```bash
rasa visualize
```

Uso del comando

```bash
usage: rasa visualize [-h] [-v] [-vv] [--quiet]
                      [--logging-config-file LOGGING_CONFIG_FILE] [-d DOMAIN]
                      [-s STORIES] [--out OUT] [--max-history MAX_HISTORY]
                      [-u NLU]

options:
  -h, --help            show this help message and exit
  -d DOMAIN, --domain DOMAIN
                        Domain specification. This can be a single YAML file,
                        or a directory that contains several files with domain
                        specifications in it. The content of these files will
                        be read and merged together. (default: domain.yml)
  -s STORIES, --stories STORIES
                        File or folder containing your training stories.
                        (default: data)
  --out OUT             Filename of the output path, e.g. 'graph.html'.
                        (default: graph.html)
  --max-history MAX_HISTORY
                        Max history to consider when merging paths in the
                        output graph. (default: 2)
  -u NLU, --nlu NLU     File or folder containing your NLU data, used to
                        insert example messages into the graph. (default:
                        None)

Python Logging Options:
  You can control level of log messages printed. In addition to these
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (default:
                        None)
  -vv, --debug          Print lots of debugging statements. Sets logging level
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (default:
                        None)
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configuration file
                        will be set to the given name. (default: None)
```

Este comando generará un gráfico que representa todas las rutas posibles en las historias de tu asistente y lo guardará en el archivo especificado. Este archivo puede abrirse con cualquier visor de imágenes.

rasa test#

To evaluate a model on your test data, run:

Copy

```
rasa test
```

This will test your latest trained model on any end-to-end stories you have defined in files with the `test_` prefix. If you want to use a different model, you can specify it using the `--model` flag.

To evaluate the dialogue and NLU models separately, use the commands below:

Copy

```
rasa test core
```

and

Copy

```
rasa test nlu
```

You can find more details on specific arguments for each testing type in Evaluating an NLU Model and Evaluating a Dialogue Management Model.

The following arguments are available for `rasa test`:

```
usage: rasa test [-h] [-v] [-vv] [--quiet]
            [--logging-config-file LOGGING_CONFIG_FILE] [-m MODEL]
            [-s STORIES] [--max-stories MAX_STORIES]
            [--endpoints ENDPOINTS] [--fail-on-prediction-errors]
            [--url URL] [--evaluate-model-directory] [-u NLU]
            [-c CONFIG [CONFIG ...]] [-d DOMAIN] [--cross-validation]
            [-f FOLDS] [-r RUNS] [-p PERCENTAGES [PERCENTAGES ...]]
            [--no-plot] [--successes] [--no-errors] [--no-warnings]
            [--out OUT]
            {core,nlu} ...
```

```

positional arguments:
 {core,nlu}
  core              Tests Rasa Core models using your test stories.
  nlu               Tests Rasa NLU models using your test NLU data.
```

```
options:
 -h, --help            show this help message and exit
 -m MODEL, --model MODEL
                       Path to a trained Rasa model. If a directory is
                       specified, it will use the latest model in this
                       directory. (default: models)
 --no-plot             Don't render evaluation plots. (default: False)
 --successes           If set successful predictions will be written to a
                       file. (default: False)
 --no-errors           If set incorrect predictions will NOT be written to a
                       file. (default: False)
 --no-warnings         If set prediction warnings will NOT be written to a
                       file. (default: False)
 --out OUT             Output path for any files created during the
                       evaluation. (default: results)
```

```
Python Logging Options:
 You can control level of log messages printed. In addition to these
 arguments, a more fine grained configuration can be achieved with
 environment variables. See online documentation for more info.
```

```
 -v, --verbose         Be verbose. Sets logging level to INFO. (default:
                       None)
 -vv, --debug          Print lots of debugging statements. Sets logging level
                       to DEBUG. (default: None)
 --quiet               Be quiet! Sets logging level to WARNING. (default:
                       None)
 --logging-config-file LOGGING_CONFIG_FILE
                       If set, the name of the logging configuration file
                       will be set to the given name. (default: None)
```

```
Core Test Arguments:
 -s STORIES, --stories STORIES
                       File or folder containing your test stories. (default:
                       .)
 --max-stories MAX_STORIES
                       Maximum number of stories to test on. (default: None)
 --endpoints ENDPOINTS
                       Configuration file for the connectors as a yml file.
                       (default: endpoints.yml)
 --fail-on-prediction-errors
                       If a prediction error is encountered, an exception is
                       thrown. This can be used to validate stories during
```

```
                        tests, e.g. on travis. (default: False)
 --url URL              If supplied, downloads a story file from a URL and
                        trains on it. Fetches the data by sending a GET
                        request to the supplied URL. (default: None)
 --evaluate-model-directory
                        Should be set to evaluate models trained via 'rasa
                        train core --config <config-1> <config-2>'. All models
                        in the provided directory are evaluated and compared
                        against each other. (default: False)
```

```

NLU Test Arguments:
 -u NLU, --nlu NLU      File or folder containing your NLU data. (default:
                        data)
 -c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]
                        Model configuration file. If a single file is passed
                        and cross validation mode is chosen, cross-validation
                        is performed, if multiple configs or a folder of
                        configs are passed, models will be trained and
                        compared directly. (default: None)
 -d DOMAIN, --domain DOMAIN
                        Domain specification. This can be a single YAML file,
                        or a directory that contains several files with domain
                        specifications in it. The content of these files will
                        be read and merged together. (default: domain.yml)
```

