ベクトル検索におけるテキスト構造化の効果分析

梶川 怜恩 [1,2]　神田 峻介 [1]　赤部 晃一 [1]　小田 悠介 [1,3]
[1] シェルパ・アンド・カンパニー株式会社 [2] 愛媛大学大学院 理工学研究科
[3] 国立情報学研究所 大規模言語モデル研究開発センター
{reon.kajikawa, shunsuke.kanda, koichi.akabe, yusuke.oda}@cierpa.co.jp
reon@ai.cs.ehime-u.ac.jp　odashi@nii.ac.jp

概要

ベクトル検索において，検索対象のテキスト同士の特徴が異なる状況で同じ埋め込みモデルを用いると，埋め込み空間上で離れた位置に写像されてしまうために正しく検索できなくなる問題がある．本研究ではこの問題に対し，構造面でのテキスト間の相違を事前に緩和することで検索精度を改善できる，という仮説の下，検索対象のテキストに対して事前に指定したフォーマットに基づいた構造化を行う．実験的分析の結果，提案手法を適用しない場合と比べて，検索精度が改善することを確認した．加えて，構造化によってテキストに含まれる特徴量を解釈可能な情報として抽出し，検索に有用な特徴量を特定できるという異なる利点についても分析した．

1　はじめに

情報検索 [1] とは，膨大なデータの中からユーザの要望に合致する情報を抽出する技術である．テキスト検索は特に文字列に関する情報検索技術を指し，文字列間の一致度や意味的関連性に基づいて該当文書を特定するものである．テキスト検索の手法としてキーワード検索とベクトル検索の 2 種類に大別される．キーワード検索は，クエリ（検索文）と検索対象の文書に含まれる文字列の表層的な一致度に基づいて該当文書を予測する手法である．ベクトル検索 [2] では，埋め込みモデルを用いてテキストを数値ベクトルに変換し，ベクトル間の類似度に基づいて該当文書を予測する手法である．ベクトル検索は文字列の表層的な一致を超えて意味的に関連のある情報を効率的に検索できる一方で，クエリと文書に対して同じ埋め込みモデルを用いる場合，それぞれのテキストの構造が異なると検索精度が低下するという課題がある．

本研究では，大規模言語モデル (LLM) を用いた前処理によって対象のテキストを加工し，テキスト同士の関連性を強化する手法を提案する．図 1 に示すように，テキストに含まれるタスクに重要な情報を列挙する構造化を行うことで，クエリと文書の比較を同一構造の文の上で行うことができるようにし，これによる検索精度の改善を検証する．

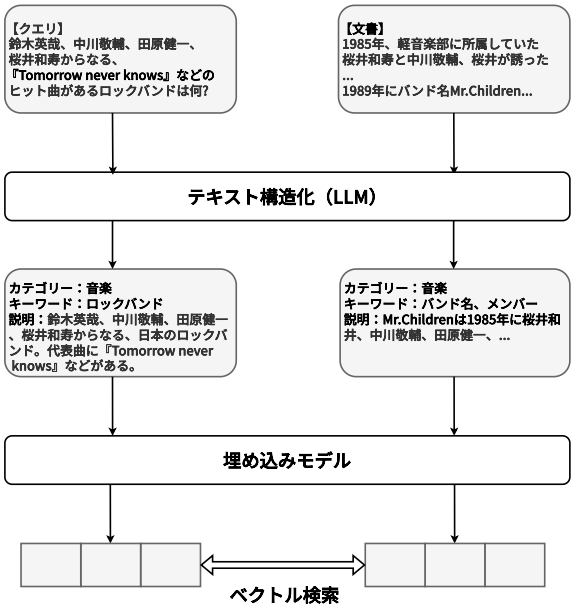

図 1　提案手法の概要図

質問応答ベンチマークを用いた実験的分析により，以下のことが確認された．

- LLM を用いた構造化により，前処理を適用しない場合と比べて，検索精度が改善した．
- LLM を用いた構造化では，純粋な文書構造の変更に加えて，疑似的な回答生成が発生し，この現象による性能向上への寄与が大きかった．
- 構造化によって，テキストに含まれる特徴量を人間が解釈可能な情報として抽出し，検索に有用な特徴量を特定できることを確認した．

