Übungsblatt 11

Präsenzaufgaben

Aufgabe 1 Weiterverarbeitung syntaktischer Analysen

In dieser Aufgabe sollen Sie die Ausgaben eines state-of-the-art-Parsers, nämlich des spacy-Parsers, weiterverarbeiten.

Mit dem Ziel, Sie erst einmal mit den typischen Strukturen einer solchen Aufgabe vertraut zu machen, sollen Sie in dieser Aufgabe lediglich entscheiden, ob die Eingabe einen Infinitivsatz mit Akkusativobjekt enthält.

Zur Klarheit betrachten Sie die folgenden positiven und negativen Beispiele:

+ Er beabsichtigt , den Kuchen ganz alleine zu essen .
+ Er behauptet , ihn gesehen zu haben .
– Er glaubt , nach Hause zu fliegen .
– Zu fliegen ist schön .
– Er will gehen .

Anmerkung: Orientieren Sie sich bei der Benutzung des spacy-Parsers am Jupyter Notebook dieser Woche.

Aufgabe 2 Informationsextraktion per Syntaxanalyse

Gegenstand dieser Aufgabe ist eine anwendungsnahe Möglichkeit, Ergebnisse einer Syntaxanalyse weiterzuverarbeiten. Aus den syntaktischen Abhängigkeiten eines Textes soll (unter Zuhilfenahme einiger Normalisierungsschritte) eine semantische Repräsentation der im Text enthaltenen Informationen gewonnen werden.

Für die syntaktische Analyse soll der `DependencyParser` von spacy verwendet werden. Die semantische Repräsentation eines Satzes sei ein Knowledge Graph Tripel bestehend aus Subjekt, Prädikat und Objekt (Bei Fehlen eines der beiden Elemente soll `None` geschrieben werden.). Die Menge der Prädikate sei durch die Lemmata der vorkommenden Verben definiert. Sie können bei der Implementierung davon ausgehen, dass kein Satz zwei verschiedene Aussagen mit dem gleichen Prädikat enthält.

Die nötigen Normalisierungsschritte umfassen insbesondere Passivkonstruktionen und Relativsätze. Folgendes Beispiel illustriert das gewünschte Ergebnis:

Eingabe:

> I shot an elephant in my pajamas. The elephant was seen by a giraffe in the desert. The bird I need is a raven. The man who saw the raven laughed out loud.

Ausgabe:

- (I, shoot, elephant)
- (giraffe, see, elephant)
- (I, need, bird)
- (bird, `IS`, raven)
- (man, see, raven)
- (man, laugh, `None`)

Anmerkung: Sie können sich insbesondere für die Benutzung des spacy-Parsers an dem im Jupyter Notebook gegebenen Code orientieren.

Hausaufgaben

Aufgabe 3 Parent Annotation

Parent Annotation kann die Performanz einer CFG wesentlich verbessern. Schreiben Sie eine Funktion, die einen gegebenen Syntaxbaum dieser Optimierung unterzieht. Auf diese Art und Weise transformierte Bäume können dann wiederum zur Grammatikinduktion verwendet werden.

Ihre Funktion sollte neben dem Syntaxbaum selbst noch folgende Parameter erwarten:

- `parentHistory` soll die Anzahl der Vorgänger sein, die zusätzlich zum direkten Elternknoten berücksichtigt werden. (Kann bei der Lösung der Aufgabe auch ignoriert werden.)
- `parentChar` soll ein Trennzeichen sein, das bei den neuen Knotenlabels zwischen dem ursprünglichen Knotenlabel und der Liste von Vorgängern eingefügt wird.

Aufgabe 4 Mehr Semantik für IE

Zusätzlich zu den in Aufgabe 2 behandelten Konstruktionen sollen jetzt auch negierte und komplexe Sätze mit Konjunktionen sinnvoll verarbeitet werden.

Eingabe:

> I see an elephant. You didn't see the elephant. Peter saw the elephant and drank wine.

Ausgabe:

- (I, see, elephant)
- (You, not_see, elephant)
- (Peter, see, elephant)
- (Peter, drink, wine)

Aufgabe 5 Fragen zu NLTK-08-extras, 2.9 ("Viterbi-Parser")

