Studienarbeit NLP WS 2021/22

Ostbayerische Technische Hochschule
Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik
Prof. Dr.-Ing. Ulrich Schäfer

Ziel ist die Entwicklung einer Semantische Suchmaschine, die es ermöglicht, in Gerichtsurteilen nach Subjekt-Prädikat-dObjekt-iObjekt-Strukturen zu suchen, unterstützt während der Eingabe der Suchterme durch n-Gramme und Word-Embeddings.

Datenquelle: Open Legal Data dump (court decisions)
http://openlegaldata.io/research/2019/02/19/court-decision-dataset.html

Alle in einem json (eine pro Zeile):
https://static.openlegaldata.io/dumps/de/2019-02-19_oldp_cases.json.gz

Sauber in Sätze splitten. Achtung: Satzgrenzen können z.B. nie innerhalb eines Zitations-Tags stehen. Abkürzungsliste für sentence splitter ggf. ergänzen um im Corpus häufig vorkommende Abkürzungen.
Umlaute (mindestens die XML-Entity-Schreibweise ᫗ und die Java/Python-Schreibweise \u2ae4) erkennen und (durch „echte“ Unicode-Zeichen) ersetzen, so dass der Dependenzparser damit klar kommt.
HTML-Tags und custom tags ordentlich behandeln (Nachschauen beim predictive N-Gram-Modell – s.u. – ist erlaubt) oder mit Beautifulsoup parsen und ersetzen bzw. löschen.

Alle Sätze mit Spacy Dependenzparser für deutsch (de_core_news_lg-Modell) parsen.
Resultat in „flache“ Quintupel zerlegen: root(Verb/Prädikat), nsubj(Subjekt), dobj, iobj und Rest und speichern. Conj-Strukuren (und-Verknüpfung von Satzteilen in nsubj, dobj, iobj) behandeln: z.B. verdoppeln von Einträgen oder in gleiches Feld mit einpacken (nicht automatisch in rest!).

Storage: Ein System aus
- Solr (Stemming!), ggf. Listenwertige Felder zur Hilfe nehmen
- Elastic Search (Stemming!), könnte auch tiefer strukturierte Bäume speichern
- Relationale DB
- GraphDB (z.B. Neo4J)
- MongoDB
- XMLDB (z.B. Oracle Berkeley DB, BaseX; Performanz?)

auswählen (Solr oder Elastic Search empfohlen) und für jeden Satz sowohl den Volltext als auch die in Quintupel zerlegten Struktur speichern. Dabei ist ein Satz = ein „Dokument“ (<doc> in Solr). Eindeutiges Namensschema Urteil-ID + Satz-Nr. für jeden Eintrag.

Benutzerschnittstelle mit
z.B. django, cherrypy oder flask (Python) oder alternativ node.js (+ggf. leichtgewichtiges Framework)

Bei Abfrage drei Varianten:
1. Normale Volltextsuche (satzweise; zur Kontrolle/zum Vergleich)
2. Experimentelle Suche auf Feldern wie subj, verb, iobj, dobj, rest (kombiniert)
3. Anfrage selbst mit Dependenzparser parsen und Struktur entsprechend suchen

Zusätzlich Unterstützung der Eingabe: mit
a) predictive N-gram model
https://github.com/openlegaldata/oldp-notebooks/blob/master/notebooks/ngram_model.ipynb

auf dem Corpus
UND
b) Word Embeddings (Minimum: die fertigen aus de_core_news_lg-Modell oder – besser: aus dem Corpus bauen, vgl. a), Vokabular reduzieren!!), trainieren z.B. mit Gensim
~~c) BERT lookup (https://deepset.ai/german-bert)~~

Mögliche Folge/Kontextworte für die Suche vorschlagen. Hier bietet sich jQueryUI Autocomplete an (https://jqueryui.com/autocomplete/), notfalls Button+Text(vorschläge).

Gefundene Sätze nach Auswahl (z.B. per Link oder Button) im Original reparsen. Dazu in eigenem Feld den Text oder das Parsingergebnis speichern oder in einem separaten core ablegen und über eindeutige ID + Satz-Nr. „verlinken“ und mit der Visualisierung von spacy ausgeben:
`displacy.render(doc, style="dep")`
Weitere Felder in Solr/ElasticSearch kosten nicht viel, erst recht nicht, wenn sie redundate (Teil-)information enthalten.

Ausgabe 24.11.2021
Abgabe 18.01.2022
Vorführung: 19.01.2022

Bewertungsgrundlagen:
Studienarbeits-Beschreibung (ca. 8-10 Seiten), Aufbau

- Code: Korrektheit, Vollständigkeit, Eleganz, Lösungswege, Fehlerbehandlung
- Code-Dokumentation: inline (Struktur in der Studienarbeitsdoku beschreiben)
- Originalität/zusätzliche sinnvolle Features
- Usability der Anwendung
- Kurzvortrag mit Demonstration der Anwendung