東京理科大学 大学院講義

第4回 情報検索特論
（スコアリング・用語重み付け・ベクトル空間モデル）

東京理科大学　創域理工学部　情報計算科学科
植松 幸生

本授業の目標とシラバスの確認

目標：情報検索/LLMの概念を理解し，実装レベルまで習得する

1 講義概要　講義概要の説明
2 特別講義　Federated Learning: Mei Kobayashi(5限なので，出れない方は応相談)
3 情報検索基礎　インデックス1　全文検索の仕組みを理解する
4 **情報検索基礎　スコアリング1　TFIDF等のscoring技術の解説**
5 情報検索基礎　インデックス2　インデックスの圧縮技術を学ぶ
6 情報検索応用　LLMを用いた情報検索　LLMを用いた情報検索の仕組みを理解する
7 情報検索応用　LLMを用いた情報検索２　第６回の授業の実装を学ぶ/RAGの実装を学ぶ
8 情報検索応用　LLMを用いた情報検索4　LLMのファインチューニングについて学ぶ
9 情報検索応用　応用例　実データを用いたEDAを学ぶ((Exploratory Data Analysis)
10 情報検索応用　応用例 実データを用いた情報検索技術の応用例
11 情報検索実践　情報検索とLLMを応用したシステムの開発１　自分で考えたシステムを実装するための方法を理解する
12 情報検索実践　情報検索とLLMを応用したシステムの開発2　システムを実装する/プレゼンテーションやエレベータピッチの方法を学ぶ
13 情報検索実践　プレゼンテーション1　実装したシステムをデモを通じたプレゼンテーションを行う
14 情報検索実践　プレゼンテーション2　引き続きプレゼンテーションを行う．また，他の学生の発表を評価する
15 情報検索実践　情報検索最新動向（講演）

本日のアジェンダ

この講義の範囲

- 先週のおさらい
- 本日の最終目標の共有
- ランク付け検索（ベクトル空間スコアリング）
  - 文書スコアリング
  - 単語頻度（tf）
- まとめ

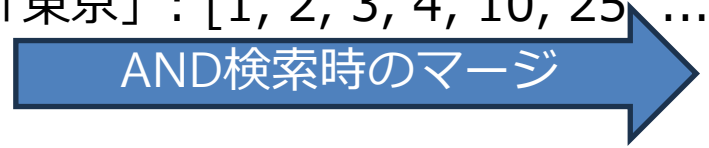
前回のおさらい

何人かの人に，コード追うには時間がなさすぎ
インデックスが何をやっているかよくわからない
・・・というご意見を頂いたので振り返ります！

「先生」: [1, 3, 7, 20, ...]
「東京」: [1, 2, 3, 4, 10, 25, ...]
・・・
・・・
「ゼブラ」: [1, 2, 3, 4, 10, 25, ...]

配列とかパトリシア木はこちら

「先生」: [1, 3, 7, 20, ...]
「東京」: [1, 2, 3, 4, 10, 25, ...]