2 関連研究

2.1 テキスト検索における前処理

テキスト検索タスクにおいて，クエリの前処理によって精度改善を行なう手法はよく研究されている．既存研究においてこれらの手法が期待する主な効果は，質問意図や内容の補強，および曖昧性の解消など，クエリそのものに内在する問題の解消である [3]．具体的な手法としては，クエリとその書き換え文でパラレルコーパスを構築し，このコーパスを用いて変換器を学習する手法 [4]，LLM によるゼロショット生成能力を活用し，LLM にクエリの書き換えを行わせる手法 [5] が存在する．また質問応答タスクにおいては，クエリの明確化のための変換だけでなく，クエリから回答文に変換する手法による精度改善も報告されている．HyDE [6] はこの考えに基づいたモデルであり，LLM を用いてクエリから擬似的な回答文を生成し，その回答文を実際の検索器へ入力する．HyDE で生成された回答文は本来のクエリと比べて検索対象の回答文と構造が近似することになり，真の回答と疑似的な回答の類似性に基づいて検索を行うことが可能となる．

2.2 埋め込みモデルによるテキスト検索

埋め込みモデルはテキストを数値ベクトルに変換するモデルであり，ベクトル間の類似度を介して，文の表層的な情報を超えたテキスト間の意味的な類似性を調べるツールとして用いられる [7]．ところがテキスト検索においては，他の自然言語処理タスクに埋め込みモデルを適用した場合と比較して埋め込みモデルによる効果が低いことが知られている [8]．これは HyDE の説明でも触れたように，テキスト検索では検索クエリと検索対象間の文書としての構造や性質が異なり，両者を同一のモデルでベクトル化した場合に類似度が低くなることが原因と考えられている [6]．一例として質問応答タスクで回答資料の収集にベクトル検索を用いる場合，クエリは疑問文，検索される文書は何らかの記事となる．このため簡潔な文と複雑な文書の間の類似性を調べることとなり，単純なベクトル化手法では本来重視すべき類似性を正確に捉えることが困難となり，結果として検索精度の低下を引き起こす [6]．

この問題の解決策として，クエリと検索対象で異なる方針のベクトル化を行うように学習された検索特化モデルが提案されている [9–11]．これらの手法は埋め込みに起因する問題を機械学習的に直接解決する一方で，モデルを新しく学習するためのデータ，計算資源，学習時間，ひいてはこれらに起因する金銭的コストの確保が必要となる．このため計算機事業者が汎用的なモデルを大規模に展開する用途には有効だが，タスクレベルの細かなチューニングを頻繁に行う必要がある場面には向かない．本研究では，このような専用の追加学習については考慮せず，一般的な埋め込みモデルを利用したテキスト検索精度の向上を対象とする．

3 提案手法

埋め込みモデルにおけるテキスト検索タスクの精度を改善するために，対象のテキストをテキスト構造化 [12] により一定のフォーマットに整形する前処理手法を提案する．提案手法の概要を図 1 に示す．本研究ではクエリだけでなく，検索対象の文書に対してもテキスト構造化を適用し，クエリと文書の両者を同一の構造を持つテキストとして扱えるようにする．まず，テキストを LLM により任意のデータ構造に変換する．本実験では，表現が簡単な “属性-値” 形式の辞書構造を採用する．属性も任意の要素が指定可能であるが，本実験ではたたき台として “カテゴリー”，“キーワード”，“説明” の 3 つの属性を採用する．検索にとって適切な属性の設計は今後の課題とする．本研究で使用するモデルは JSON フォーマットによる出力制約を課すことができるため，変換後の情報は JSON として取得する．本実験で使用したプロンプトを図 2 に示す．ここで入力文の内容や LLM の挙動によっては，期待する値がうまく抽出できない可能性がある．このような場合，LLM の内部知識を元に値を自動生成するものとした．

以上の処理で生成された辞書をテキストとして書き下し，この結果を埋め込みモデルで処理することで，最終的な埋め込みベクトルを取得する．書き下しでは単純に属性-値のペアを “{ 属性 }: { 値 }” のように整形し，それぞれを改行で連結したものを使用した．

4 評価実験

ここでは日本語のテキスト検索タスクにおける評価を通して，提案手法の有効性を検証する．

表 1　日本語のテキスト検索タスクにおける実験結果．**太字**は最高精度を示し，#Params. はモデルのパラメータ数を示す．◇が付記されたモデルは質問応答データセットで学習されたモデルである．