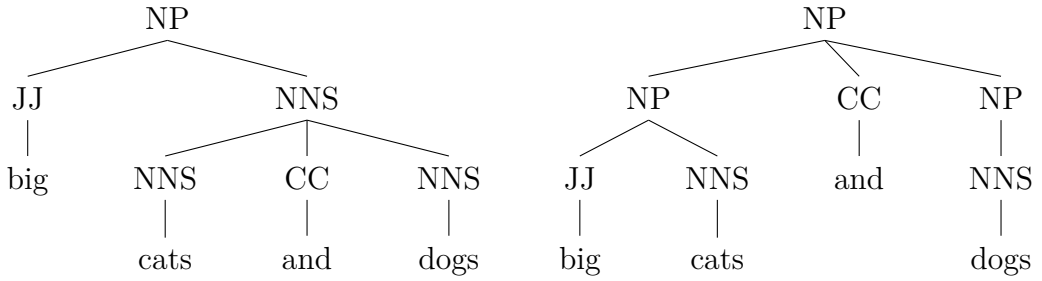
(a) Betrachten Sie die folgende im Kapitel gegebene PCFG:

```
grammar = nltk.PCFG.fromstring('''
  NP  -> NNS [0.5] | JJ NNS [0.3] | NP CC NP [0.2]
  NNS -> "cats" [0.1] | "dogs" [0.2] | "mice" [0.3]
  NNS -> NNS CC NNS [0.4]
  JJ  -> "big" [0.4] | "small" [0.6]
  CC  -> "and" [0.9] | "or" [0.1]
  ''')
viterbi_parser = nltk.ViterbiParser(grammar)

sent = 'big cats and dogs'.split()
for tree in viterbi_parser.parse(sent):
    print(tree)
# (NP (JJ big) (NNS (NNS cats) (CC and) (NNS dogs))) (p=***)
```

Berechnen Sie die Wahrscheinlichkeit für die Ableitung in der letzten Zeile.

(b) Betrachten Sie das dazugehörige Tracing-Output des ViterbiParsers:

```
>>> sent = 'big cats and dogs'.split()
>>> viterbi_parser = nltk.ViterbiParser(grammar, trace = 3)
>>> for tree in viterbi_parser.parse(sent):
>>>     print(tree)

Inserting tokens into the most likely constituents table...
   Insert: |=...| big
   Insert: |.=..| cats
   Insert: |..=.| and
   Insert: |...=| dogs
Finding the most likely constituents spanning 1 text elements..
   Insert: |=...| JJ -> 'big' [0.4]                0.4000000000
   Insert: |.=..| NNS -> 'cats' [0.1]              0.1000000000
   Insert: |.=..| NP -> NNS [0.5]                  0.0500000000
   Insert: |..=.| CC -> 'and' [0.9]                0.9000000000
   Insert: |...=| NNS -> 'dogs' [0.2]              0.2000000000
   Insert: |...=| NP -> NNS [0.5]                  0.1000000000
Finding the most likely constituents spanning 2 text elements..
   Insert: |==..| NP -> JJ NNS [0.3]               0.0120000000
Finding the most likely constituents spanning 3 text elements..
   Insert: |.===| NP -> NP CC NP [0.2]             0.0009000000
   Insert: |.===| NNS -> NNS CC NNS [0.4]          0.0072000000
```




```
   Insert: |.===| NP -> NNS [0.5]                    0.0036000000
  Discard: |.===| NP -> NP CC NP [0.2]               0.0009000000
  Discard: |.===| NP -> NP CC NP [0.2]               0.0009000000
Finding the most likely constituents spanning 4 text elements..
   Insert: |====| NP -> JJ NNS [0.3]                 0.0008640000
  Discard: |====| NP -> NP CC NP [0.2]               0.0002160000
  Discard: |====| NP -> NP CC NP [0.2]               0.0002160000
(NP (JJ big) (NNS (NNS cats) (CC and) (NNS dogs))) (p=***)
```

Warum werden die Analysen in den Discard-Zeilen verworfen?

- ◯ Das Verwerfen verhindert eine Endlosrekursion.
- ◯ Die Analysen sind mit der gegebenen Grammatik nicht möglich.
- ◯ Ihre Wahrscheinlichkeiten sind zu gering.
- ◯ Sie analysieren die Eingabesequenz nicht komplett.

(c) Ein statistischer ChartParser findet folgende zwei Ableitungen für die NP *big cats and dogs*:

```
1 (NP (JJ big) (NNS (NNS cats) (CC and) (NNS dogs)))
2 (NP (NP (JJ big) (NNS cats)) (CC and) (NP (NNS dogs)))
```

- Um welchen Parser kann es nicht handeln?
  - ◯ `InsideChartParser`
  - ◯ `InsideChartParser` mit `beam-size`
  - ◯ `LongestChartParser`
  - ◯ `ViterbiParser`
- Nach welchem Kriterium wird beim Parsen mit dem `InsideChartParser` (= Lowest-Cost-First-Strategie) die *edge queue* sortiert?
  - ◯ nach der Länge der Ableitung
  - ◯ nach der Wahrscheinlichkeit
  - ◯ nach der `beam-size`
- Um welche Art der Ambiguität handelt es sich bei den beiden gefundenen Ableitungen?
  - ◯ Koordinationsambiguität
  - ◯ PP-Attachment-Ambiguität
  - ◯ Temporale Ambiguität