python-reason Documentation

Release latest

Jan 25, 2023

CONTENTS

1 Install **3**

2 License **5**

3 Quick-Start **7**

3.1 Classification . . . 7
3.2 Clustering . . . 7
3.3 Confusion Matrix . . . 7
3.4 Part-of-Speech Tagging . . . 8
3.5 Word Tokenization . . . 8
3.6 Sentence Tokenization . . . 8
3.7 Lemmatization . . . 8
3.8 Preprocess Text (Tokenizing + Stemming) . . . 9
3.9 Frequency Distribution . . . 9
3.10 N-Grams . . . 9

Reason

Python easy-to-use natural language processing toolbox.

CHAPTER

ONE

INSTALL

Install latest stable version using pip:

```
$ pip install reason
```

CHAPTER

TWO

LICENSE

MIT – See LICENSE for details.

CHAPTER

THREE

QUICK-START

3.1 Classification

```
>>> from reason.classify import NaiveBayesClassifier
>>> classifier = NaiveBayesClassifier()
>>> classifier.fit(x, y)
>>> y_pred = classifier.predict(new_data)
>>> from reason.metrics import accuracy
>>> accuracy(y_true, y_pred)
0.9358
```

3.2 Clustering

```
>>> from reason.cluster import KMeansClusterer
>>> from reason.cluster import elbow_method
>>> elbow_method(x, clusterer=KMeansClusterer, max_k=10)
5
>>> clusterer = KMeansClusterer()
>>> labels = clusterer.fit(x, k=5)
>>> pred = clusterer.predict(new_data)
>>> from reason.cluster import DBSCAN
>>> clusterer = DBSCAN()
>>> labels = clusterer.fit(x, eps=0.21)
```

3.3 Confusion Matrix

```
>>> from reason.metrics import ConfusionMatrix
>>> cm = ConfusionMatrix(y_true, y_pred)
>>> cm
68 21 13
16 70 11
14 10 77
>>> cm[actual, predicted]
16
>>> from reason.metrics import BinaryConfusionMatrix
```

(continues on next page)

(continued from previous page)

```
>>> bcm = BinaryConfusionMatrix(b_y_true, b_y_pred)
>>> bcm.precision()
0.7837
>>> bcm.recall()
0.8055
>>> bcm.f1_score()
0.7944
```

3.4 Part-of-Speech Tagging

```
>>> from reason.tag import POSTagger
>>> text = "10 tools from the file"
>>> tagger = POSTagger()
>>> tagger.tag(text)
[('10', 'CD'), ('tools', 'NNS'), ('from', 'IN'), ('the', 'AT'), ('file', 'NN')]
```

3.5 Word Tokenization

```
>>> from reason.tokenize import word_tokenize
>>> text = "Testing reason0.1.0, (on: 127.0.0.1). Cool stuff..."
>>> word_tokenize(text, 'alphanumeric')
['Testing', 'reason0.1.0', 'on', '127.0.0.1', 'Cool', 'stuff']
```

3.6 Sentence Tokenization

```
>>> from reason.tokenize import sent_tokenize
>>> text = "Hey, what's up? I love using Reason library!"
>>> sents = sent_tokenize(text)
>>> for sent in sents:
...     print(sent)
Hey, what's up?
I love using Reason library!
```

3.7 Lemmatization

```
>>> from reason.stem import PorterStemmer
>>> text = "watched birds flying"
>>> stemmer = PorterStemmer()
>>> stemmer.stem(text)
['watch', 'bird', 'fly']
>>> from reason.stem import regex_stem
>>> regex_pattern = r'^(.*?)(ous)?$'
```

(continues on next page)

(continued from previous page)

```
>>> regex_stem('dangerous', regex_pattern)
danger
```

3.8 Preprocess Text (Tokenizing + Stemming)

```
>>> from reason import preprocess
>>> text = "What's up? I love using Reason library!"
>>> preprocess(text)
[["what's", 'up', '?'], ['i', 'love', 'us', 'reason', 'librari', '!']]
```

3.9 Frequency Distribution

```
>>> from reason.analysis import FreqDist
>>> words = ['hey', 'hey', 'oh', 'oh', 'oh', 'yeah']
>>> fd = FreqDist(words)
>>> fd
Frequency Distribution
Most-Common: [('oh', 3), ('hey', 2), ('yeah', 1)]
>>> fd.most_common(2)
[('oh', 3), ('hey', 2)]
>>> fd['yeah']
1
```

3.10 N-Grams

```
>>> sent = "Reason is easy to use"
>>> from reason.util import bigrams
>>> bigrams(sent)
[('Reason', 'is'), ('is', 'easy'), ('easy', 'to'), ('to', 'use')]
>>> from reason.util import trigrams
>>> trigrams(sent)
[('Reason', 'is', 'easy'), ('is', 'easy', 'to'), ('easy', 'to', 'use')]
>>> from reason.util import ngrams
>>> ngrams(sent, 4)
[('Reason', 'is', 'easy', 'to'), ('is', 'easy', 'to', 'use')]
```