Exercise in Rule-based Extraction

Tobias Eder

CIS

tobias.eder@in.tum.de

8.-9. November 2017

Übersicht

Task

Wir suchen spezifische Informationen in unstrukturieren (Text-)Daten.

Dataset: Carniege Mellon School of Computer Science (CMU) Seminar Announcements

Beispiel

```
<0.5.5.93.11.12.56.skees+@SKEES.ADM.CS.CMU.EDU (Jim Skees).0>
Type:     cmu.cs.scs
Topic:    Healthy Office Seminar
Dates:    12-May-93
Time:     <stime>1:00 PM</stime>
PostedBy: skees+ on 5-May-93 at 11:12 from SKEES.ADM.CS.CMU.EDU (Jim Skees)
Abstract:

SCS will sponsor a seminar on ways to achieve a healthy
office environment at <stime>1 p.m</stime>. Wednesday, May 12th, in <location>Wean
Hall 5409</location>.
The speaker will be <speaker>Karlis Mateus</speaker>, a representative of
Steelcase, Inc., the office equipment manufacturer.  He will
focus on the ways in which poorly designed office furniture
can contribute to carpal tunnel syndrome and other illnesses.

I have asked <speaker>Mr. Mateus</speaker> to keep the discussion as generic as
possible.  However, I do anticipate he will use some illustrations
of Steelcase products during his presentation.
```

Tags

- stime - Startzeit
- etime - Endzeit
- location - Veranstaltungsort
- speaker - Vortragender

Setup

- Material von der Kurswebsite herunterladen.
- Verzeichnis entpacken: `$ tar xzf IE_exercise1.tgz`
- Ins Verzeichnis `IE_exercise1/toy-example` wechseln.
- Datei `cmu.andrew.org.sds.pol_sci-226_0` von Hand annotieren.

Setup Part 2

- Im Root-Verzeichnis der Übung folgendes Ausführen:
- Dateien vorbereiten: `$ make data`
- Erster Test-Run: `$ make`
- Gibt uns einen F-Measure Score wenn alles geklappt hat:

```
ederto@fruehjahrslorchel.cip.ifi.lmu.de:~/Uni/WiSe1718/IE_exercise1 $ make
Applying rules...done
true positives: 16
false positives: 1046
false negatives: 365
all: 381
Precision: 0.015065913370998116
Recall: 0.04199475065616798
F1: 0.022176022176022176
ederto@fruehjahrslorchel.cip.ifi.lmu.de:~/Uni/WiSe1718/IE_exercise1 $
```

Regeln formulieren

- Datei `extractor.py` im Editor öffnen:

```
1  @Rule
2  def hasRoom(tokenGroup, checksize=2):
3      if tokenGroup[0].lower().strip() == 'room':
4          return tokenGroup
5      return []
6
7  rules = [
8            hasRoom,
9            #unusedRule,
10         ]
```

- Eigene Regeln formulieren!

Regeln Part 2

- In der Liste 'rules' stehen alle Regeln die ausgeführt werden sollen.
- Der Parameter 'checksize' gibt die länge der Token an die pro Durchlauf überprüft werden.
- Bsp: checksize=4 heisst es werden fortlaufend immer 4 Token übergeben.
- Wenn neue Regeln eingetragen sind 'extractor.py' speichern und wieder mit `$ make` ausführen.
- Fehlererkennung mit `$ make compare` versuchen!

Tips

- Es können mehrere Listen mit Regeln erstellt werden.
- Regeln können entweder als 'one' oder 'all' gematcht werden.
- Siehe Funktion 'applyRules()' in Zeile 60 von 'extractor.py'

Train/Test-Split

- Wir evaluieren und testen Regeln auf dem Trainingsset.
- Am Ende kann mit dem Testset verglichen werden: `$ make test`