UNIVERSITÄT
LEIPZIG

Research Seminar

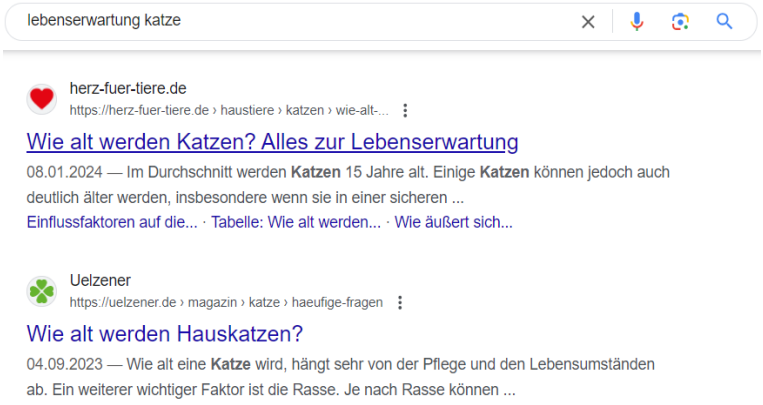
Ranking Explanations via Query Reformulation for Web Search

April 24, 2024

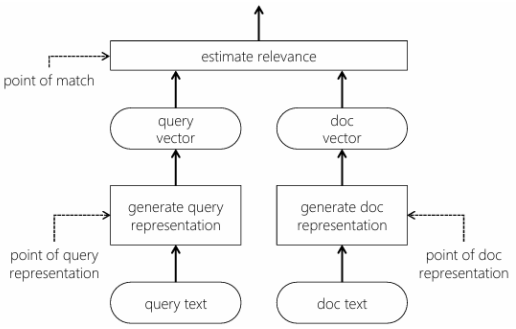
Justin Löscher

Betreut von Maik Fröbe und Lukas Gienapp

Motivation


lebenserwartung katze
herz-fuer-tiere.de
https://herz-fuer-tiere.de › haustiere › katzen › wie-alt-...
Wie alt werden Katzen? Alles zur Lebenserwartung
08.01.2024 — Im Durchschnitt werden **Katzen** 15 Jahre alt. Einige **Katzen** können jedoch auch deutlich älter werden, insbesondere wenn sie in einer sicheren ...
Einflussfaktoren auf die... · Tabelle: Wie alt werden... · Wie äußert sich...
Uelzener
https://uelzener.de › magazin › katze › haeufige-fragen
Wie alt werden Hauskatzen?
04.09.2023 — Wie alt eine **Katze** wird, hängt sehr von der Pflege und den Lebensumständen ab. Ein weiterer wichtiger Faktor ist die Rasse. Je nach Rasse können ...

Neuronale Ranking-Modelle (NRMs)



Einsatz von neuronalen Netzwerken:

- Generieren von **Query-** und **Dokumentrepräsentation**
- **Relevanzfunktion**

Neuronale Ranking-Modelle (NRMs)

Vorteil:

- haben semantisches Verständnis, indem sie Terme, die in einem Kontext stehen, miteinander assozieren können
- gut gegen *vocabulary mismatch*

Beispiele:

- ”australia” $\rightarrow$ ”australia sydney kangaroo”
- “how many people live in germany?” $\rightarrow$ ”population germany”

Neuronale Ranking-Modelle (NRMs)

Nachteile:

- sind schwer interpretierbar
- Overfitting
- empfindlich für adversariale Fehler

Traditionelle Ranking-Modelle wie BM25 haben diese Probleme nicht

Lokale Approximation via Query Reformulation

Idee: Approximiere top-k NRM-Ranking mit einfachen Ranker und reformulierter Query

- liefert interpretierbares Ersatzmodell
- reformulierte Query verbessert Effektivität von einfachen Rankern

Verwandte Arbeiten: BFS-Algorithmus

Llordes et al. [2023]

Gegeben:

- Korpus D, Query q
- NRM Θ
- NRM-Ranking $L_k(q, \Theta)$
- Einfaches Ranking-Modell ϕ (z.B. BM25)
- Ähnlichkeitsmaß ω (z.B. Jaccard-Distanz)