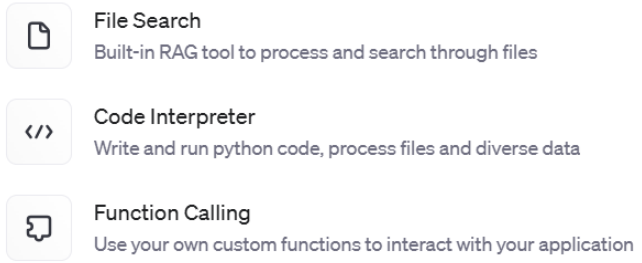
Posting listのskip listの話はこちら

先週のプログラムをもう一度見てみよう！

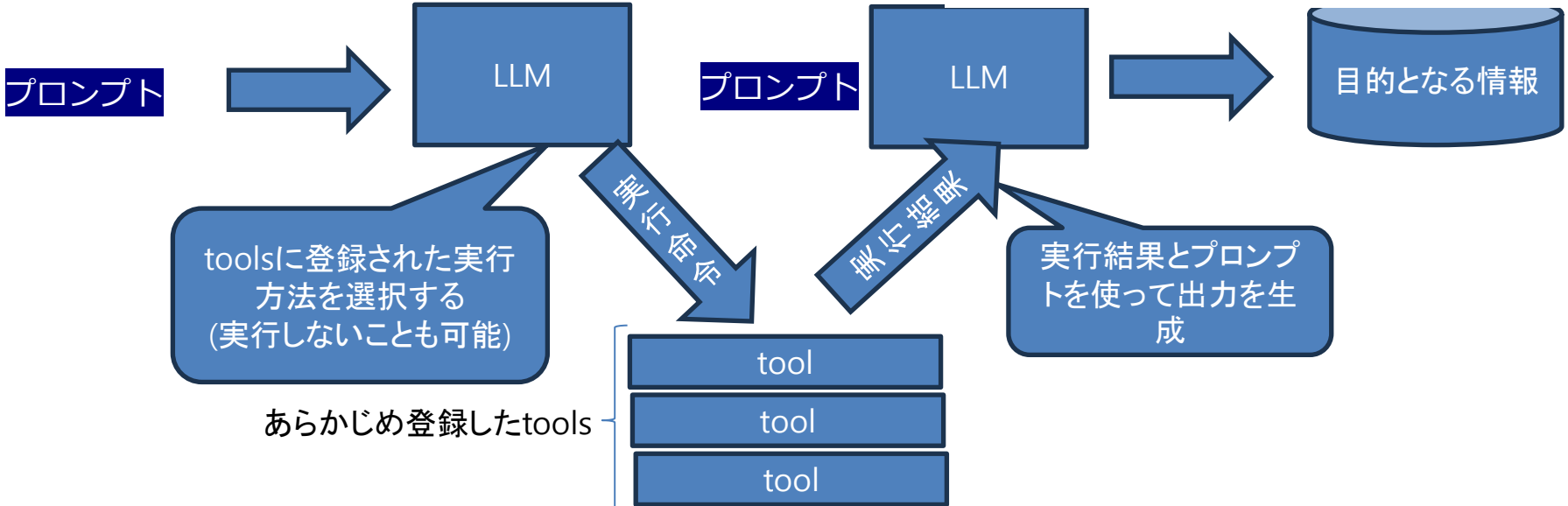
LLMを利用したRAG等のtools実行

Toolsとは，LLMが出来ないことを外部から与えて実行する機能です

https://platform.openai.com/docs/assistants/tools



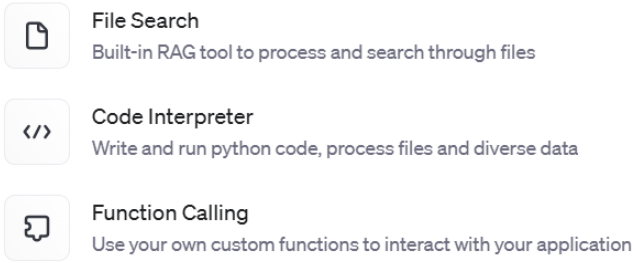
API toolsの動作フロー



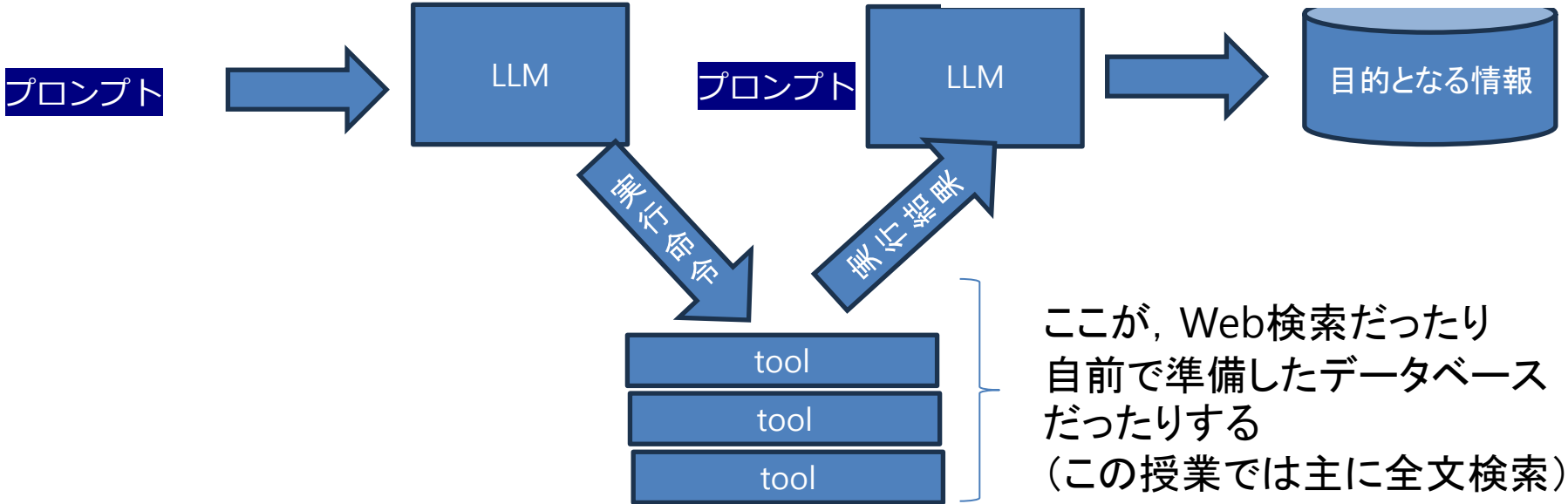
LLMを利用したRAG等のtools実行

Toolsとは，LLMが出来ないことを外部から与えて実行する機能です

https://platform.openai.com/docs/assistants/tools



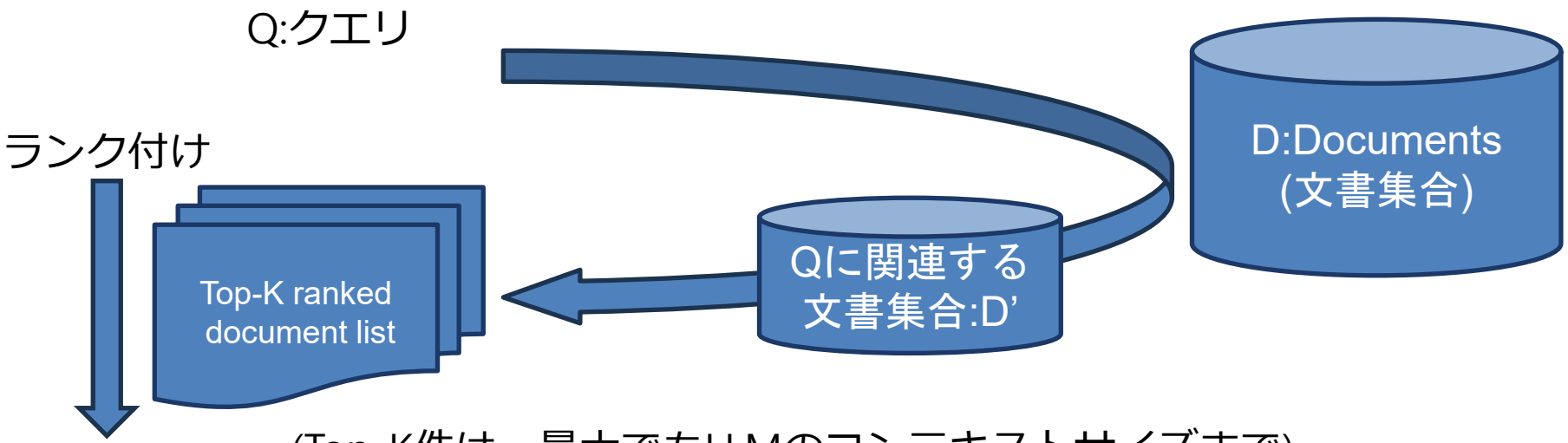
API toolsの動作フロー



ここが，Web検索だったり
自前で準備したデータベース
だったりする
（この授業では主に全文検索）

今週最終的に学ぶこと

あるクエリ(Q)に対する，文書(D)の**スコア付けの方法**に関して学ぶ
最終的に必要なのは，クエリQに関連するTopK件の文書リスト



(Top-K件は，最大でもLLMのコンテキストサイズまで)

ランク付け検索（Ranked Retrieval）

答えの記載があれば極端な話1文書で構わない！
