Quotation Extractor for Portuguese
Testes

Rafael Reis
rrsilva@inf.puc-rio.br
Orientador: Ruy Milidiú

9 de Dezembro de 2015

Conteúdo

1 Introdução **2**

2 Testes Unitários **2**
2.1 Teste wis.sortTasksByEnd 2
2.2 Teste wis.createPreviousArray 3
2.3 Teste wis.schedule 4
2.4 Logs gerados 5

3 Testes do Modelo **5**

1 Introdução

Foram implementados dois tipos de testes:

- Testes unitários: testes unitários automatizados, utilizando o pacote `unittest` do Python.
- Teste de qualidade do modelo: utilizando as métricas *precision* e *recall* de aprendizado de máquina.

2 Testes Unitários

Como o software foi projetado em pequenos módulos e funções, foi fundamental a criação de testes unitários automatizados para cada parte, adotando-se a prática de se escrever o teste em paralelo com o desenvolvimento. Isso foi fundamental para que os erros fossem corrigidos o mais cedo possível, sem afetar a corretude das fases posteriores.

Os testes de cada pacote foram colocados no subdiretório *tests*, seguindo as boas práticas para Python.

No total, foram criados 27 testes unitários.

Tentou-se sempre que possível isolar o teste de uma função das outras partes do *software*, como nos **exemplos** a seguir:

2.1 Teste wis.sortTasksByEnd

```
def test_sortTasksByEnd(self):

        tasks = [Task(0,2,1,0), Task(1,4,1,0), Task(0,1,1,0)]
        tasksSorted = [Task(0,1,1,0), Task(0,2,1,0), Task(1,4,1,0)]

        tasks = wis.sortTasksByEnd(tasks)

        ends = [t.end for t in tasks]
        endsSorted = [t.end for t in tasksSorted]

        self.assertEqual(ends, endsSorted)
```

2.2 Teste wis.createPreviousArray

```
def test_createPreviousArray(self):

        tasks = [Task(0,0,0,0), Task(0,1,1,0), Task(0,2,1,0),
                        Task(1,4,1,0), Task(4,5,1,0), Task(2,6,1,0),
                        Task(5,7,1,0)]

        p = wis.createPreviousArray(tasks)
        pCorrect = [0, 0, 0, 1, 3, 2, 4]

        self.assertEqual(p, pCorrect)
```

2.3 Teste wis.schedule

```
def test_schedule(self):

        tasks = [Task(0,1,1,0), Task(0,2,1,0),
                        Task(1,4,1,0), Task(4,5,1,0), Task(2,6,1,0),
                        Task(5,7,1,0)]

        set_correct = [Task(0,1,1,0), Task(1,4,1,0), Task(4,5,1,0),
                        Task(5,7,1,0)]
        max_w, set_tasks = wis.schedule(tasks)

        self.assertEqual(set_correct, set_tasks)
```