Ziel: Finde eine Query $q^+ \subset V(L_k(q, \Theta))$, sodass $\omega(L_k(q, \Theta), L_k(q^+, \phi)$ maximal ist

Verwandte Arbeiten: BFS-Algorithmus

Llordes et al. [2023]

- Finden einer reformulierten Query q^+ ist die Suche nach einer optimalen Teilmenge aus $V(L_k(q, \Theta)$
- Teilmenge über Traversierung eines Baumes finden
- Anwendung eines Best-First-Search-Algorithmus

Methodik: BFS-Algorithmus

- Ein Knoten ist genau eine mögliche Teilmenge von $V(L_k(q, \Theta)$
- Ausgehend von der aktuell besuchten Query werden *b* Kindknoten generiert
- Entweder wird ein Term hinzugefügt, oder ein Term wieder entfernt
- Die Kindknoten, die den besten Korrelationswert ω erreichen, werden zu einer Priority Queue hinzugefügt

Methodik: BFS-Algorithmus

- Algorithmus startet bei einer leeren Query und terminiert, sobald eine optimale Query q^+ gefunden wurde oder keine Elemente mehr in der queue sind
- Hinzufügen eines Terms orientiert sich an der **RM3**-Gewichtung
- Entfernen von Termen orientiert sich an **TF-IDF**-Gewicht

Methodik: BFS-Algorithmus

Query q = ”lebenserwartung katze”

NRM-Ranking:

docno	text
d_1	”Wie alt werden Katen? Im Durchschnitt werden Katzen 15 Jahre alt.”
d_2	”Das Alter einer Hauskatze, hängt von der Pflege und Lebensumständen ab.”
d_3	”Die durchschnittliche Lebenserwartung von Katzen liegt bei etwa 15 Jahren.”
d_4	”Im Allgemeinen werden Katzen im Durchschnitt 14 bis 16 Jahre alt.”
d_5	”Katzen können ziemlich alt werden. Die derzeit älteste Katze ist 27 Jahre alt.”

Methodik: BFS-Algorithmus

Ziel: Reformuliere Query q so, dass das Top-3-Ranking von BM25 mit dem des NRM übereinstimmt.

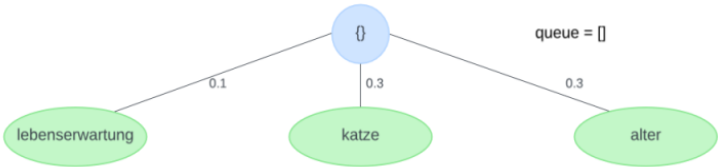
NRM(q)	BM25(q)
d_1	**d_2**
d_2	**d_4**
d_3	**d_1**
d_4	d_5
d_5	d_3

Methodik: BFS-Algorithmus

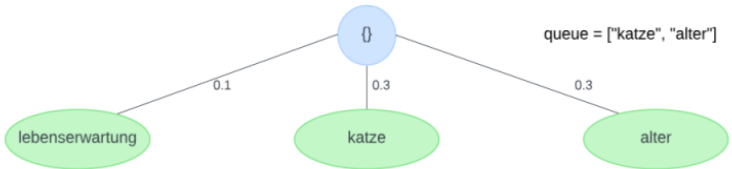
docno	text
d_1	”Wie alt werden Katen? Im Durchschnitt werden Katzen 15 Jahre alt.”
d_2	”Das Alter einer Hauskatze, hängt von der Pflege und Lebensumständen ab.”
d_3	”Die durchschnittliche Lebenserwartung von Katzen liegt bei etwa 15 Jahren.”