		ベースライン		提案手法		HyDE		HyDE+提案手法	
Model	#Params.	nDCG@10	MRR@10	nDCG@10	MRR@10	nDCG@10	MRR@10	nDCG@10	MRR@10
BM25	-	0.458	0.702	0.620	0.814	0.620	0.814	**0.627**	**0.819**
教師なし SimCSE base	111M	0.312	0.521	0.492	0.710	0.475	0.721	**0.505**	**0.728**
SimCSE large	337M	0.393	0.625	0.502	0.725	0.520	**0.758**	**0.528**	0.750
教師あり SimCSE base	111M	0.324	0.541	0.457	0.682	**0.490**	**0.726**	0.467	0.698
SimCSE large	337M	0.357	0.573	0.484	0.717	**0.512**	**0.741**	0.491	0.726
GLuCoSE base◇	133M	0.309	0.519	0.550	0.750	0.567	0.785	**0.582**	**0.792**
Sarashina-Embedding◇	1,224M	0.630	0.816	0.704	0.858	0.608	0.785	**0.719**	**0.873**
text-embedding-3 small	-	0.388	0.611	0.588	0.772	**0.651**	**0.828**	0.628	0.808
text-embedding-3 large	-	0.566	0.782	0.658	0.827	**0.714**	**0.882**	0.684	0.851

表 2　検索結果の事例．負例は×，正例は○を示す．モデルは教師あり SimCSE-large.

構造化なし（ベースライン）
Q. 日本の市の中で唯一、名前のすべてが漢数字のみで構成されている、高知県の市は何でしょう?
top 1: × 土佐市 (とさし) は、高知県中部に位置する市。面積は県内 11 市の中で最小となっている。
top 2: × 高知市 (こうちし) は、高知県の中部に位置する市。高知県の県庁所在地及び最大の都市で、中核市に指定されている。旧土佐郡・長岡郡・吾川郡。
top 3: × 南国市 (なんこくし) は、高知県の市である。高知県の中東部に位置し、県庁所在地の高知市の東に隣接する。
...
top11: ○ 四万十市 (しまんとし) は、高知県南西部に位置する市。
構造化あり（提案手法）
Q. カテゴリー：日本の地理 ↵ キーワード：高知県の市, 漢数字の名前 ↵ 説明：高知県には、名前がすべて漢数字で構成されている市が 1 つあります。
top 1: × カテゴリー：地理 ↵ キーワード：土佐市 ↵ 説明：土佐市 (とさし) は、高知県中部に位置する市。面積は県内 11 市の中で最小となっている。
top 2: ○ カテゴリー：地理 ↵ キーワード：四万十市 ↵ 説明：四万十市（しまんとし）は、高知県南西部に位置する市です。
top 3: × カテゴリー：地理 ↵ キーワード：南国市 ↵ 説明：南国市（なんこくし）は、高知県の市である。高知県の中東部に位置し、県庁所在地の高知市の東に隣接する。

4.1　実験設定

テキスト構造化　テキスト構造化のための LLM として Gemini 1.5 flash[1] [13] を採用した．なお，Gemini の安全性フィルタなどが原因でモデルが正常に返答を返さなかったものについては除外し，以降の評価には用いないものとした．これは事例全体の 0.04%に相当する．

埋め込みモデル　日本語の埋め込みモデルとして，日本語 SimCSE [14]，GLuCoSE[2]，Sarashina-Embedding[3] を用いた．SimCSE は，Wikipedia や NLI データセットから対照学習を用いて訓練されたモデルである [15]．GLuCoSE と Sarashina-Embedding は，質問応答データセット [16] で追加学習されたモデルである．Sarashina-Embedding は LLM ベースのモデルである [17]．また商用モデルとして，OpenAI 社の text-embedding-3[4] を用いた．埋め込み同士の類似度にはコサイン類似度を用いた．

キーワード検索　キーワード検索手法である BM25 を用いた場合の性能を評価した．分かち書きには MeCab [18] を利用した．

評価方法　JQaRA データセット [19] を用いて日本語のテキスト検索タスクを実験した．JQaRA は回答に有用な記事の検索を評価するために設計されており，クエリは AI 王公式配布データセット [20]，文書は Wikipedia の記事で構成されている．本実験の評価に用いたクエリは 1,667 件，文書は 166,629 件である．評価指標には nDCG および MRR を採用し，ranx [21] を用いて評価を実施した．

