INR Teil 2 – Implementierung und Evaluation von Ranking-Modellen

Erweitern Sie Ihr im ersten Teil entwickeltes IR-System um die Verarbeitung von Bag-of-Words-Anfragen. Dazu sollen Sie zwei Modelle implementieren und evaluieren:

1) **Verpflichtend**: Ein tf-idf-basiertes Vektorraummodell;
2) **Optional**: Wahlweise ein auf LSI, word2vec oder BERT basierendes Modell;

tf-idf-Modell

Implementieren Sie den FastCosine-Algorithmus aus Kapitel 8, Folie 9. Für die Termgewichtung $w_{t,d}$ nehmen Sie die Formel aus Kapitel 7, Folie 41. Behandeln sie k als frei wählbaren Parameter und experimentieren Sie mit unterschiedlichen Werten.

LSI-Modell

Transformieren Sie die auf rohen Term Frequencies basierende Term-Document-Matrix mithilfe von LSI in einen latenten Raum von d Dimensionen und werten Sie die Anfragen mithilfe der Cosinus-Ähnlichkeit im latenten Raum aus. Dabei soll d konfigurierbar sein (experimentieren zunächst mit einem Wert von $d = 50$). Sie können die Transformation wie in der Musterlösung zur LSI-Programmieraufgabe per Hand programmieren oder auch auf Bibliotheken wie gensim zurückgreifen.

Word2Vec-Modell

Überlegen Sie sich, wie Sie word2vec zum Dokumenten-Retrieval nutzen können. Eine einfache Möglichkeit besteht z.B. darin, für alle Terme einer Anfrage/eines Dokuments den Mittelwert über die zugehörigen Word Embeddings zu bilden. Die Anfrage und die Dokumente werden somit jeweils durch einen mittleren Embedding-Vektor repräsentiert und können gemäß Cosinus-Ähnlichkeit verglichen und gerankt werden.

Die Word Embeddings müssen Sie nicht selbst trainieren, sondern können auf vortrainierte Embeddings zurückgreifen. Als Quellen bieten sich z.B. folgende Embeddings an:

- Auf Google News trainiertes Embedding (word2vec-google-news-300). Dieses lässt sich in gensim wie folgt laden:
  ```
  import gensim.downloader as api
  wv = api.load('word2vec-google-news-300')
  ```
- Auf Wikipedia trainierte Embeddings: https://wikipedia2vec.github.io/wikipedia2vec/pretrained/

Terme aus den Anfragen und Dokumenten, die in den vortrainierten Embeddings nicht enthalten sind, sollen ignoriert werden.

BERT-Modell

Kontextuelle Sprachmodelle wie BERT können auf sehr vielfältige Art und Weise für das Information Retrieval eingesetzt werden. Eine umfangreiche Übersicht finden Sie in [Lin et al., 2021]. Als einfachster Ansatz für eine Implementierung bietet sich MonoBERT an, dessen Grundidee darin besteht, zunächst mithilfe eines traditionellen Ranking-Algorithmus eine Kandidatenmenge von Dokumenten zu erzeugen, auf der dann mithilfe von BERT ein Re-Ranking vorgenommen wird. Details dazu finden Sie in der Originalarbeit [Nogueira, 2019] sowie in [Lin et al., 2021; Abschnitt 3.2].

Um Ihnen eine Nachimplementierung dieses Ansatz zu erleichtern, stellen wir Ihnen ein Grundgerüst in Form der Skripte finetune.ipynb und monobert.py zur Verfügung (s.u.).

INR Teil 2 – Implementierung und Evaluation von Ranking-Modellen

Testdaten

Zum Testen verwenden Sie den CISI-Corpus (weitere Infos auf kaggle). Der CISI-Korpus besteht aus drei Dateien:

- CISI.ALL enthält 1.460 Dokumente mit einer eindeutigen ID (.I), Titel (.T), Autor(.A), Abstract (.W) und einer Liste von Cross-Referenzen zu anderen Dokumenten (.X). Extrahieren Sie die einzelnen Dokumente mit ihrer ID und dem Abstract.
- CISI.QRY enthält 112 Volltextanfragen mit einer eindeutigen ID (.I) und dem Anfragetext (.W). Die meisten Anfragen haben die Form einer oder mehrerer natürlich-sprachlicher Fragen. Parsen und tokenisieren den Fragetext genauso wie den Text der Dokumente.
- CISI.REL enthält die Groundtruth-Daten für die Relevanz der Dokumente aus CISI.ALL und die Anfragen aus CISI.QRY. Jede Zeile in CISI.REL enthält vier Werte: QueryID, DocID sowie zwei nicht weiter erläuterte Zahlen, die Sie ignorieren können.

