Institut für Informatik
Datenbanken und Informationssysteme
Prof. Dr. Stefan Conrad
Julia Romberg, Philipp Grawe, Thomas Germer

WiSe 2019
19. Dezember 2019
Blatt 3

Information Retrieval und Natural Language Processing

Übung

Abgabefrist: 13. Januar 2020, 14:30 Uhr!
über ILIAS

Aufgabe 1 *Analogy Puzzles*

In der Vorlesung wurden *Analogy Puzzles* mit dem Paper von Bolukbasi et al.[1] vorgestellt. Verwenden Sie dafür GloVe[2].

(a) Implementieren Sie die Suche nach Analogien wie zum Beispiel: "*man* is to *king* as *woman* is to x" in der Datei `PuzzleSolver.py`.

(b) Finden Sie fünf eigene Beispiele (d.h. diese Beispiele sollen nicht auf z.B. der GloVe Website oder in der Vorlesung genannt werden) für Analogien. Erweitern Sie dafür die Funktion `test_solve_puzzle` analog.

Hinweise:

- Wir stellen Ihnen 400.000 GloVe Vektoren[3] in der Datei `glove_word_embeddings.npy` im schneller zu ladenden NumPy-Format zur Verfügung. Die dazugehörigen Wörter sind in der Datei `words.txt` in der gleichen Reihenfolge enthalten.

Aufgabe 2 *Word Embeddings zur semantischen Suche*

Zum Suchen von ähnlichen Dokumenten zu einer gegebenen *Query* können *Word Embeddings* verwendet werden. Hierfür sollen Sie *ELMo* verwenden.

(a) Implementieren Sie diese Suchfunktionalität auf dem gegebenen Datensatz. Dabei gibt es folgenden Bedingungen:

1. Die *Query* soll aus mehreren, durch Leerzeichen getrennte Begriffen bestehen können.
2. Um ähnliche Dokumente zu finden, sollen die Dokumente nicht nach Begriffen gefiltert oder andere Techniken, die nicht die Ähnlichkeit von *Word Embeddings* berechnen, verwendet werden. Dies schließt wiederum nicht aus, dass Dokumente vorverarbeitet werden dürfen, um *Word Embeddings* zu berechnen.
3. Ihre Implementierung soll die Klasse `Database` erweitern, sodass `run_query` die Aufgabe erfüllt.
4. Die Methode soll die Ergebnissliste absteigend nach Ähnlichkeit sortiert zurück gegeben und aus Tupeln von Ähnlichkeit und Dokumenten-ID bestehen. Details und ein Beispiel können dem Quelltext entnommen werden.

[1] Bolukbasi, Tolga, et al. "Man is to computer programmer as woman is to homemaker? Debiasing word embeddings." Advances in neural information processing systems. 2016.

[2] Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation."Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[3] `https://nlp.stanford.edu/projects/glove/`

5. Um die Embeddings zu berechnen, sollen Sie die Bibliothek `flair`[4] und die dort bereits trainierten Modelle verwenden.
6. Schreiben sie eine *Readme*, die beschreibt wie Ihr `Database` Objekt zu verwenden ist.
7. Führen Sie die *Queries* aus der Datei `queries.txt` aus. Speichern Sie die dazugehörigen Ergebnisse für $k = 10$ in der gleichen Reihenfolge in der Datei `results.txt`, mit einer Ergebnissmenge pro Zeile im Format:

$$score_1, document\_id_1; score_2, document\_id_2; ...; score_k, document\_id_k$$

Hinweise:

- Die Datei `Database.py` enthält die zu erweiternde Klasse.
- `documents.txt` enthält ein Dokument pro Zeile, dabei ist die Zeilennummer die ID des Dokuments. Diese Reihenfolge darf von Ihnen nicht verändert werden.
- Sie können einmal berechnete Embeddings der Dokumente speichern, um sie nicht bei jedem Aufruf neu zu berechnen.

(b) Beantworten Sie folgende Fragen mit jeweils ein bis zwei Sätzen in einer Textdatei mit dem Namen `aufgabe_2b.txt`:

1. Wie kann ELMo auf Sätzen bzw. Dokumenten verwendet werden?
2. Welche Probleme sehen Sie dabei?

Zum Bestehen notwendige allgemeine Bedingungen:

- Das Verwenden von Python ≥ 3.7.
- Die Abgabe des Readme und der bereitgestellten Python-Dateien und nicht als Jupyter Notebook.
- Das Verwenden von ausschließlich in der `requirements.txt` genannten externen Bibliotheken.
- Eine selbständige Einzelbearbeitung ohne kopierten Code.
- Die korrekte Bearbeitung aller Aufgaben.

[4] `https://github.com/zalandoresearch/flair`