Übungsblatt 13

23.01.19

Präsenzaufgaben

Aufgabe 1 Evaluationsmetriken

Betrachten Sie die Daten in Tabelle 1. Es handelt sich um ein vereinfachtes Tagging-Schema fürs Chunking, bei dem nur zwischen „Teil einer NP“ (`NP`) und „nicht Teil einer NP“ (`O`) unterschieden wird.

Füllen Sie Tabelle 2 aus, indem Sie die geforderten Evaluationsmetriken für binäre Klassifikation berechnen.

Sample	0	1	2	3	4	5	6	7	8	9
Ground Truth	NP	O	NP	O	O	NP	NP	NP	NP	O
Chunker 1	NP	NP	NP	O	NP	O	NP	NP	NP	NP
Chunker 2	NP	O	NP	O	O	O	O	O	NP	O
Chunker 3	O	O	O	O	O	NP	NP	NP	NP	O

Tabelle 1: Daten für Aufgabe 1

	Accuracy	Precision	Recall	F1-Score
Chunker 1				
Chunker 2				
Chunker 3				

Tabelle 2: Evaluationstabelle

Aufgabe 2 Herunterladen von Ressourcen

Das CoNLL 2000 Korpus ist ein POS- und Chunk-getaggtes Korpus (IOB-Format), das in ein Test- und ein Trainingskorpus aufgeteilt ist. Wir werden es zum Training und zur

Evaluation von Chunk-Parsern verwenden. Laden Sie es sich dafür zunächst über die Ressource `corpora/conll2000` herunter.

Wenn Sie es erfolgreich heruntergeladen haben, können Sie folgendermaSSen darauf zugreifen:

```
from nltk.corpus import conll2000
conll2000.chunked_sents('train.txt', chunk_types=['NP'])[99]
```

Das `chunk_types`-Argument dient der Auwahl von Chunk-Typen (in diesem Beispiel Nominalphrasen).

Aufgabe 3 Chunking mit regulären Ausdrücken

Erstellen Sie einen einfachen `RegexpParser`, der für Nominalphrasen charakteristische Tags zu NPs zusammenfasst. Solche charakteristischen Tags sind z. B. Kardinalzahlen (`CD`), Artikel (`DT`), Adjektive (`JJ`, `JJR`, `JJS`) und natürlich Substantive (`NN`, `NNS`, `NNP`, `NNPS`).

Weitere interessante Tags wären `PDT` (z. B. *both*, *a lot of*), `POS` (*'s*), `PRP` (Personalpronomen), `PRP$` (Possessivpronomen).

Evaluieren Sie Ihren Parser anschlieSSend auf dem CoNLL 2000 Korpus:

```
test_sents = conll2000.chunked_sents('test.txt', chunk_types=['NP'])
cp = nltk.RegexpParser(regex)
print(cp.evaluate(test_sents))
```

Aufgabe 4 Datenbasiertes Chunking

(a) Betrachten Sie den folgenden Code für einen Chunker, der für jedes POS-Tag das wahrscheinlichste Chunk-Tag berechnet (Training) und dieses dann zur Testzeit ausgibt.

```
class UnigramChunker(nltk.ChunkParserI):
    def __init__(self, train_sents):
        train_data = [
            [
                (t,c)
                for w,t,c in nltk.chunk.tree2conlltags(sent)
            ]
            for sent in train_sents
        ]
        self.tagger = nltk.UnigramTagger(train_data)

    def parse(self, sentence):
        pos_tags = [pos for (word,pos) in sentence]
        tagged_pos_tags = self.tagger.tag(pos_tags)
        chunktags = [
            chunktag for (pos, chunktag) in tagged_pos_tags
```

```
        ]
        conlltags = [
            (word, pos, chunktag)
            for ((word, pos), chunktag)
            in zip(sentence, chunktags)
        ]
        return nltk.chunk.conlltags2tree(conlltags)
```

Trainieren und evaluieren Sie den `UnigramChunker` auf dem CoNLL 2000 Korpus.

(b) Der `ConsecutiveNPChunker`, dessen Code Sie sich aus dem Jupyter Notebook dieser Woche herauskopieren können, basiert auf einem Klassifikator. Dies erlaubt uns, die Features, die für die Klassifikation extrahiert werden, genauer zu bestimmen.

Ein Feature-Extraktor lässt sich als Funktion z. B. so definieren:

```
def npchunk_features(sentence, i, history):
    word, pos = sentence[i]
    return {"pos": pos}
```

Dieser Feature-Extraktor extrahiert genau ein Feature, nämlich das POS-Tag, das auch der `UnigramChunker` verwendet hat.

Evaluieren Sie den `ConsecutiveNPChunker` mit diesem Feature-Extraktor und vergleichen Sie seine Performanz mit der des `UnigramChunker`s.

(c) Fügen Sie weitere Features für das aktuelle Wort, das vorhergehende POS-Tag und das vorhergehende Chunk-Tag zur Extraktion hinzu und beobachten Sie jeweils die Auswirkungen auf die Performanz in der Evaluation.

Hausaufgaben

Aufgabe 5 Feature Engineering

AnschlieSSend an Aufgabe 4 sollen Sie hier weitere Features implementieren und jeweils deren Auswirkungen auf die Performanz beobachten.

Fügen Sie von oben nach unten immer jeweils ein neues Feature nach dem anderen hinzu und behalten Sie alle bisherigen Features bei. Dann notieren Sie jeweils den Performanzgewinn (oder -verlust) im Vergleich zur vorherigen Zeile in untenstehender Tabelle.

Fügen Sie auSSerdem noch zwei neue Features an das Ende der Tabelle, die Sie sich selbst ausdenken!

Zum Schluss können Sie auch eine sogenannte *ablation study* durchführen. Dabei wird immer genau eines der Features entfernt, um anhand des Performanzverlustes dessen Beitrag zum Ergebnis zu bemessen.

Neues Feature	Performanzgewinn (F1)	Ablation Study
Lookahead: nächstes POS-Tag		
Paar-Feature: vorhergehendes und aktuelles POS-Tag konkateniert		
Paar-Feature: aktuelles und nächstes POS-Tag konkateniert		
Lookahead: nächstes Wort		
Menge aller POS-Tags seit dem Artikel (`DT`)		