(10000件の検索結果は必要がない)

- これまでの講義では、クエリは主にブール式（AND/OR/NOT）
- 文書は「合致する/しない」の二値判定
  - Ex.) ある文書に「先生」という単語が存在する
- 多くのユーザは何千件もの結果を読みたくない（特にWeb検索）
- たとえ同じ情報でも信頼できるソースの情報が見たい

ブール検索の課題：feast or famine

- ブールクエリは「少なすぎる（=0）」または「多すぎる（数千）」に陥りがち
- 例（大量）："東京 OR 先生" → 何十万件
- 例（0件）："東京 AND 先生 AND 研究 AND 交通政策" → 0件
- 適切な件数に調整するには高度なスキルが必要
- AND は件数が少なくなりがち／OR は多くなりがち

ランク付け検索モデル

- ランク付けでは、集合ではなく「並び順（上位文書の順序）」を返す

- Jaccard係数
- Bag-of-words

最初の一歩： Jaccard係数

- 二つの集合 A, B の重なり度合い
- Jaccard(A,B) = |A ∩ B| / |A ∪ B|
- Jaccard(A,A)=1、A∩B=∅なら0、常に0～1の範囲
- 集合のサイズが異なっても定義できる

Jaccard係数：東京／先生の例

- クエリ：東京 先生
- 文書1：東京の大学の先生が研究発表
- 文書2：大阪の先生が講演
- （集合として語の有無のみを考える：頻度は無視）
- → クエリ語の両方を含む文書1のスコアが高い

Jaccard の課題

- 単語頻度（term frequency）を考慮しない
