Praktische Dialogmodellierung, Universität Potsdam, SoSe 19, David Schlangen

Aufgabe nach Sitzung 3: NLU mit regulären Ausdrücken

Ihre Aufgabe ist es, mittels regulären Ausdrücken eine NLU-Komponente zu erstellen, die in der DSTC2-Domäne arbeitet.

Vorbereitungen

Zur Vorbereitung sollten Sie `compile_nlu_regexp_data.py` im `DSTC`-Verzeichnis ausführen. (Davor sollten Sie den Anweisungen in `README.md` gefolgt sein.) Danach sollten Sie nach dem Muster `usr_sem-dstc2_SPLIT.json` benannte Dateien vorfinden, jeweils für `dev` und `train` als `SPLIT`, und jeweils noch einmal mit `-test.json`.

Hier ist ein Ausschnitt aus `usr_sem-dstc2_dev-test.json`:

```
{
  "turn_id": 0,
  "dial_id": 0,
  "usr_utt": "i would like to find an expensive restaurant in the south part"
},
```

Hier ist der korrespondierende Abschnitt aus `usr_sem-dstc2_dev.json`:

```
{
  "turn_id": 0,
  "dial_id": 0,
  "usr_utt": "i would like to find an expensive restaurant in the south part",
  "usr_sem": [
    {
      "slots": [
        [
          "pricerange",
          "expensive"
        ]
      ],
      "act": "inform"
    },
    {
      "slots": [
        [
          "area",
          "south"
        ]
      ],
      ],
      "act": "inform"
    }
  ]
}
```


NLU

Ihre Aufgabe ist es also, den Eintrag unter `usr_sem` selbst zu erzeugen. Dazu sollen Sie reguläre Ausdrücke erstellen, und eine Methode, diese auf `usr_utt` anzuwenden, so dass sie die gewünschte Information extrahieren können.

Dazu ist möglicherweise die Information aus der Domänen-Ontologie (siehe Notebook) hilfreich.

Überlegen können Sie, ob Sie *intent* und *slot+value* getrennt oder gemeinsam matchen wollen.

Beachten müssen Sie, dass manche Äußerungen durch mehr als einen Dialogakt wiedergegeben sind.

Ein Muster für Ihren Code, das zeigt, wie die Daten eingelesen und die Ergebnisse ausgegeben werden sollen, finden Sie in `regexp_nlu.py`.

Arbeiten Sie mit dem `dev`-Teil der Daten bei der Entwicklung der Muster. (Dazu könnte Ihnen vielleicht das Notebook nützlich sein mit dem Pandas-DataFrame, in welchem Sie einfach nach Beispielen für Intents und Slots suchen können.)

Versuchen Sie es zu vermeiden, während der Entwicklung in den `train`-Split zu schauen.

Evaluation

Der zweite Teil der Aufgabe ist es, sich Gedanken über die Evaluation der Ergebnisse zu machen. (Das sollten Sie gleichzeitig mit dem anderen Teil, oder gar zuerst machen, damit Sie während der Entwicklung immer nachvollziehen können, ob sich die Ergebnisse verbessern oder nicht.)

Hierzu steht wieder ein Rahmen bereit, in `eval_nlu.py`. Die Tests müssen Sie sich allerdings noch überlegen. Es empfiehlt sich, zu überlegen, wie man “teilweise richtig” operationalisieren kann und dafür Punkte verteilen kann. Ihr Evaluationsskript soll am Ende jedenfalls (verschiedene) Zahlen ausspucken, die die Qualität einfangen, sowie zur genaueren Fehlerkontrolle für jedes einzelne Beispiel protokollieren, was falsch oder richtig war.

Während der Entwicklung sollten Sie nur auf `dev` evaluieren. Erst beim allerletzten Durchgang, wenn Sie mit der Entwicklung der regulären Ausdrücke fertig sind, sollten Sie auch auf `train` evaluieren. (Warum?)

Einreichung

Bitte reichen Sie ihre Skripte bis Ende Montag, 6.5., per email ein. Die Skripte sollten jeweils in einer Kopie vom `DSTC`-Ordner (mit heruntergeladenen und kompilierten Daten) ausführbar sein, so dass ich sie problemlos testen kann.