Automated Collection of Evaluation Dataset for Semantic Search in Low-Resource Domain Language

Anastasia Zhukova[1], Christian E. Matt[2], Bela Gipp[1]
[1]University of Göttingen, [2]eschbach GmbH

TL;DR

We explored an automated method of generating a test collection for domain-specific IR evaluation by using **an ensemble of “weak” (L)LM bi-encoders combined with an LLM for re-ranking**, which is prompted with specific examples of relevance score assignments.

Motivation

Domain-specific languages that use a lot of specific terminology often fall into the **category of low-resource languages**. Collecting test datasets in a narrow domain is time-consuming and requires skilled human resources with domain knowledge and training for the annotation task. Moreover, the existing methods of IR dataset collection fall short due to the limitations of the language models trained on high-resource languages of common knowledge do not transfer well to these low-resource domain contexts.

Ensemble learning is a machine learning technique that **combines multiple individual models**, often called "weak learners," to create **a more powerful and accurate predictive model by mitigating each other's weaknesses**.

Time stamp	Functional locations	Product	Description
2021/08/01 10:04	Alpha-L1-R111-T5002 Tank 5002	ABC	Gesendet an HAH Transfer von B6 nach B1 98779 H2 Wasser nach B6 98781 H2 Organik bleibt bei SFP Wasser D.O. 2-1 .59 2-3 11.06 Kohlenstofftransfer zu K2 B4 32' B9 18' K2 20' Loto't BAC-Zulaufwasser

An example of a shift log in a process industry in German that document system statuses, production metrics, and any incidents or anomalies. The domain-specific language usus a lot of abbreviations, codes, and terminology.

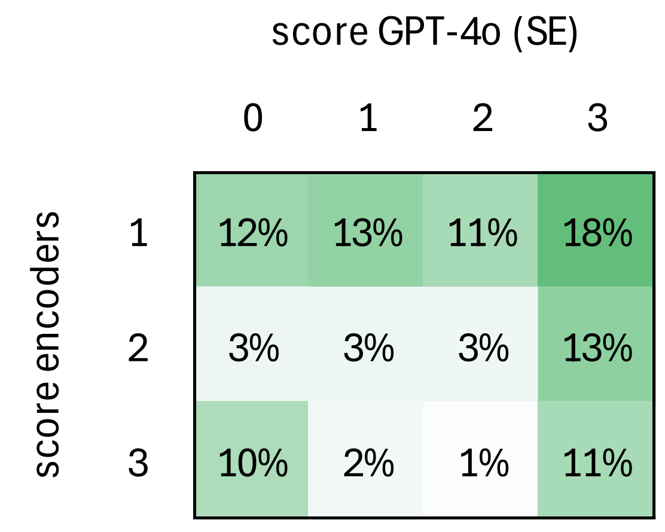
RQ: How can **a principle of the ensemble learning** be transferred to **“weak” (L)LMs** to collect evaluation **dataset for semantic search**?

Methodology

The methodology of the ensemble for annotating a test collection for semantic search comprises two main parts:

(1) document indexing
(2) creation of the query-document pairs.