Testen Sie Ihr System, indem Sie sich für die ersten 35 Anfragen aus CISI.QRY jeweils eine gerankte Liste von Dokumenten zurückgeben lassen.

Evaluation

Werten Sie für jede Anfrage und jedes von Ihnen implementierte Modell folgende Kennzahlen mithilfe der Ground Truth-Daten aus CISI.REL aus:

- Precision@{5|10|20|50}
- Recall@{5|10|20|50}
- F1@{5|10|20|50}
- 11-point-precision-recall-curve
- R-Precision
- MAP

Stellen Sie die Evaluationsdaten übersichtlich in Tabellen und Diagrammen dar. Erstellen Sie für folgenden Anfragen aus CISI.QRY zusätzlich noch jeweils eine 11-point-precision-recall-curve:

- Q11, Q14, Q19 und Q20

Grundgerüste für die Implementierung

Als Startpunkt und Hilfestellung für Ihre Implementierung können Sie die folgenden Python-Dateien verwenden:

- `retrieval.py`: Diese Datei enthält die abstrakte Klasse `InitRetrievalSystem`. Ihre eigenen Retrieval-Systeme (VR, LSI, word2vec, MonoBERT) sollen Sie von dieser Klasse ableiten und die Methoden `retrieve()` bzw. `retrieve_k()` implementieren. Zur Illustration stellen wir Ihnen eine konkrete Beispiel-Implementierung mithilfe der `BM25`-Klasse aus der gensim-Bibliothek zur Verfügung. Sie können dieses Retrieval-System als Baseline zum Vergleich mit Ihren eigenen Implementierungen sowie als initiales Ranking-System für die MonoBERT-Implementierung verwenden.
- `retrieval_metrics.py`: Diese Datei enthält Funktionssignaturen für die Evaluationsmetriken, die Sie in INR, Kap. 13, kennengelernt haben. Diese Funktionen müssen von Ihnen noch ausimplementiert werden.

- `finetune.ipynb` und monobert.py: Falls Sie den MonoBERT-Ansatz implementieren, können Sie diese Dateien als Startpunkt verwenden. Sie enthalten das Grundgerüst für das Finetuning eines BERT-Models zur Berechnung einer Wahrscheinlichkeit, ob ein Dokument bzgl. einer Anfrage relevant ist, bzw. für die Anwendung dieses Modells zum Re-Ranking von Dokumenten gemäß dem MonoBERT-Ansatz. Die grundsätzliche Algorithmik ist bereits vorgegeben und muss an den mit `TODO` kommentierten Stellen von Ihnen ergänzt werden.

Implementierungsumgebung

Prinzipiell sind Sie frei in der Wahl der von Ihnen verwendeten Programmiersprache, Entwicklungs- und Rechnerumgebung. Aufgrund der Vorgaben und der zu nutzenden Bibliotheken bietet sich jedoch die Implementierung mit Python in Form von Jupyter Notebooks an. Wenn Sie nicht Ihren eigenen Rechner verwenden wollen, stehen Ihnen nach Registrierung auf den jeweiligen Diensten folgende Optionen mit GPU-Support zur Verfügung:

- Google Colab: https://colab.research.google.com
- Kaggle: https://www.kaggle.com/
- DataScience JupyterHub an der HSNR (für einen Account wenden Sie sich bitte an pascal.quindeau@hsnr.de): https://data-science.hsnr.de/jupyter/hub/login

Literatur

- [Lin et al., 2021] Lin, Jimmy, Rodrigo Nogueira, and Andrew Yates. "Pretrained transformers for text ranking: Bert and beyond." arxiv preprint:2010.06467 (2021) https://doi.org/10.48550/arxiv.2010.06467
- [Nogueira et al., 2019] Nogueira, Rodrigo, and Kyunghyun Cho. "Passage Re-ranking with BERT." arXiv preprint arXiv:1901.04085 (2019). https://doi.org/10.48550/arxiv.1901.04085