Yellow

Low:
High:
Gamma:

Merge

☑ Merge by k-means
Clusters: 50

Cluster

□ Columns
☑ Rows

Annotation & Legends

☑ Show legend
☑ Stripes with averages
Row Annotations
(None)
Column Labels Position
Top

Resize

□ Keep aspect ratio

☑ Send Automatically

Save Image Report

CHAPTER 2

Scripting

2.1 Corpus

class orangecontrib.text.corpus.**Corpus**(*domain=None*, *X=None*, *Y=None*, *metas=None*, *W=None*, *text_features=None*, *ids=None*)

Internal class for storing a corpus.

__init__(*domain=None*, *X=None*, *Y=None*, *metas=None*, *W=None*, *text_features=None*, *ids=None*)

Parameters

- **domain** (*Orange.data.Domain*) – the domain for this Corpus
- **X** (*numpy.ndarray*) – attributes
- **Y** (*numpy.ndarray*) – class variables
- **metas** (*numpy.ndarray*) – meta attributes; e.g. text
- **W** (*numpy.ndarray*) – instance weights
- **text_features** (*list*) – meta attributes that are used for text mining. Infer them if None.
- **ids** (*numpy.ndarray*) – Indices

copy()

Return a copy of the table.

dictionary

corpora.Dictionary – A token to id mapper.

documents

Returns – a list of strings representing documents — created by joining selected text features.

documents_from_features(*feats*)

Parameters **feats** (*list*) – A list fo features to join.

Returns: a list of strings constructed by joining feats.

extend_attributes(*X*, *feature_names*, *feature_values=None*, *compute_values=None*, *var_attrs=None*)

Append features to corpus. If *feature_values* argument is present, features will be Discrete else Continuous.

Parameters

- **X** (*numpy.ndarray or scipy.sparse.csr_matrix*) – Features values to append
- **feature_names** (*list*) – List of string containing feature names
- **feature_values** (*list*) – A list of possible values for Discrete features.
- **compute_values** (*list*) – Compute values for corresponding features.
- **var_attrs** (*dict*) – Additional attributes appended to variable.attributes.

extend_corpus(*metadata*, *Y*)

Append documents to corpus.

Parameters

- **metadata** (*numpy.ndarray*) – Meta data
- **Y** (*numpy.ndarray*) – Class variables

static from_documents(*documents*, *name*, *attributes=None*, *class_vars=None*, *metas=None*, *title_indices=None*)

Create corpus from documents.

Parameters

- **documents** (*list*) – List of documents.
- **name** (*str*) – Name of the corpus
- **attributes** (*list*) – List of tuples (Variable, getter) for attributes.
- **class_vars** (*list*) – List of tuples (Variable, getter) for class vars.
- **metas** (*list*) – List of tuples (Variable, getter) for metas.
- **title_indices** (*list*) – List of indices into domain corresponding to features which will be used as titles.

Returns Corpus.

has_tokens()

Return whether corpus is preprocessed or not.

ngrams

generator – Ngram representations of documents.

static retain_preprocessing(*orig*, *new*, *key=Ellipsis*)

Set preprocessing of 'new' object to match the 'orig' object.

set_text_features(*feats*)

Select which meta-attributes to include when mining text.

Parameters **feats** (*list or None*) – List of text features to include. If None infer them.

store_tokens(*tokens*, *dictionary=None*)

Parameters **tokens** (*list*) – List of lists containing tokens.

titles

Returns a list of titles.

tokens

np.ndarray – A list of lists containing tokens. If tokens are not yet present, run default preprocessor and save tokens.

2.2 Preprocessor

This module provides basic functions to process *Corpus* and extract tokens from documents.

To use preprocessing you should create a corpus:

```
>>> from orangecontrib.text import Corpus
>>> corpus = Corpus.from_file('book-excerpts')
```

And create a `Preprocessor` objects with methods you want:

```
>>> from orangecontrib.text import preprocess
>>> p = preprocess.Preprocessor(transformers=[preprocess.LowercaseTransformer()],
...                             tokenizer=preprocess.WordPunctTokenizer(),
...                             normalizer=preprocess.SnowballStemmer('english'),
...                             filters=[preprocess.StopwordsFilter('english'),
...                                      preprocess.FrequencyFilter(min_df=.1)])
```

Then you can apply you preprocessor to the corpus and access tokens via `tokens` attribute:

```
>>> new_corpus = p(corpus)
>>> new_corpus.tokens[0][:10]
['hous', 'say', ';', 'spoke', 'littl', 'one', 'hand', 'wall', 'hurt', '?']
```

This module defines `default_preprocessor` that will be used to extract tokens from a *Corpus* if no preprocessing was applied yet:

```
>>> from orangecontrib.text import Corpus
>>> corpus = Corpus.from_file('deerwester')
>>> corpus.tokens[0]
['human', 'machine', 'interface', 'for', 'lab', 'abc', 'computer', 'applications']
```

class orangecontrib.text.preprocess.**Preprocessor**(*transformers=None*, *tokenizer=None*, *normalizer=None*, *filters=None*, *ngrams_range=None*, *pos_tagger=None*)

Holds document processing objects.

transformers

List([BaseTransformer] – transforms strings

tokenizer

BaseTokenizer – tokenizes string

normalizer

BaseNormalizer – normalizes tokens

filters

List[BaseTokenFilter] – filters unneeded tokens

__call__(*corpus*, *inplace=True*, *on_progress=None*)

Runs preprocessing over a corpus.

