Quoll Documentation

Florian Kunneman, Maarten van Gompel

May 29, 2018

Table of Contents

Quoll is the name of carnivorous marsupials living in Australia, New Guinea and Tazmania. It is also a python library for running NLP pipelines, based on the LuigiNLP workflow system. Quoll takes care of the sequence of tasks that are common to basic Machine Learning experiments with textual input: preprocessing, feature extraction, vectorizing, classification and evaluation. Provided that you prepare instances and labels, all these tasks can be ran as a pipeline, in one go. Quoll is built on top of several applications in the LaMachine Software distribution (Ucto, Frog, Colibri-core, PyNLPl), as well as popular python packages (Numpy, Scipy and Scikit-learn).

Quoll has the following advantages:

- Can run full supervised machine learning pipeline with one command.
- Stores intermediate output of the pipeline.
- Maintains a full log of your experiments.
- Offers various options at each stage of the pipeline.
- If part of the pipeline is already completed, will continue from that point.
- Experiments with different settings can be distinguished based on filenames.

CHAPTER 1

The Featurize module

The Featurize module is the second module in the pipeline, taking care of feature extraction from the output of the Preprocess module. It makes use of ColibriCore to count features, and its output forms the input to the Vectorize module.

1.1 Input

If the input to Preprocess_ (.txt or .txtdir) is given as inputfile, this module is ran prior to the Featurize module.

--inputfile

- The featurize module takes preprocessed documents as input. They can come in four formats:

1. Extension **.tok.txt** - File with tokenized text documents on each line.
2. Extension **.tok.txtdir** - Directory with tokenized text documents (files ending with **.tok.txt**).
3. Extension **.frog.json** - File with frogged text documents.
4. Extension **.frog.jsondir** - Directory with frogged text documents (files ending with **.frog.json**).

1.2 Options

--featuretypes

- Specify the types of features to extract
- Options: **tokens**, **lemmas**, **pos**
- lemmas and pos only apply to input with extension *.frog.json* and *.frog.json.dir*
- multiple options can be given with quotes, divided by a space (for example: *'tokens lemmas pos'*)

- String parameter; default: **tokens**

--ngrams
- Specify the length of the N-grams that you want to include
- Ngram values should be given within quotes, divided by a space (for example: *'1 2'*)
- String parameter; default: **'1 2 3'**

--blackfeats
- In order to exclude words from the feature space, specify them here
- Each feature to be excluded should be given within quotes, divided by a space (for example: *'do re mi'*)
- Each ngram within the feature space that includes any of the given blackfeats will be removed
- String parameter; default: **False**

--lowercase
- Choose to lowercase all text and lemma features
- Boolean parameter; default: **False**

--minimum-token-frequency
- Option to delete all features that occur below the given threshold
- Recommended to set to 5 or 10 when applying tfidf or infogain weighting in the Vectorize module
- Integer parameter; default: **1**

1.3 Output

.features.npz Binary file in Numpy format, storing the extracted features per document in sparse format

.vocabulary.txt File that stores the index of each feature

1.4 Overview

–inputfile	–featuretypes	–ngrams	–blackfeats	–lowercase	–minimum-token-frequency	Output
docs.tok.txt	tokens	'1 2 3'	False	True	2	• docs.tokens.n_1_2_3.min2 • docs.tokens.n_1_2_3.min2
dos.frog.jsondir	'tokens lemmas pos'	1	'koala kangaroo'	False	10	• docs.tokens.lemmas.pos.n_ • docs.tokens.lemmas.pos.n_

1.5 Examples of command line usage

Extract word Ngrams from tokenized text document, lowercasing them and stripping away token Ngrams that occur less than 5 times

$ luiginlp Featurize –module quoll.classification_pipeline.modules.featurize –inputfile docs.tok.txt –lowercase –minimum-token-frequency 5

Extract lemma and pos Ngrams from directory with frogged texts

$ luiginlp Featurize –module quoll.classification_pipeline.modules.featurize –inputfile docs.frog.jsondir –featuretypes 'lemmas pos'

Frog text document, extract text and pos features and strip away any feature with the word 'snake'

$ luiginlp Featurize –module quoll.classification_pipeline.modules.featurize –inputfile docs.txt –frogconfig /mylamachinedir/share/frog/nld/frog.cfg –featuretypes 'tokens pos' –blackfeats snake

CHAPTER 2

The Vectorize module

The Vectorize module is the third module in the pipeline, taking care of weighting and pruning features based on characteristics in the training data. In contrast to the preliminary **Featurize_** and Preprocess modules, this module requires separate train and test input.