- コレクションで稀な語ほど情報量が高いが、反映できない
- 文書長に対する適切な正規化がない

クエリと文書のマッチング・スコア

TF-IDF: Term Frequency and Inverse Document Frequency

- 一語クエリから考える：語が出現しなければスコア0
- 文書内の出現回数が多いほどスコアは高くすべき
- この考えを多語クエリに拡張（tf, idf, tf-idf へ）

復習：二値の語-文書インシデンス行列

各文書を {0,1}^{|V|} の二値ベクトルで表現（語彙 V）

例：語が出たら1、出なければ0

語	文書A	文書B	文書C	文書D	文書E	文書F
東京	1	1	0	0	0	1
先生	1	1	0	1	0	0
研究	1	0	0	1	1	1
大学	0	1	0	0	0	0
講演	1	0	0	0	0	0
やさしさ	1	0	1	1	1	1

語-文書カウント行列（出現回数ベース）

語 t の文書 d における出現回数を用いる

各文書は ℕ^{|V|} のカウントベクトル（右表の列）

語	文書A	文書B	文書C	文書D	文書E	文書F
東京	15	7	0	0	0	0
先生	4	15	0	1	0	0
研究	23	22	0	2	1	1
大学	0	10	0	0	0	0
講演	5	0	0	0	0	0
やさしさ	2	0	3	5	5	1

Bag of Words（語順を無視するモデル）

- ベクトル表現では語の並び（順序）を考慮しない
- 「東京の先生は優しい」と「優しい先生は東京にいる」はほぼ同じベクトル
- 位置情報を使う位置索引（positional index）なら区別できる