Parameters

- **corpus** (*orangecontrib.text.Corpus*) – A corpus to preprocess.
- **inplace** (*bool*) – Whether to create a new Corpus instance.

set_up()
Called before every __call__. Used for setting up tokenizer & filters.

tear_down()
Called after every __call__. Used for cleaning up tokenizer & filters.

2.3 Twitter

class orangecontrib.text.twitter.**Credentials**(*consumer_key*, *consumer_secret*)
Twitter API credentials.

class orangecontrib.text.twitter.**TwitterAPI**(*credentials*, *on_progress=None*, *should_break=None*, *on_error=None*, *on_rate_limit=None*)
Fetch tweets from the Tweeter API.

Notes

Results across multiple searches are aggregated. To remove tweets form previous searches and only return results from the last search either call *reset* method before searching or provide *collecting=False* argument to search method.

reset()
Removes all downloaded tweets.

search_authors(*authors*, ***, *max_tweets=0*, *collecting=False*)
Search by authors.

Parameters

- **authors** (*list of str*) – A list of authors to search for.
- **max_tweets** (*int*) – If greater than zero limits the number of downloaded tweets.
- **collecting** (*bool*) – Whether to collect results across multiple search calls.

Returns Corpus

search_content(*content*, ***, *max_tweets=0*, *lang=None*, *allow_retweets=True*, *collecting=False*)
Search by content.

Parameters

- **content** (*list of str*) – A list of key words to search for.
- **max_tweets** (*int*) – If greater than zero limits the number of downloaded tweets.
- **lang** (*str*) – A language's code (either ISO 639-1 or ISO 639-3 formats).
- **allow_retweets** (*bool*) – Whether to download retweets.
- **collecting** (*bool*) – Whether to collect results across multiple search calls.

Returns Corpus

2.4 New York Times

class orangecontrib.text.nyt.**NYT**(*api_key*)

Class for fetching records from the NYT API.

__init__(*api_key*)

Parameters **api_key** (*str*) – NY Time API key.

api_key_valid()

Checks whether api key given at initialization is valid.

search(*query*, *date_from=None*, *date_to=None*, *max_docs=None*, *on_progress=None*, *should_break=None*)

Parameters

- **query** (*str*) – Search query.
- **date_from** (*date*) – Start date limit.
- **date_to** (*date*) – End date limit.
- **max_docs** (*int*) – Maximal number of documents returned.
- **on_progress** (*callback*) – Called after every iteration of downloading.
- **should_break** (*callback*) – Callback for breaking the computation before the end. If it evaluates to True, downloading is stopped and document downloaded till now are returned in a Corpus.

Returns Search results.

Return type *Corpus*

2.5 The Guardian

This module fetches data from The Guardian API.

To use first create *TheGuardianCredentials*:

```
>>> from orangecontrib.text.guardian import TheGuardianCredentials
>>> credentials = TheGuardianCredentials('<your-api-key>')
```

Then create *TheGuardianAPI* object and use it for searching:

```
>>> from orangecontrib.text.guardian import TheGuardianAPI
>>> api = TheGuardianAPI(credentials)
>>> corpus = api.search('Slovenia', max_documents=10)
>>> len(corpus)
10
```

class orangecontrib.text.guardian.**TheGuardianCredentials**(*key*)

The Guardian API credentials.

__init__(*key*)

Parameters **key** (*str*) – The Guardian API key. Use *test* for testing purposes.

valid

Check if given API key is valid.

class orangecontrib.text.guardian.**TheGuardianAPI**(*credentials*, *on_progress=None*, *should_break=None*)

__init__(*credentials*, *on_progress=None*, *should_break=None*)

Parameters

- **credentials** (*TheGuardianCredentials*) – The Guardian Creentials.
- **on_progress** (*callable*) – Function for progress reporting.
- **should_break** (*callable*) – Function for early stopping.

search(*query*, *from_date=None*, *to_date=None*, *max_documents=None*, *accumulate=False*)
Search The Guardian API for articles.

Parameters

- **query** (*str*) – A query for searching the articles by
- **from_date** (*str*) – Search only articles newer than the date provided. Date should be in ISO format; e.g. '2016-12-31'.
- **to_date** (*str*) – Search only articles older than the date provided. Date should be in ISO format; e.g. '2016-12-31'.
- **max_documents** (*int*) – Maximum number of documents to retrieve. When not given, retrieve all documents.
- **accumulate** (*bool*) – A flag indicating whether to accumulate results of multiple consequent search calls.