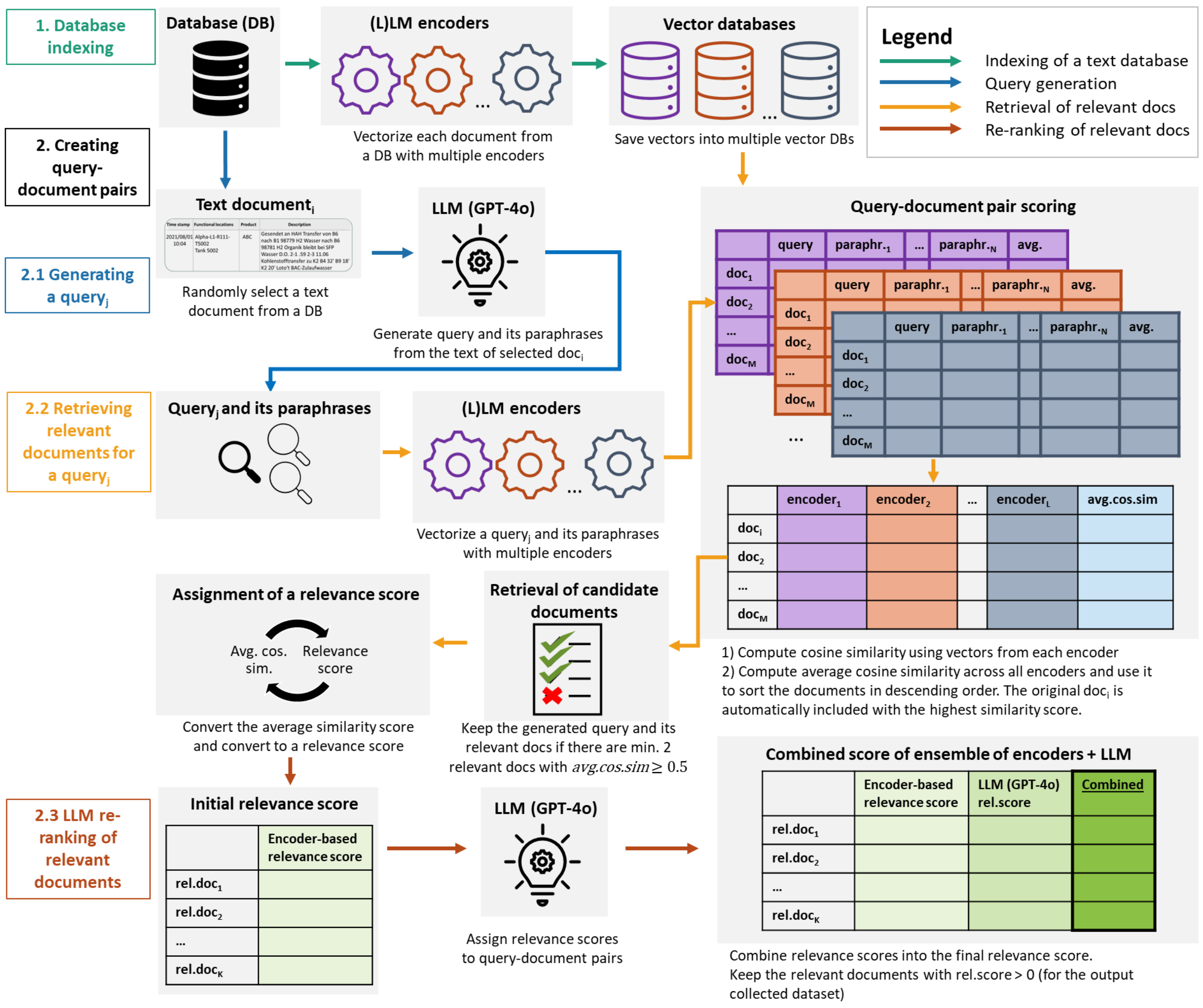
The key aspect of document indexing is using **a set of encoders with various architectures and training strategies**. The goal is to combine **different aspects of the document similarity** that each encoder has learned. Re-ranking combines the relevance score based on the document similarity with the score generated by a generative LLM. **LLM assesses the relevance of the query-document pair independently** from the score used for the retrieval, thus allowing the combining of another "point of view" to the query-document relevance.



		score GPT-4o (SE)			
		0	1	2	3
score encoders	1	12%	13%	11%	18%
	2	3%	3%	3%	13%
	3	10%	2%	1%	11%

Combining relevance scores

The agreement matrix shows different trend in the relevance score assignment: **the ensemble assigns low relevance scores** whereas **GPT-4o tend to give high relevance scores**. Hence, when computing the combined score, we give more weight to the GPT scores when the score is 3 or to the ensemble scores when it is 1; otherwise, we compute their average.



Legend
Indexing of a text database
Query generation
Retrieval of relevant docs
Re-ranking of relevant docs

1. Database indexing

Database (DB)

(L)LM encoders

Vectorize each document from a DB with multiple encoders

Vector databases

Save vectors into multiple vector DBs

2. Creating query-document pairs

2.1 Generating a query$_j$

Text document$_i$

Randomly select a text document from a DB

LLM (GPT-4o)

Generate query and its paraphrases from the text of selected doc$_i$

2.2 Retrieving relevant documents for a query$_j$

Query$_j$ and its paraphrases

(L)LM encoders

Vectorize a query$_j$ and its paraphrases with multiple encoders

Query-document pair scoring

	query	paraphr.$_1$	…	paraphr.$_N$	avg.
doc$_1$					
doc$_2$					
…					
doc$_M$					

	encoder$_1$	encoder$_2$	…	encoder$_L$	avg.cos.sim
doc$_i$					
doc$_2$					
…					
doc$_M$					

1) Compute cosine similarity using vectors from each encoder
2) Compute average cosine similarity across all encoders and use it to sort the documents in descending order. The original doc$_i$ is automatically included with the highest similarity score.