rasa data split#

To create a train-test split of your NLU training data, run:

Copy

```
rasa data split nlu
```

This will create a 80/20 split of train/test data by default. You can specify the training data, the fraction, and the output directory using the following arguments:

```
usage: rasa data split nlu [-h] [-v] [-vv] [--quiet]
                           [--logging-config-file LOGGING_CONFIG_FILE]
                           [-u NLU] [--training-fraction TRAINING_FRACTION]
                           [--random-seed RANDOM_SEED] [--out OUT]
```

```
options:
  -h, --help            show this help message and exit
  -u NLU, --nlu NLU     File or folder containing your NLU data. (default:
                        data)
  --training-fraction TRAINING_FRACTION
                        Percentage of the data which should be in the training
                        data. (default: 0.8)
  --random-seed RANDOM_SEED
                        Seed to generate the same train/test split. (default:
                        None)
  --out OUT             Directory where the split files should be stored.
                        (default: train_test_split)

Python Logging Options:
  You can control level of log messages printed. In addition to these
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (default:
                        None)
  -vv, --debug          Print lots of debugging statements. Sets logging level
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (default:
                        None)
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configuration file
                        will be set to the given name. (default: None)
```

If you have NLG data for retrieval actions, this will be saved to separate files:

```
ls train_test_split

      nlg_test_data.yml     test_data.yml
      nlg_training_data.yml training_data.yml
```

To split your stories, you can use the following command:

rasa data split stories

It has the same arguments as `split nlu` command, but loads yaml files with stories and perform random splitting. Directory `train_test_split` will contain all yaml files processed with prefixes `train_` or `test_` containing train and test parts.

rasa data convert nlu#

You can convert NLU data from

- LUIS data format,
- WIT data format,
- Dialogflow data format, or
- JSON

to

- YAML or
- JSON

You can start the converter by running:
Copy

```
rasa data convert nlu
```

You can specify the input file or directory, output file or directory, and the output format with the following arguments:

```
usage: rasa data convert nlu [-h] [-v] [-vv] [--quiet]
                             [--logging-config-file LOGGING_CONFIG_FILE]
                             [-f {json,yaml}] [--data DATA [DATA ...]]
                             [--out OUT] [-l LANGUAGE]

options:
  -h, --help            show this help message and exit
  -f {json,yaml}, --format {json,yaml}
                        Output format the training data should be converted
                        into. (default: yaml)
  --data DATA [DATA ...]
                        Paths to the files or directories containing Rasa NLU
```

```
                      data. (default: data)
  --out OUT             File (for `json`) or existing path (for `yaml`) where
                      to save training data in Rasa format. (default:
                      converted_data)
  -l LANGUAGE, --language LANGUAGE
                      Language of data. (default: en)

Python Logging Options:
  You can control level of log messages printed. In addition to these
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (default:
                      None)
  -vv, --debug          Print lots of debugging statements. Sets logging level
                      to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (default:
                      None)
  --logging-config-file LOGGING_CONFIG_FILE
                      If set, the name of the logging configuration file
                      will be set to the given name. (default: None)
```

rasa data migrate#

The domain is the only data file whose format changed between 2.0 and 3.0. You can automatically migrate a 2.0 domain to the 3.0 format.