Returns *Corpus*

2.6 Wikipedia

class orangecontrib.text.wikipedia.**WikipediaAPI**(*on_error=None*)
Wraps Wikipedia API.

Examples

```
>>> api = WikipediaAPI()
>>> corpus = api.search('en', ['Barack Obama', 'Hillary Clinton'])
```

search(*lang*, *queries*, *articles_per_query=10*, *should_break=None*, *on_progress=None*)
Searches for articles.

Parameters

- **lang** (*str*) – A language code in ISO 639-1 format.
- **queries** (*list of str*) – A list of queries.
- **should_break** (*callback*) – Callback for breaking the computation before the end. If it evaluates to True, downloading is stopped and document downloaded till now are returned in a Corpus.
- **on_progress** (*callable*) – Callback for progress bar.

2.7 Topic Modeling

class orangecontrib.text.topics.**LdaWrapper**(***kwargs*)

fit(*corpus*, ***kwargs*)
Train the model with the corpus.

Parameters **corpus** (Corpus) – A corpus to learn topics from.

transform(*corpus*)
Create a table with topics representation.

class orangecontrib.text.topics.**LsiWrapper**(***kwargs*)

fit(*corpus*, ***kwargs*)
Train the model with the corpus.

Parameters **corpus** (Corpus) – A corpus to learn topics from.

transform(*corpus*)
Create a table with topics representation.

class orangecontrib.text.topics.**HdpWrapper**(***kwargs*)

fit(*corpus*, ***kwargs*)
Train the model with the corpus.

Parameters **corpus** (Corpus) – A corpus to learn topics from.

transform(*corpus*)
Create a table with topics representation.

2.8 Tag

A module for tagging *Corpus* instances.

This module provides a default *pos_tagger* that can be used for POSTagging an English corpus:

```
>>> from orangecontrib.text.corpus import Corpus
>>> from orangecontrib.text.tag import pos_tagger
>>> corpus = Corpus.from_file('deerwester.tab')
>>> tagged_corpus = pos_tagger.tag_corpus(corpus)
>>> tagged_corpus.pos_tags[0]  # you can use `pos_tags` attribute to access tags␣
↪directly
['JJ', 'NN', 'NN', 'IN', 'NN', 'NN', 'NN', 'NNS']
>>> next(tagged_corpus.ngrams_iterator(include_postags=True))  # or `ngrams_iterator`␣
↪to iterate over documents
['human_JJ', 'machine_NN', 'interface_NN', 'for_IN', 'lab_NN', 'abc_NN', 'computer_NN
↪', 'applications_NNS']
```

class orangecontrib.text.tag.**POSTagger**(*tagger*, *name='POS Tagger'*)
A class that wraps *nltk.TaggerI* and performs Corpus tagging.

tag_corpus(*corpus*, ***kwargs*)
Marks tokens of a corpus with POS tags.

Parameters **corpus** (orangecontrib.text.corpus.Corpus) – A corpus instance.

class orangecontrib.text.tag.**StanfordPOSTagger**(**args*, ***kwargs*)

classmethod check(*path_to_model*, *path_to_jar*)

Checks whether provided *path_to_model* and *path_to_jar* are valid.

Raises ValueError – in case at least one of the paths is invalid.

Notes

Can raise an exception if Java Development Kit is not installed or not properly configured.

Examples

```
>>> try:
...     StanfordPOSTagger.check('path/to/model', 'path/to/stanford.jar')
... except ValueError as e:
...     print(e)
Could not find stanford-postagger.jar jar file at path/to/stanford.jar
```

2.9 Async Module

Helper utils for Orange GUI programming.

Provides asynchronous() decorator for making methods calls in async mode. Once method is decorated it will have task.on_start(), task.on_result() and task.callback() decorators for callbacks wrapping.

- *on_start* must take no arguments
- *on_result* must accept one argument (the result)
- *callback* can accept any arguments

For instance:

```
class Widget(QObject):
    def __init__(self, name):
        super().__init__()
        self.name = name

    @asynchronous
    def task(self):
        for i in range(3):
            time.sleep(0.5)
            self.report_progress(i)
        return 'Done'

    @task.on_start
    def report_start(self):
        print('`{}` started'.format(self.name))

    @task.on_result
    def report_result(self, result):
        print('`{}` result: {}'.format(self.name, result))
    @task.callback
    def report_progress(self, i):
        print('`{}` progress: {}'.format(self.name, i))
```

Calling an asynchronous method will launch a daemon thread:

```
first = Widget(name='First')
first.task()
second = Widget(name='Second')
second.task()

first.task.join()
second.task.join()
```

A possible output:

```
`First` started
`Second` started
`Second` progress: 0
`First` progress: 0
`First` progress: 1
`Second` progress: 1
`First` progress: 2
`First` result: Done
`Second` progress: 2
`Second` result: Done
```

In order to terminate a thread either call `stop()` method or raise `StopExecution` exception within `task()`:

```
first.task.stop()
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index