Retrieval of candidate documents

Keep the generated query and its relevant docs if there are min. 2 relevant docs with $avg.cos.sim \geq 0.5$

Assignment of a relevance score

Avg. cos. sim. Relevance score

Convert the average similarity score and convert to a relevance score

2.3 LLM re-ranking of relevant documents

Initial relevance score

	Encoder-based relevance score
rel.doc$_1$	
rel.doc$_2$	
…	
rel.doc$_K$	

LLM (GPT-4o)

Assign relevance scores to query-document pairs

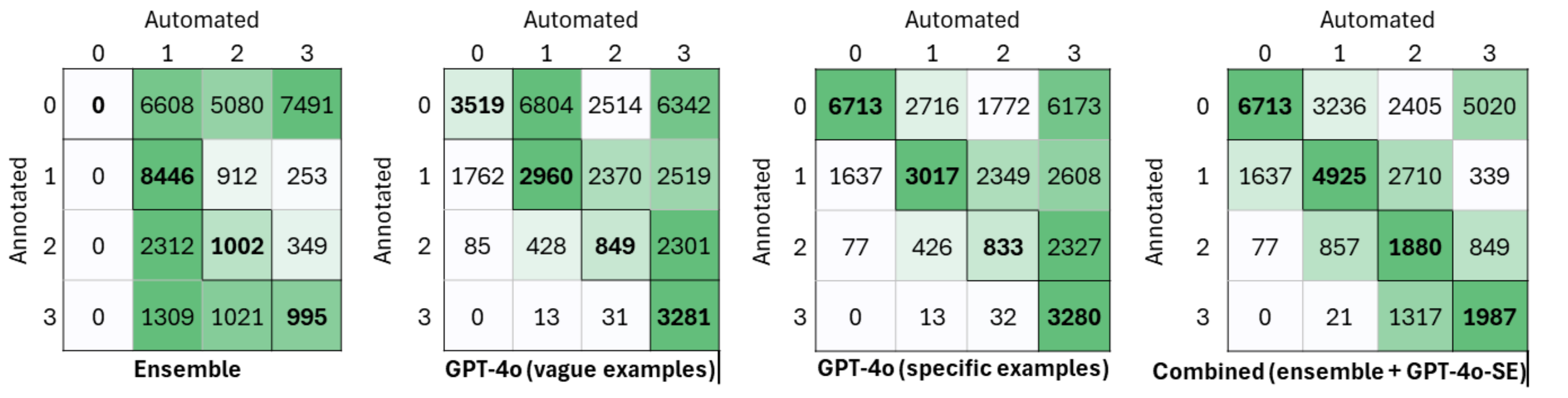
Combined score of ensemble of encoders + LLM

	Encoder-based relevance score	LLM (GPT-4o) rel.score	Combined
rel.doc$_1$			
rel.doc$_2$			
…			
rel.doc$_K$			

Combine relevance scores into the final relevance score.
Keep the relevant documents with rel.score > 0 (for the output collected dataset)

Evaluation

- The goal was **to evaluate how the proposed approach agreed with how a human assessed the query-document pairs**.
- 7 shift books, 28-30 queries with up to 1000 relevant documents each for the manual annotation to make the task feasible.
- Annotator: a native German speaker familiar with the domain. Instructions were identical to those used in the prompt for LLM.
- Metrics:
  - (1) **inter-coder agreement** between two annotators (i.e., automated and manual) measured by Krippendoff's alpha,
  - (2) **classification metrics** for the imbalanced classes, i.e., macro precision, recall, and F1-score,
  - (3) **a ranking metric for IR**, such as nDCG.
- The proposed approach **outperformed the baselines in all metrics**, especially **improved the inter-coder agreement by a factor of 4**



		Automated			
		0	1	2	3
Annotated	0	**0**	6608	5080	7491
	1	0	**8446**	912	253
	2	0	2312	**1002**	349
	3	0	1309	1021	**995**

Ensemble

		Automated			
		0	1	2	3
Annotated	0	**3519**	6804	2514	6342
	1	1762	**2960**	2370	2519
	2	85	428	**849**	2301
	3	0	13	31	**3281**