順序性がない

Term Frequency

- tf(t,d)：語 t が文書 d に出現した回数
- 検索の適合度計算に tf を使いたいが、単純な比例は望ましくない
- 10回出た文書は1回より関連が高いが、10倍とは限らない

$$\mathrm{TF}(t,d) = \begin{cases} 1 + \log(f_{t,d}) & (f_{t,d} > 0) \\ 0 & (f_{t,d} = 0) \end{cases}$$

付録：TFアタック（頻度スパム）と対策

- 攻撃：同じ語を文書中に過度に繰り返し、tfを人工的に増やして順位を上げる手法
- 単純なtf比例スコアは脆弱（例："東京 東京 東京 … 先生"）
- 主な対策：
  - ・対数tf（1+log(tf))やBM25の飽和関数でtfの利得を逓減
  - ・文書長正規化（長いだけの羅列を不利に）
  - ・スパン/セクション多様性（同一セクションの反復を割引）
  - ・品質/スパムシグナル（リンク、言語モデル判定、重複率）
  - ・CAPTCHA/投稿制限など運用側の対策

希少語は情報量が高い

- コレクションで稀な語ほど、出現は関連性の手掛かりになりやすい
- 例：クエリに「王来王家（おくめか）」が含まれると、その語を含む文書は重要度が高い
- → 希少語に高い重みを与えたい（idf）

$$\mathrm{IDF}(t, D) = \log \frac{N}{n_t}$$

N: コーパス全体(D)の文書数
n_t: 語 t を含む文書の数

コレクション頻度（cf）と文書頻度（df）

- cf_t：コレクション全体での出現回数
- df_t：語 t を含む文書数（df ≤ N）
- どちらが検索に有用か？ → 一般に df が良い重みづけ指標

語	cf（総出現回数）	df（文書数）
東京	10440	3997
先生	10422	8760

TF・IDF
(Term Frequency Inverse Document Frequency)

この単語の頻度と単語の珍しさを掛け合わせてTFIDFという重み（単語の重要度）を計算します

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

このtfidf値を重みとしたベクトルを使って，文書dとqの類似度を計算します．それが**コサイン類似度**です

$$w_{t,d} = \text{TF}(t, d) \times \text{IDF}(t, D)$$

$$\vec{d} = (w_{1,d}, w_{2,d}, \ldots, w_{T,d})$$

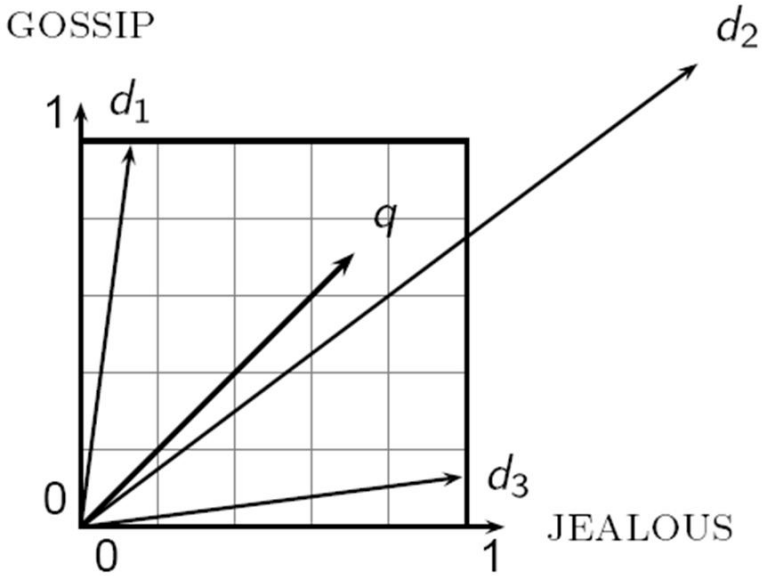
$$\vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{T,q})$$