$V(\{d_1,d_2,d_3\})$ = {”katze”, ”lebenserwartung”, ”alter”, ”durchschnitt”, ”15”,...}

Methodik: BFS-Algorithmus

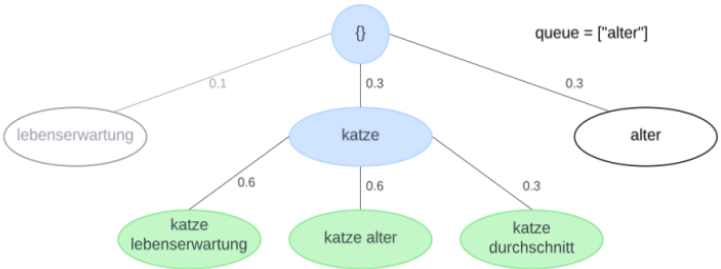


Methodik: BFS-Algorithmus

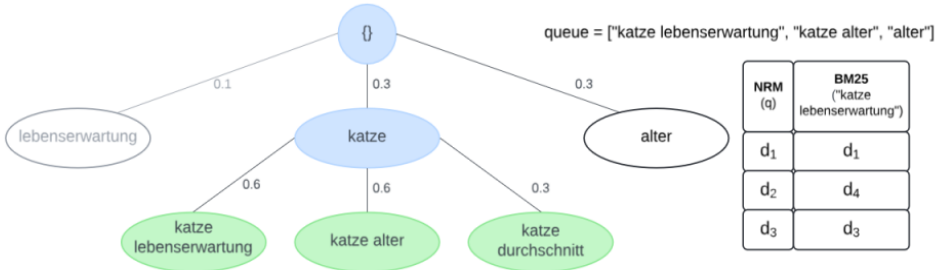


NRM (q)	BM25 ("katze")
d_1	d_4
d_2	d_2
d_3	d_5

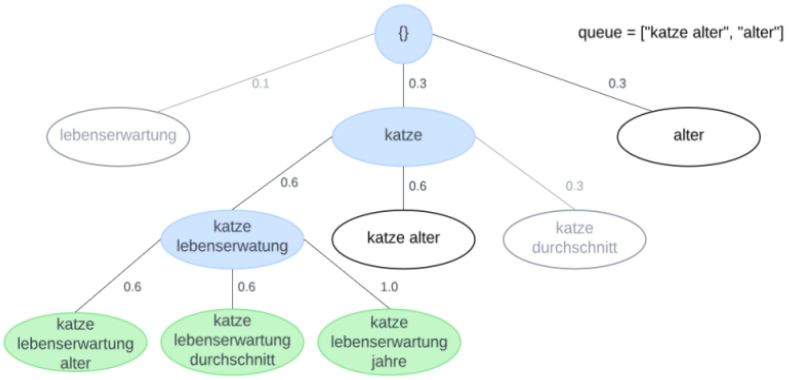
Methodik: BFS-Algorithmus



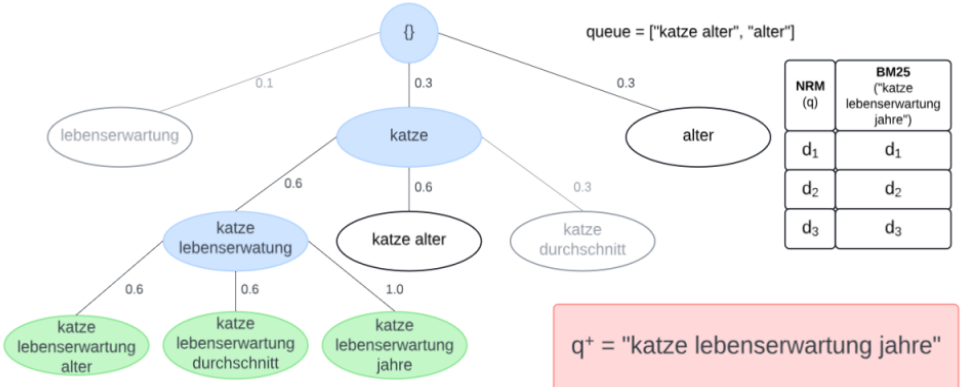
Methodik: BFS-Algorithmus



Methodik: BFS-Algorithmus



Methodik: BFS-Algorithmus



Methodik: Eigener Ansatz

Problem

- BFS-Ansatz bisher nur auf MS-MARCO Passage Ranking Collection angewendet
- Webdokumente enthalten deutlich mehr Terme als Passagen
- Query q^+ mit guter Korrelation wird schnell gefunden und viele weitere Queries, die danach generiert wurden, sind auf dem selben Niveau
- Folge: Länge der queue steigt immer weiter, was enorme Kosten verursacht