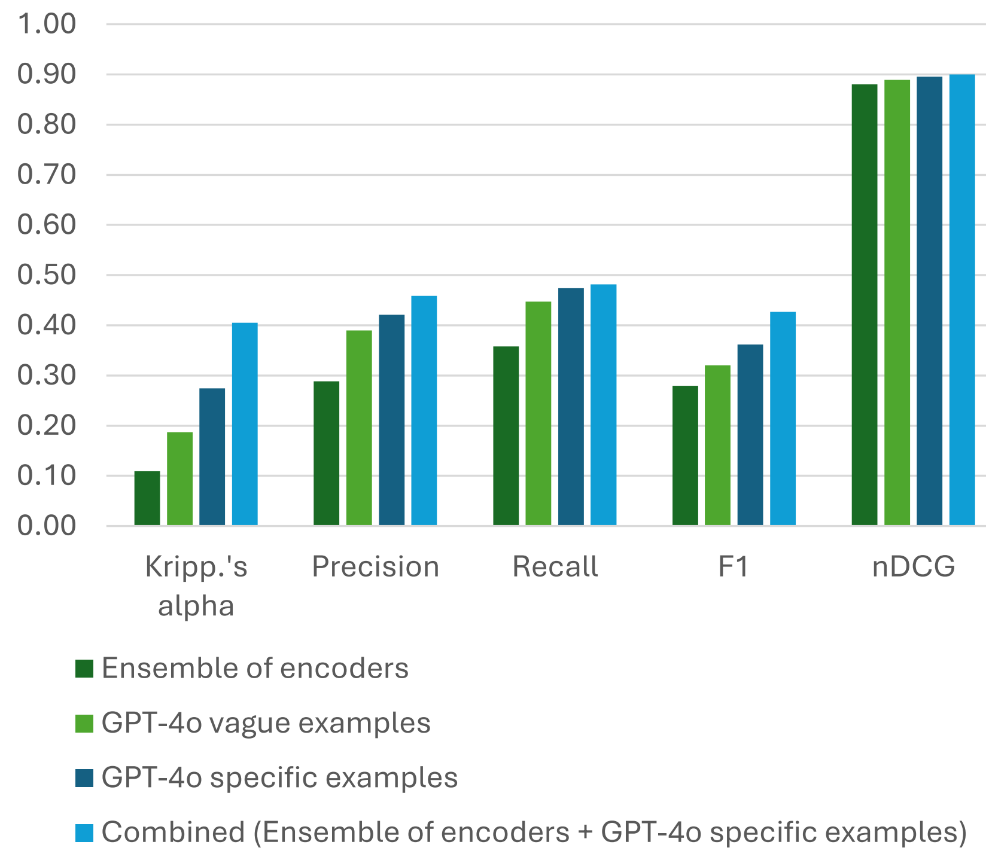
GPT-4o (vague examples)

		Automated			
		0	1	2	3
Annotated	0	**6713**	2716	1772	6173
	1	1637	**3017**	2349	2608
	2	77	426	**833**	2327
	3	0	13	32	**3280**

GPT-4o (specific examples)

		Automated			
		0	1	2	3
Annotated	0	**6713**	3236	2405	5020
	1	1637	**4925**	2710	339
	2	77	857	**1880**	849
	3	0	21	1317	**1987**

Combined (ensemble + GPT-4o-SE)



Recall	Ens.	GPT-4o VE	GPT-4o SE	Combined (ens. + GPT-4o SE)
0	0	18.3	38.6	38.6
1	87.9	30.8	31.4	51.2
2	27.4	23.2	22.7	51.3
3	29.9	98.7	98.6	59.8
Mean	36.3	42.8	47.9	50.2

Combining an ensemble of encoders (Ens.) with GPT-4o-SE yielded worse recall for relevance scores 1 and 3 but significantly improved the recall on the more ambiguous score 2.

Discussion

- **Specific examples** of query-document pairs and their scores **significantly improve the results**.
- The recent development of the multilingual encoder and decoder (L)LMs **make the approach transferrable to other low-resource languages**, e.g., Multilingual-E5-base, EuroLLM-9B, Salamandra-7B, etc.
- **Multi-agent LLM** can facilitate solving the complicated task of the relevance score assignment.

Contact

Anastasia Zhukova
@ana_zhukova
anastasia.zhukova@uni-goettingen.de
https://gipplab.org/projects/plant-assistant/