2.1 Input

If the input to Preprocess_ (.txt or .txtdir) or Featurize (.tok.txt, .tok.txtdir, .frog.json or .frog.jsondir) is given as input to traininstances or testinstances, this module is ran prior to the Vectorize module.

--traininstances

- Traininstances takes featurized or vectorized documents as input. They can come in the following formats:

1. Extension **.features.npz** - Output of the Featurize module.
2. Extension **.vectors.npz** - Output of the Vectorize module (can be used for vectorizing test documents).
3. Extension **.csv** - File with feature vectors formatted as comma-separated-values (useful when applying feature extraction that is not accomodated by quoll). When working with '.csv'-input, a file with featurenames should be created that has the same path and name as the '.csv'-file, replacing .csv with .featureselection.txt.

--trainlabels

- Extension **.labels** - File with a label per line, should be as many instances as traininstances, where the position of the label corresponds to the instance on the same position.

--testinstances

- Like traininstances, test instances takes featurized or vectorized documents as input. They can come in the following formats:

1. Extension **.features.npz** - Output of the Featurize module.
2. Extension **.csv** - File with feature vectors formatted as comma-separated-values (useful when applying feature extraction that is not accomodated by

quoll). Can only be used when the input to traininstances is of the same format; the number of columns should be as many as for the traininstances input. Like

2.2 Options

Options for Preprocess_ and Featurize_ also apply and are effective in combination with the inputfiles for these modules.

--weight
- Specify the feature weighting
- Does not work in combination with a '.csv'-file
- Options: **frequency**, **binary**, **tfidf**, **infogain
- For tfidf or infogain, it is recommended to set minimum feature frequency to 5 or 10 in the Featurize module
- String parameter; default: **frequency**

--prune
- Specify the number of features to maintain after pruning
- Does not work in combination with a '.csv'-file
- Pruning is done by ranking features based on their feature weight (total count of a feature is taken in case of 'frequency' and 'binary' weighting
- Integer parameter; default: 5000

--balance
- Choose to balance the number of train instances
- The number of instances for each class label is decreased to the instance count of the least frequent class
- Can help in case of strong class skewness
- Boolean parameter; default: **False**

--delimiter
- Specify the delimiter by which columns in the 'csv'-file are separated
- Only applies to '.csv'-files
- String parameter; default: **,**

--scale
- Option to normalize feature values to the same scale
- Only applies to '.csv'-file
- Useful in combination with some classifiers, if the features in the '.csv'-file are from different sources and have a wide range of values
- Boolean parameter; default: **False**

2.3 Output

.balanced.features.npz Balanced instances Only applied when 'balance' is chosen

.balanced.labels Labels related to balanced instances Only applies when 'balance' is chosen

.balanced.vocabulary Vocabulary related to balanced instances

2.4 Overview

–inputfile	–featuretypes	–ngrams	–blackfeats	–lowercase	–minimum-token-frequency	Output
docs.tok.txt	tokens	'1 2 3'	False	True	2	• docs.tokens.n_1_2_3.min2 • docs.tokens.n_1_2_3.min2
dos.frog.jsondir	'tokens lemmas pos'	1	'koala kangaroo'	False	10	• docs.tokens.lemmas.pos.n_ • docs.tokens.lemmas.pos.n_

2.5 Examples of command line usage

Extract word Ngrams from tokenized text document, lowercasing them and stripping away token Ngrams that occur less than 5 times

$ luiginlp Featurize –module quoll.classification_pipeline.modules.featurize –inputfile docs.tok.txt –lowercase –minimum-token-frequency 5

Extract lemma and pos Ngrams from directory with frogged texts

$ luiginlp Featurize –module quoll.classification_pipeline.modules.featurize –inputfile docs.frog.jsondir –featuretypes 'lemmas pos'

Frog text document, extract text and pos features and strip away any feature with the word 'snake'

$ luiginlp Featurize –module quoll.classification_pipeline.modules.featurize –inputfile docs.txt –frogconfig /mylamachinedir/share/frog/nld/frog.cfg –featuretypes 'tokens pos' –blackfeats snake

CHAPTER 3

Indices and tables

- genindex
- modindex
- search