$$\text{sim}(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{\|\vec{q}\| \, \|\vec{d}\|} = \frac{\sum_t w_{t,q} \, w_{t,d}}{\sqrt{\sum_t w_{t,q}^2} \sqrt{\sum_t w_{t,d}^2}}$$

qとdのベクトルの方向が近い(0度だと1になる)ほど類似している

コサイン類似度

qとdの角度Θが小さいほど大きな値になる



$$\text{sim}(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{\|\vec{q}\| \, \|\vec{d}\|} = \frac{\sum_t w_{t,q} \, w_{t,d}}{\sqrt{\sum_t w_{t,q}^2} \sqrt{\sum_t w_{t,d}^2}}$$

Introduction to information retrieval より抜粋

コサイン類似度を計算する疑似コード

```
CosineScore(q)
 1  float Scores[N] = 0
 2  float Length[N]
 3  for each query term t
 4  do calculate w_{t,q} and fetch postings list for t
 5      for each pair(d, tf_{t,d}) in postings list
 6      do Scores[d] += w_{t,d} × w_{t,q}
 7  Read the array Length
 8  for each d
 9  do Scores[d] = Scores[d]/Length[d]
10  return Top K components of Scores[]
```

実は順序がないため先週学んだposting Listにtfの情報を持たせれば計算可能

Introduction to information retrieval より抜粋

BM25 (Best Matching 25について)

Tfidfのコサイン類似度に対して，良く使われる手法としてOkapi IR (BM25)というRobertsonらが開発したスコアリング方式があります

$$\mathrm{BM25}(q,d) = \sum_{t \in q} \mathrm{IDF}(t) \cdot \frac{f_{t,d}(k_1+1)}{f_{t,d} + k_1\left(1 - b + b \cdot \frac{|d|}{\mathrm{avgdl}}\right)}$$

fはtf

文書の長さを，文書集合全体の
平均文書長で正規化している

Stephen E. Robertson; Steve Walker; Susan Jones; Micheline Hancock-Beaulieu & Mike Gatford (November 1994). Okapi at TR EC-3. Proceedings of the Third Text REtrieval Conference (TREC 1994). Gaithersburg, USA.

BM25っておいしいの？

恐ろしいことに30年以上たった今でも使われています。。。。

Model (→)	Lexical	Sparse			Dense				Late-Interacti	
Dataset (↓)	BM25	DeepCT	SPARTA	docT5query	DPR	ANCE	TAS-B	GenQ	ColBERT	
MS MARCO	0.658	0.752‡	0.793‡	0.819‡	0.552	0.852‡	**0.884**‡	**0.884**‡	0.865‡	
TREC-COVID	0.498*	0.347*	0.409*	**0.541***	0.212*	0.457*	0.387*	0.456*	0.464*	
BioASQ	**0.714**	0.699	0.351	0.646	0.256	0.463	0.579	0.627	0.645	
NFCorpus	0.250	0.235	0.243	0.253	0.208	0.232	**0.280**	**0.280**	0.254	
NQ	0.760	0.636	0.787	0.832	0.880‡	0.836	0.903	0.862	**0.912**	
HotpotQA	0.740	0.731	0.651	0.709	0.591	0.578	0.728	0.673	**0.748**	
FiQA-2018	0.539	0.489	0.446	0.598	0.342	0.581	0.593	**0.618**	0.603	
Signal-1M (RT)	**0.370**	0.299	0.270	0.351	0.162	0.239	0.304	0.281	0.283	
TREC-NEWS	0.422	0.316	0.262	**0.439**	0.215	0.398	0.418	0.412	0.367	
Robust04	**0.375**	0.271	0.215	0.357	0.211	0.274	0.331	0.298	0.310	
ArguAna	0.942	0.932	0.893	0.972	0.751	0.937	0.942	**0.978**	0.914	
Touché-2020	0.538	0.406	0.381	**0.557**	0.301	0.458	0.431	0.451	0.439	
CQADupStack	0.606	0.545	0.521	0.638	0.403	0.579	0.622	**0.654**	0.624	
Quora	0.973	0.954	0.896	0.982	0.470	0.987	0.986	0.988	**0.989**	
DBPedia	0.398	0.372	0.411	0.365	0.349	0.319	**0.499**	0.431	0.461	0.398
SCIDOCS	0.356	0.314	0.297	**0.360**	0.219	0.269	0.335	0.332	0.344	0.356
FEVER	0.931	0.735	0.843	0.916	0.840	0.900	**0.937**	0.928	0.934	0.931
Climate-FEVER	0.436	0.232	0.227	0.427	0.390	0.445	**0.534**	0.450	0.444	0.436
SciFact	0.908	0.893	0.863	**0.914**	0.727	0.816	0.891	0.893	0.878	0.908