Lösung: Terminiere Algorithmus sobald queue eine Maximallänge erreicht hat

Methodik: Eigener Ansatz

Varianten

1. Starte mit leerer Query
2. Starte mit RM3-expanded Query

Experiment: Forschungsfragen

1. Wie gut lassen sich NRM-Rankings mit dem BFS-Algorithmus approximieren?
2. Kann die Effektivität von einfachen Rankern mit den reformulierten Queries verbessert werden?
3. Inwiefern variiert die Effektivität der reformulierten Queries unter sich verändernden Datensätzen?

Experiment: Daten

MS-MARCO passage ranking collection

- 8.8 Millionen Passagen
- TREC DL 2019 topic set (43 Queries)

TREC Robust 2004

- 528.155 Dokumente (Zeitungsartikel)
- 250 Queries

Experiment: Setup

Neuronales Ranking-Modell: MonoT5

Baselines

- BM25
- Bo1
- RM3

Eigene Ansätze

- BFS I = Start mit leerer Query
- BFS II = Start mit RM3-expanded Query (6 Terme)

Experiment: Setup

BFS-Parameter

- Einfacher Ranker: BM25
- Rankinglänge: 10
- Anzahl generierter Kindknoten: 30
- Maximale Query-Terme: 10
- Maximale Queue-Länge: 1000

Experiment: Setup

Ähnlichkeit

- Jaccard-Distanz
- RBO

Effektivität

- $nDCG_{10}$
- MAP

Experiment: Ergebnisse

RQ1

	MS-MARCO		Robust04	
	Jaccard	*RBO*	*Jaccard*	*RBO*
BM25	0.1914	0.2049	0.2642	0.3303
Bo1	0.3069	0.4333	0.3328	0.4791
RM3	0.3144	0.4126	0.3440	**0.5001**
BFS I	0.6149	**0.4810**	0.4535	0.3579
BFS II	**0.6197**	0.4762	**0.5819**	0.4553

Table 1: Durchschnittliche Korrelationen aller Queries der top-10 Rankings

Experiment: Ergebnisse

RQ2

	MS-MARCO		Robust04	
	$nDCG_{10}$	*MAP*	*$nDCG_{10}$*	*MAP*
MonoT5	0.7238	0.5246	0.5450	0.3121
BM25	0.4795	0.3700	0.4244	0.2369
Bo1	0.6352	**0.4847**	0.5114	**0.3018**
RM3	0.6363	0.4732	0.5087	0.2966
BFS I	0.6463	0.3658	0.5104	0.2521
BFS II	**0.6598**	0.3915	**0.5277**	0.2745

Experiment: Beispiel-Queries

Query Typ	Beispiel-Queries	*Jaccard*	*RBO*	$nDCG_{10}$
Original	school prayer banned	0.4286	0.7457	0.7460
BFS	school prayer ban court footbal vote georgia religi legal	0.8182	0.8793	0.8764

Experiment: RQ3

LongEval

- Dokumentkollektionen erfasst zu verschiedenen Zeitpunkten
- 3 Datensätze aus Webdokumenten aus den Monaten Juni, Juli und September (jeweils ca. 1 Mio Dokumente)
- 129 Queries

Vorgehen

1. Reformuliere Queries mit MonoT5-Rankings auf Datensatz von Juni
2. Evaluiere Effektivität der Queries auf Datensätzen von Juli und August

Experiment: Ergebnisse

RQ3

	Juni	Juli	September
BFS I	0.1612	0.1472	0.1330
BFS II	0.1770	0.1566	0.1493

Table 2: Effektivität der reformulierten Queries mit $nDCG_{10}$

Fazit

- Approximation der Rankings klappt für manche Queries sehr gut, bei anderen aber auch eher schlecht
- Effektiv sind sie trotzdem mit bis zu 97% der Effektivität von MonoT5
- Über einen längeren Zeitraum verlieren die reformulierten Queries an Effektivität