4.2　実験結果

実験結果を表 1 に示す．ここでは，左二列の「ベースライン」と「提案手法」に着目する．全ての埋め込みモデルについて，nDCG と MRR の両方の指標で提案手法はベースラインを上回っている．SimCSE モデルについては，最大で nDCG が 18%，MRR が 19%改善している．質問応答データセットで学習された GLuCoSE と Sarashina-Embedding や，商用モデルである text-embedding-3 についても，最

1）https://ai.google.dev/gemini-api/docs/models/gemini#gemini-1.5-flash
2）https://huggingface.co/pkshatech/GLuCoSE-base-ja
3）https://huggingface.co/sbintuitions/sarashina-embedding-v1-1b
4）https://platform.openai.com/docs/guides/embeddings/#embedding-models

大で nDCG が 24%，MRR が 23%改善しており，ベクトル検索における提案手法の有効性を示している．また，BM25 でも nDCG と MRR がそれぞれ 16%と 11%改善しており，キーワード検索についても提案手法の有効性が確認された．

5 分析

ここでは，4 節で得られた結果について詳細な分析を与え，性能向上の要因や提案手法の有用性について考察する．

5.1 事例分析

実際の検索事例を確認し，提案手法によってどのようなテキストの修正が行われたのかを分析する．検索結果の一例を表 2 に示す．表は，構造化を適用する前と適用した後について，高知県内の特定の市の名称を問うクエリに対して得られた検索結果の上位 3 件を示している．上段の構造化なしの例では，市の名称に関する記事は検索されているものの，正例の四万十市に関する記事は 3 位以内に検索できていない．対して，下段の構造化ありの例では，クエリと記事の内容が整理されており，四万十市を第 2 位に検索することができている．

一方で他の事例を確認すると，提案手法によって構造化されたクエリの中には，HyDE のように LLM の内部知識を用いて疑似回答が出力されている事例もあった．性能向上の要因が構造化以外に存在する疑いがあるため，次節でその要因を検証する．

5.2 性能向上の要因分析

5.1 節の分析に基づき，提案手法から疑似回答生成の要因を取り除いた場合の有効性を検証する．方法としては，HyDE を適用した場合の検索結果と，HyDE を適用した後に追加で提案手法を適用した場合の検索結果を比較し，その差分を観測する．もし構造化が性能向上に寄与する場合は，HyDE と提案手法を組み合わせた結果が HyDE 単体を用いた結果を上回るはずである．

実験結果を表 1 の右二列「HyDE」「HyDE+提案手法」に示す．HyDE 単体の結果と提案手法を組み合わせた場合の結果を比較して，有意な性能差は観測されなかった．「提案手法」で見られた性能向上は，擬似的な回答生成に起因した結果であると考察できる．

表 3 アブレーション分析．

カテゴリー	キーワード	説明	nDCG@10	MRR@10
			0.357	0.573
✓			0.123	0.221
	✓		0.400	0.589
		✓	0.483	0.714
✓	✓		0.400	0.598
✓		✓	0.460	0.690
	✓	✓	0.497	0.728
✓	✓	✓	0.484	0.717

5.3 アブレーション分析

テキスト構造化は，属性を特徴量として捉え，人間が解釈可能なものとして抽出や識別することを可能とする．本節では，このような属性の異なる組合せによって得られる検索精度の変化を分析し，テキスト検索に有用な特徴を選択する例を示す．

3 種類の属性の異なる組合せによって得られた実験結果を表 3 に示す．実験に使用したモデルは，教師なし SimCSE large である．結果より，キーワード属性と説明属性は特徴量として有効であるのに対し，カテゴリー属性は有効で無いことがわかる．最適な属性の組合せは，キーワードと説明であった．

このように属性の組合せについて性能を検証することで，提案手法は検索に有効な属性を定量的に評価することが可能である．属性の有効性を数値として分析し，それらを操作することで精度改善を図ることは，HyDE などの既存の前処理手法ではできない提案手法の利点である．