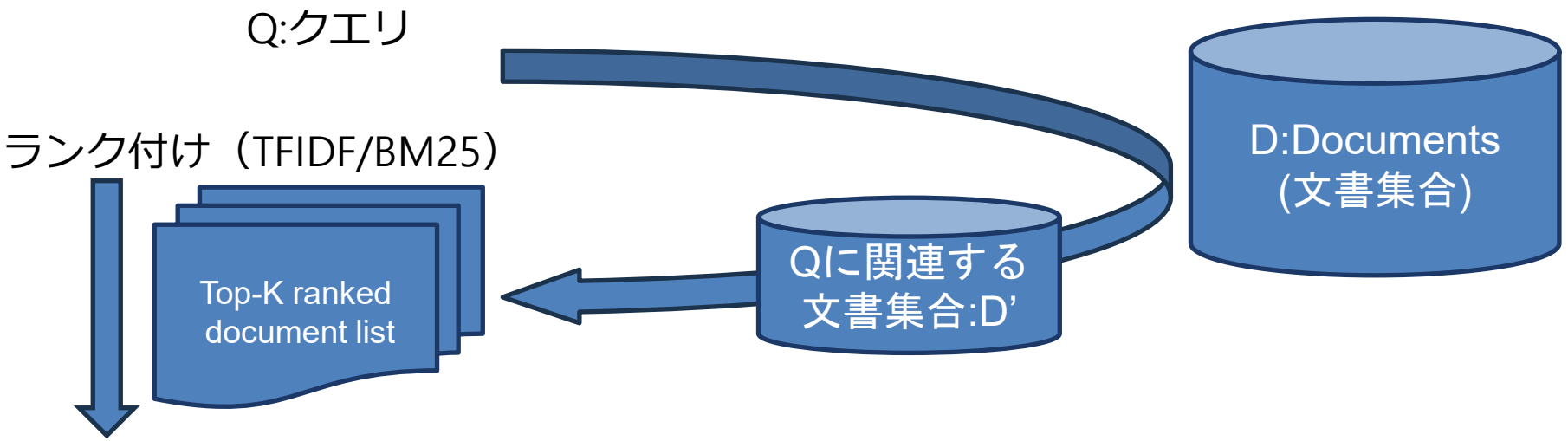
Table 9: In-domain and zero-shot retrieval performance on BEIR datasets. Scores denote **Recall@100**. The best retrieval performance on a given dataset is marked in **bold**, and the second best performance is underlined. ‡ indicates in-domain retrieval performance. ⋆ shows the capped Recall@100 score (Appendix G).

https://learn.microsoft.com/ja-jp/azure/search/index-ranking-similarity

https://github.com/beir-cellar/beirより抜粋

まとめ

あるクエリ(Q)に対する，文書(D)の**スコア付けの方法**に関して学ぶ
最終的に必要なのは，クエリQに関連するTopK件の文書リスト



I am a THINKER!

Thank you!

Contact:

Yukio Uematsu

yukio@rs.tus.ac.jp

yukio@cs.stanford.edu