2.4 Logs gerados

O pacote de teste `unittest` do Python permite a execução de todos os *scripts* de testes gerados para um projeto. A seguir, o resultado desta execução:

```
python -m unittest -v
/Users/rafaelreis/anaconda/lib/python3.5/site-packages/sklearn/utils/fixes.py:64: DeprecationWarning:
  if 'order' in inspect.getargspec(np.copy)[0]:
test_precision (qextractor.metrics.tests.test_metrics.TestMetrics) ... ok
test_recall (qextractor.metrics.tests.test_metrics.TestMetrics) ... ok
test_train (qextractor.model.tests.test_train.TestTrain) ... examples size: 372
ok
test_load (qextractor.model.tests.test_train_example.TestTrainExample) ... X length:  147
Y length:  8
ok
test_boundedChunk (qextractor.preprocessing.tests.test_baseline.TestBaseline) ... ok
test_detoken (qextractor.preprocessing.tests.test_baseline.TestBaseline) ... ok
test_firstLetterUpperCase (qextractor.preprocessing.tests.test_baseline.TestBaseline) ... ok
test_quotationEnd (qextractor.preprocessing.tests.test_baseline.TestBaseline) ... ok
test_quotationStart (qextractor.preprocessing.tests.test_baseline.TestBaseline) ... ok
test_quoteBounds (qextractor.preprocessing.tests.test_baseline.TestBaseline) ... ok
test_verbSpeechNeighb (qextractor.preprocessing.tests.test_baseline.TestBaseline) ... ok
test_binary (qextractor.preprocessing.tests.test_feature.TestFeature) ... ok
test_columns (qextractor.preprocessing.tests.test_feature.TestFeature) ... ok
test_create (qextractor.preprocessing.tests.test_feature.TestFeature) ... ok
test_pos (qextractor.preprocessing.tests.test_feature.TestFeature) ... ok
test_load (qextractor.preprocessing.tests.test_globoquotes.TestGloboQuotes) ... test_load
ok
test_createInput (qextractor.preprocessing.tests.test_preprocessing.TestPreprocessing) ... ok
okst_createInputTest (qextractor.preprocessing.tests.test_preprocessing.TestPreprocessing) ...
testConverter (qextractor.preprocessing.tests.test_verbspeech.TestVerbspeech) ... ok
testSearchAfirmou (qextractor.preprocessing.tests.test_verbspeech.TestVerbspeech) ... ok
testSearchDissemos (qextractor.preprocessing.tests.test_verbspeech.TestVerbspeech) ... ok
testSearchExplicaram (qextractor.preprocessing.tests.test_verbspeech.TestVerbspeech) ... ok
test_createPreviousArray (qextractor.wis.tests.test_wis.TestWis) ... ok
test_schedule (qextractor.wis.tests.test_wis.TestWis) ... ok
test_sortTasksByEnd (qextractor.wis.tests.test_wis.TestWis) ... ok
testInterval (qextractor.wis.tests.test_wisinput.TestWisInput) ... ok
test_corefAnnotated (qextractor.wis.tests.test_wisinput.TestWisInput) ... ok

----------------------------------------------------------------------
Ran 27 tests in 62.257s

OK
```

3 Testes do Modelo

Para garantir a qualidade do modelo, foi criado o módulo `metrics`, que calcula as métricas *precision* e *recall* a partir de um conjunto de dados e do resultado

w do modelo.

Durante a etapa de calibração do modelo, é esperado que essas medidas se aproximem. Abaixo, os valores para a *precision* e *reacall* nos dados de treino e de teste, respectivamente.

1	0.7321958456973294	0.6573643410852713	0.7387096774193549	0.5175438596491229
2	0.7418397626112759	0.6589147286821705	0.7612903225806451	0.5175438596491229
3	0.7477744807121661	0.6604651162790698	0.7483870967741936	0.5
4	0.7522255192878339	0.6573643410852713	0.7483870967741936	0.5
5	0.7626112759643917	0.6542635658914728	0.7548387096774194	0.5087719298245614
6	0.7640949554896143	0.6496124031007752	0.7612903225806451	0.5
7	0.7692878338278932	0.6449612403100775	0.7677419354838709	0.49122807017543857
8	0.7655786350148368	0.6387596899224807	0.7677419354838709	0.4824561403508772
9	0.7737388724035609	0.6372093023255814	0.7677419354838709	0.4824561403508772
10	0.7759643916913946	0.6403100775193798	0.7774193548387097	0.4824561403508772
11	0.7752225519287834	0.6387596899224807	0.7774193548387097	0.4824561403508772
12	0.7781899109792285	0.6372093023255814	0.7806451612903226	0.49122807017543857
13	0.7781899109792285	0.6403100775193798	0.7806451612903226	0.49122807017543857
14	0.7796735905044511	0.6372093023255814	0.7806451612903226	0.49122807017543857
15	0.7789317507418397	0.6372093023255814	0.7838709677419354	0.49122807017543857
16	0.7781899109792285	0.6372093023255814	0.7774193548387097	0.49122807017543857
17	0.7774480712166172	0.6294573643410852	0.7838709677419354	0.4824561403508772
18	0.7818991097922848	0.6294573643410852	0.7870967741935484	0.4824561403508772