TBD

TBD

TBD

Jonas Michel

Student ID: 24238749

Liselotte Lichtenstein

Student ID: 24240597

TBD

1 Question 1 (10 Marks)

Given a query that a user submits to an IR system and the top N documents that are returned as relevant by the system, devise an approach (high-level algorithmic steps will suffice)

to suggest query terms to add to the query. Typically, we wish to give a large range of suggestions to the users capturing potential intended query needs, i.e., high diversity of terms that may capture the intended query context/content.

Ziel: query expanden können - which makes the querey better the terms should be relevant (able to expand the query in a meaningful way - find the correct topic), diverse (cover different topics)

we should design a heuristic allowing that

Ansatz: clusters, there are papers on that

1. query with q_n terms, for the whole vocuabulary we get the most similar terms (maybe 1000 t_n) 2. cluster the t_n terms into k clusters - then we get the most central term from every cluster

no terms that were in the query

how to get the most describtive terms for the query

when - runtime vs offline

there are multiple expansion methods out here: like sysnonm, related term, contextual,

recall vs precision tradeoff what we trying to solve and how we are acutal doing it

depending what a system we are designing (incooperate user data, e.g. Galway, Ireland, Europe)

user can give a temperature - to select the cluster

Algorithm for QE suggestions:

0. have embedding and clustering for vocab in document collection 1. have query terms q and their embedding [use sentence embedding to make sure additional terms constrict/specify possible similar terms] 2. get r relevant terms (made up of n and m) 2.1 get n most similar terms from embedding space 2.2 get m co-occurence terms from top N returned documents on original query 3. get corresponding clusters for r 4. select top k clusters (balancing max distance and max size) * use multi resolution clustering to be able to tackle different levels of specificity in the query 5. get one descriptive term for each cluster (e.g. closest to centroid) -¿ need to check its not in the original query 6. return these terms as QE suggestions

The goal of the query expansion is to improve the search results by adding terms that increase the specificity of the query. (xxx do not want to just add terms to query that

Algorithm 1 Query Expansion (QE) Suggestions

```
1: Input: Query terms $q$ and document collection with embeddings and clustering
2: Output: Query expansion (QE) suggestions
3:
4: Step 1: Get query terms $q$ and their embedding (use sentence embeddings to ensure
   additional terms constrain/specify possible similar terms)
5: Step 2: Get $r$ relevant terms, made up of $n$ and $m$
6:     Step 2.1: Get $n$ most similar terms from embedding space
7:     Step 2.2: Get $m$ co-occurrence terms from top $N$ returned documents on the
       original query
8: Step 3: Get corresponding clusters for $r$
9: Step 4: Select top $k$ clusters (balancing max distance and max size)
10:    Step 4.1: Use multi-resolution clustering to address different levels of specificity
       in the query
11: Step 5: Get one descriptive term for each cluster (e.g., closest to centroid)
12:    Step 5.1: Check that the descriptive term is not in the original query
13: Step 6: Return the descriptive terms as QE suggestions
```

further define a given topic bust suggest terms that would clarify which topic is meant) - thats what most methods do but we want diversity + relevance -¿ use global method and select clusters accordingly The terms should be relevant to what has already been provided and diverse in order to cover a wide range of topics. QE is always a tradeoff between recall and precision, xxx here xxx Our solution incorporates xxxcontext basedxxx as well as pseudo relevance feedback (PRF) based concepts, that is another way to incorporate relevance and diversity in the suggestions.

The algorithmxxx (do we call it that? or heuristic?) takes the query and its embedding as input and is based on the foundation of an embedded and clustered vocabulary of the document collection. Based on the embedded query an number of semantically similar terms from the vocabulary as well as cooccurring terms from the top-N returned documents are selected as xxx relevant to query termsxxx. Query embedding: here important to capture context/use sentence embedding/... to make sure additional terms constrict/specify possible similar terms and do not expand results The clusters which contain these terms are then considered as the clusters that contain candidates for query expansion. Since users can only process a limited number of suggestions, the top-k clusters are selected based on a tradeoff between maximal distance and maximal size. (again: diversity (distance) and relevance(size)) This ensures that a variety of topics is covered while still providing terms that are relevant and have a high probability of being a relevant topic (not just in context of query but generally). From these selected clusters the terms that will be suggested are extracted as the terms closest to the centroid, that are not in query of the cluster as it can be assumed, that these are terms that are most descriptive for the cluster.

Use mix of global context based QE and PRF based QE to ensure that terms that are similar as well as terms that co-occur influence the suggestions. Whilst there may be a significant overlap,

Precomputing the vocabulary embedding and clustering reduces runtime complexity significantly. xxx embedding model for query can also be pretrained In order to be able to preocess different levels of specificity in the query, a multi resolution clustering approaches could be applied. These allow to select a cluster granularity that is relevant for the given query.

2 Section TBD.

Appendix A

Appendix B

	Low Risk	**Mid Risk**	**High Risk**
Total Number of Entries	404	336	272
Percentage	40%	33%	27%

Table 1: Class Distribution within the Dataset

	Decision Tree			**KNN**		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Low Risk	0.90	0.75	0.82	0.67	0.81	0.74
Mid Risk	0.76	0.84	0.80	0.74	0.51	0.60
High Risk	0.84	0.89	0.87	0.77	0.86	0.81
Weighted	0.83	0.83	0.83	0.72	0.72	0.72

Table 2: Numerical Results