Prüfungsaufgabe 3 - 2 a)

Der Quellcode hat verschiedene Bereiche.
Direct Code Area: Dies sind die Python Dateien, welche NICHT in einem package (Unterordner) gespeichert wurden. Das sind die Dateien, welche direkt zum Prozess gehören, welche man Schritt für Schritt ausführen muss, um die gewollten Daten zu QASparql zu erhalten.

Diese werden laut der Dokumentation des Codes in einer bestimmten Reihenfolge ausgeführt. Bevor man jedoch zu dem Prozess kommen kann, muss man ein Preprocessing durchführen. In diesem Schritt kann man manuell oder mithilfe eines Powershell Skriptes die zuvor Trainierten Worteinbettungsmodelle herunterladen (GloVe). Außerdem wird FastText verwendet, um die Modelle zu trainieren.

Der Hauptprozess besteht aus insgesamt 10 Schritten. Der Code hat für jeden Schritt eine bestimmte Python Datei strukturiert und diese sollen nacheinander ausgeführt werden, sodass der gewünschte Output erzeugt wird.
Diese Code-Dateien werden nun einzeln betrachtet, um zu untersuchen, welche Verarbeitungen darin stattfinden und wofür diese Verarbeitungen stattfinden.

lcquad_dataset.py
In dieser Python Datei wird der LCQuad Datensatz betrachtet und eine weitere Datei erzeugt. Linked_Answer.json beinhaltet Queries und deren dazugehörige Antworten und SPARQL Abfragen. Diese Datei liest also nur den LCQuad Datensatz, welcher dann verarbeitet und in Linked_Answers gespeichert wird.

lcquad_answer.py
Diese Datei erzeugt die Antworten und Status für den LCQuad Datensatz. Darunter, ob die Antwort korrekt, falsch oder nicht vorhanden ist, welche Features vorhanden sind und welche Queries erzeugt wurden.

preprocess_lcquad.py
Diese Datei ist in ihrer eigenen Directory und ist dafür da, um den LC_Quad Datensatz zu bearbeiten, sodass es für Tree-LSTM Training angewendet werden kann. Hierfür wird der originale Datensatz zu Train, Trial und Test gesplittet. Diese haben jeweils 70%, 20% und

10% vom originalen Datensatz. Daraufhin werden dann die Abhängigkeits-Parse-Bäume, die passenden Inputs und Outputs generiert, um das Tree-LSTM Modell trainieren zu können.

treelstm/main.py

Diese Datei ist die Main Datei vom Tree-LSTM Package. Hier wird das TREE-LSTM trainiert. Dieses Modell ist eine Erweiterung von Long Short-Term Memory, sodass man mit Input arbeiten kann, welche in einer Baumstruktur ist. Normalerweise arbeitet LSTM mit einer Sequenz von Input, in dem die Reihenfolge auch wichtig ist. Hier ist es nun Möglich die Daten von Eltern-Knoten mit zu beziehen.

entity_lcquad.py

Ein Hauptfokus in diesem Programmteil ist das EARL. Entity and Relation Mapping. Die Daten, welche eingelesen werden, sind im Data Directory enthalten und das Resultat des Mappings wird auch gespeichert in derselben Directory. Nachdem die Inputdaten eingelesen und geparsed (also verarbeitet) wurden, wird durch die Frage-Antwort-Paare iteriert, um dann die Entitäten und Relationen zu extrahieren. Hierfür werden verschiedene APIs genutzt. Für jede verschiedene API gibt es eigene Funktionen, in der die Entitäten/Relationen aus diesen API-Datensätzen extrahiert werden. Wie zum Beispiel Falcon, Spotlight, TagMe und EARLDemo. Während des Durchlaufs werden die Entitäten gemerged. Dieser Schritt ist besonders wichtig, denn es werden nicht nur die alten Entitäten mit den neuen Entitäten ergänzt, sondern die Confidence-Werte werden aktualisiert. Mithilfe der aktualisierten Confidence-Werte werden Redundanzen verhindert und es kann vergewissert werden, dass es zuverlässige Ergebnisse sind.

entity_qald.py

Die Datei entity_qald.py hat einen sehr ähnlichen Prozess wie die entity_lcquad.py Datei. Der einzige Unterschied sind die Daten, die verarbeitet werden. LCQUAD, auch Large-scale Complex Question Answering Data, und QALD, auch Question Answering over Linked Data, sind beides Datensätze. Diese Datensätze werden genutzt, um Frage-Antwort-Systeme zu trainieren und zu testen. Heißt, der einzige Unterschied, der zwischen dieser und vorheriger Datei besteht, ist der Datensatz. Es werden also in beiden Dateien Entity Relation Mapping betrieben, aber basierend auf verschiedenen Datensätzen.

entity_qald.py

Diese Datei ähnelt ebenfalls den Prozessen der vorherigen Entity-Dateien, jedoch werden hier die Mappings für die QALD-7 (Question Answering over Linked Data) Datensätze generiert. Diese ist als JSON Datei in das Repository mitgegeben.

lcquad_test.py, lcquad_alltest.py und qald_test.py

Diese Dateien führen Tests durch, um das Question Answering System zu prüfen. Die verschiedenen Dateien Testen die Systeme basierend auf die verschiedenen generierten Datensätzen.

question_type_analysis.py

Diese Datei analysiert die Genauigkeit der Fragetypklassifikation basierend auf LC-QuAD und QALD-7 Datensätze. Es werden dann PNG-Dateien generiert, welche die Wahrheitsmatrix für die einzelnen Datensatztypen (QALD, LCQUAD) darstellen. Diese geben an, welche Fragen richtig bzw. falsch gekennzeichnet wurden.

result_analysis.py

Die result_analysis Datei analysiert und speichert die Ergebnisse in die output Directory. Hier werden dann für die verschiedenen Datensätze und es wird die Präzision, der Recall und der F1-Score von den Ergebnissen berechnet.