You can start the migration by running:
Copy

```
rasa data migrate
```

You can specify the input file or directory and the output file or directory with the following arguments:
Copy

```
rasa data migrate -d DOMAIN --out OUT_PATH
```

If no arguments are specified, the default domain path (`domain.yml`) will be used for both input and output files.

This command will also back-up your 2.0 domain file(s) into a different `original_domain.yml` file or directory labeled `original_domain`.

Note that the slots in the migrated domain will contain mapping conditions if these slots are part of a form's `required_slots`.

caution

Exceptions will be raised and the migration process terminated if invalid domain files are provided or if they are already in the 3.0 format, if slots or forms are missing from your original files or if the slots or forms sections are spread across multiple domain files. This is done to avoid duplication of migrated sections in your domain files. Please make sure all your slots' or forms' definitions are grouped into a single file.

You can learn more about this command by running:
Copy

```
rasa data migrate --help
```

rasa data validate#

You can check your domain, NLU data, or story data for mistakes and inconsistencies. To validate your data, run this command:
Copy

```
rasa data validate
```

The validator searches for errors in the data, e.g. two intents that have some identical training examples. The validator also checks if you have any stories where different assistant actions follow from the same dialogue history. Conflicts between stories will prevent a model from learning the correct pattern for a dialogue.

Searching for the `assistant_id` key introduced in 3.5

The validator will check whether the `assistant_id` key is present in the config file and will issue a warning if this key is missing or if the default value has not been changed.

If you pass a `max_history` value to one or more policies in your `config.yml` file, provide the smallest of those values in the validator command using the `--max-history <max_history>` flag.

You can also validate only the story structure by running this command:
Copy

```
rasa data validate stories
```

note

Running `rasa data validate` does not test if your rules are consistent with your stories. However, during training, the `RulePolicy` checks for conflicts between rules and stories. Any such conflict will abort training.

Also, if you use end-to-end stories, then this might not capture all conflicts. Specifically, if two user inputs result in different tokens yet exactly the same featurization, then conflicting actions after these inputs may exist but will not be reported by the tool.

To interrupt validation even for minor issues such as unused intents or responses, use the `--fail-on-warnings` flag.

check your story names

The `rasa data validate stories` command assumes that all your story names are unique!

You can use `rasa data validate` with additional arguments, e.g. to specify the location of your data and domain files:

```
usage: rasa data validate [-h] [-v] [-vv] [--quiet]
                          [--logging-config-file LOGGING_CONFIG_FILE]
                          [--max-history MAX_HISTORY] [-c CONFIG]
                          [--fail-on-warnings] [-d DOMAIN]
                          [--data DATA [DATA ...]]
                          {stories} ...

positional arguments:
  {stories}
    stories             Checks for inconsistencies in the story files.

options:
  -h, --help            show this help message and exit
  --max-history MAX_HISTORY
                        Number of turns taken into account for story structure
                        validation. (default: None)
  -c CONFIG, --config CONFIG
                        The policy and NLU pipeline configuration of your bot.
                        (default: config.yml)
  --fail-on-warnings    Fail validation on warnings and errors. If omitted
                        only errors will result in a non zero exit code.
                        (default: False)
  -d DOMAIN, --domain DOMAIN
                        Domain specification. This can be a single YAML file,
                        or a directory that contains several files with domain
                        specifications in it. The content of these files will
                        be read and merged together. (default: domain.yml)
  --data DATA [DATA ...]
                        Paths to the files or directories containing Rasa
                        data. (default: data)

Python Logging Options:
  You can control level of log messages printed. In addition to these
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (default:
                        None)
  -vv, --debug          Print lots of debugging statements. Sets logging level
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (default:
                        None)
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configuration file
                        will be set to the given name. (default: None)
```

rasa export#

To export events from a tracker store using an event broker, run:
Copy

```
rasa export
```