6 おわりに

本研究では，ベクトル検索の精度改善を目的として，LLM を用いてテキストを構造化する手法を提案した．質問応答ベンチマークを用いた実験の結果，前処理を適用しない場合と比較して性能の向上が確認された．ただし，この性能向上は構造化の過程で暗黙的に発生する疑似回答生成に大きく起因するものであり，純粋な文書構造の修正による改善は限定的であった．一方で，属性の組合せにより検索精度の改善が可能であるという，提案手法特有の利点も観測された．属性を設計し，その組合せを定量的に比較することで，提案手法はさらなる性能の向上を目指すことが可能である．

参考文献

[1] Bhaskar Mitra and Nick Craswell. An Introduction to Neural Information Retrieval. **Foundations and Trends® in Information Retrieval**, Vol. 13, No. 1, pp. 1–126, 2018.

[2] Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. Dense Text Retrieval based on Pretrained Language Models: A Survey. **arXiv:2211.14876**, 2022.

[3] Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. AmbigQA: Answering Ambiguous Open-domain Questions. In **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)**, pp. 5783–5797, 2020.

[4] Xiao Wang, Sean MacAvaney, Craig Macdonald, and Iadh Ounis. Generative Query Reformulation for Effective Adhoc Search. In **Proceedings of the First Workshop on Generative Information Retrieval**, 2023.

[5] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query Rewriting in Retrieval-Augmented Large Language Models. In **Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing**, pp. 5303–5315, 2023.

[6] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise Zero-Shot Dense Retrieval without Relevance Labels. In **Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 1762–1777, 2023.

[7] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In **Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)**, pp. 1–14, 2017.

[8] Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. MTEB: Massive Text Embedding Benchmark. In **Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics**, pp. 2014–2037, 2023.

[9] Hayato Tsukagoshi and Ryohei Sasano. Ruri: Japanese General Text Embeddings. **arXiv:2409.07737**, 2024.

[10] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Multilingual E5 Text Embeddings: A Technical Report. **arXiv:2402.05672**, 2024.

[11] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text Embeddings by Weakly-Supervised Contrastive Pre-training, 2024.

[12] Xuanfan Ni, Piji Li, and Huayang Li. Unified Text Structuralization with Instruction-tuned Language Models. **arXiv:2303.14956**, 2023.

[13] Gemini Team and Google. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. **arXiv preprint arXiv:2403.05530**, 2024.

[14] Hayato Tsukagoshi, Ryohei Sasano, and Koichi Takeda. Japanese SimCSE Technical Report. **arXiv preprint arXiv:2310.19349**, 2023.

[15] Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In **Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing**, pp. 6894–6910, 2021.

[16] Xinyu Zhang, Xueguang Ma, Peng Shi, and Jimmy Lin. Mr. TyDi: A Multi-lingual Benchmark for Dense Retrieval. In **Proceedings of the 1st Workshop on Multilingual Representation Learning**, pp. 127–137, 2021.

[17] Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. LLM2Vec: Large Language Models Are Secretly Powerful Text Encoders. In **First Conference on Language Modeling**, 2024.

[18] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying Conditional Random Fields to Japanese Morphological Analysis. In **Proc. of EMNLP**, pp. 230–237, 2004.

[19] Yuichi Tateno. JQaRA: Japanese Question Answering with Retrieval Augmentation - 検索拡張 (RAG) 評価のための日本語 Q&A データセット.

[20] 鈴木正敏, 鈴木潤, 松田耕史, 京介, 井之上直也. JAQKET: クイズを題材にした日本語 QA データセットの構築. 2020.

[21] Elias Bassani. ranx: A Blazing-Fast Python Library for Ranking Evaluation and Comparison. In **ECIR (2)**, Vol. 13186 of **Lecture Notes in Computer Science**, pp. 259–264. Springer, 2022.

A　プロンプト

次の JSON スキーマを使用して、テキストを構造化してください。
テキストから得られない情報は、あなたの知識をもとに埋めてください。

テキスト：（ここに対象のテキストを入力）

```
Text_structure = {
   ”カテゴリー”: str,
   ”キーワード”: str,
   ”説明”: str,
}

Return: Text_structure
```

図 2　テキスト構造化に用いたプロンプト

```
質問に答える文章を書いてください。
質問：（ここに対象のテキストを入力）
文章：
```

図 3　HyDE に用いたプロンプト