故障解析における事前学習済み Sentence-DeBERTa による拡張ナレッジグラフとクエリ分解を用いた GraphRAG

小島 湧太 [1]　坂地 泰紀 [2]　鈴木 雅弘 [1]　中村 格士 [3]
坂田 大晃 [3]　関 和也 [3]　勅使河原 優 [3]　山下 雅己 [3]　青山 和浩 [1]
[1] 東京大学大学院工学系研究科　[2] 北海道大学大学院情報科学研究院
[3] いすゞ自動車株式会社
y_ojima@m.sys.t.u-tokyo.ac.jp

概要

熟練から若手エンジニアへの故障解析の知識継承は自動車業界喫緊の課題である．故障に関する文書は非構造的でデータ量が多いため，そのままでは活用が困難である．このような文書をデータの構造化に優れるナレッジグラフ (KG) に変換することは有効であるが，依然として理解が難しい．GraphRAG は，KG と大規模言語モデル (LLM) による RAG (Retrieval-Augmented Generation) であり，KG 自体の理解を深めることも期待される．

既存の KG を用いた GraphRAG に必要な処理を調査するため，本稿では LLM と継続事前学習済み Sentence-DeBERTa で拡張した既存の KG と，クエリの分解に焦点を当てた EdgeGraphRAG を提案する．独自のデータセット構築と自動評価により，現時点での有効性の検討と既存の KG を用いた際の課題を論じる．

1　導入

昨今自動車業界において，経験のあるエンジニアから若いエンジニアへの故障解析の知識継承の必要性が高まっている．少子高齢化及び CASE に代表される新技術の台頭のために，若手技術者の対応する故障解析が広範囲かつ高い専門性が求められるようになっている事が理由に挙げられる．一般に自動車は複数の部品で構成されており，一つの部品の故障が他の部品の故障にも連鎖する可能性があるため，故障解析は複雑である．部品の関連性の理解等は故障原因を特定するうえで重要となるが，これらは主に経験によって得られる知識となっているため，故障解析の継承に課題感を持つ人は多いとされる [1]．

トラックメーカーは故障の**不具合対応文書**を故障が起きた際に発行しており，文書にある知識の共有による知識継承を目指している．文書には，メーカーの販売会社に来る故障車の故障状態・修理方法などが記載されているが，非構造的で人間にとってデータ量も多いため，理解が難しい．

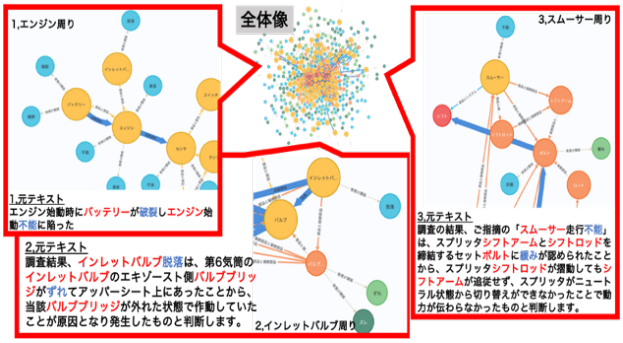

図 1　故障 BOM の概要図 [2]

原ら (2022) は，これら不具合対応文書に対して構文解析を施し，**故障 BOM** と呼ばれるナレッジグラフ (KG) を構築した [2]．KG は意味的な関係性や構造的な情報を格納することができるため，多くの領域の問題解決策として用いられており [3]，自動車業界も例外ではない．故障 BOM には，主に部品やシステムの関係性や状態遷移が記述されており，ノードについては人手でラベル付けがされている．

しかしながら，故障 BOM の理解には図 1 のように前提となる知識が要求されることに加え，エッジが因果関係，弱因果関係，状態関係，階層関係といった抽象的な概念でしか記述されておらず，本来存在していた情報が欠落しているという状況となっているため，**若手エンジニアにとっては未だに理解しがたい状態**となっている．そのため，故障 BOM のアップデート及び，それを用いたアプリケーションが必要であると考える．

ChatGPT[4] に代表される大規模言語モデル (LLM)

は非常に高い性能を誇っており, 対話能力はもちろんのこと, 複雑なタスクも遂行可能 [5] なため, 企業における知識マネジメントにも活用が期待される.

一方で, 学習データに含まれていない知識・情報について正しい情報を生成することは困難で, 現時点で LLM の学習は大量のコストが要求されるため, 多くの企業にとっては未だに現実的な手段とは言えない. そのため, 質問回答システムのアプローチの一つである **RAG** (Retrieval-Augmented Generation) に注目が集まっている. RAG はユーザーからの質問 (ユーザークエリ) を元に, 外部のデータソースから関連する関連する情報を検索して抽出し, 得られた情報を元に LLM が回答を生成する [6]. データソースには, 主に文書のベクトルデータベースが用いられるが, 正確性の高さ及びハルシネーションの抑制という観点から, 直近では KG を用いた **GraphRAG**[6] の注目度が急速に高まっている.

しかしながら多くの GraphRAG の研究は, 既に構築された大規模なオープンドメインの KG を用いる, もしくは KG の構築から LLM が実行する (Microsoft による GraphRAG[7]) 場合が多い一方で, 企業独自, 特に製造業のデータ及び KG を用いた研究は少なく, 実際に企業内で展開していくうえでどのような処理や工夫が必要かは不透明なのが現状である.

本研究では, 既存の KG を LLM と継続事前学習させた Sentence-DeBERTa で拡張をし, その KG とクエリ分解を用いた EdgeGraphRAG を提案する.

2 関連研究

2.1 GraphRAG

GraphRAG のアプローチは現時点では複数存在するが, 基本的には, Graph-Based Indexing: グラフデータベース (GDB) の構築・インデックスの構築, Graph-Guided Retrieval: グラフデータからサブグラフの抽出, Graph-Enhanced Generation: 得られたサブグラフから回答生成の 3 ステップに分けられる [8].

2.2 IR-based Graph RAG

Ojima ら (2024) [9] は, ユーザークエリからキーワードを抽出し, キーワードを含むサブグラフ (より詳細にはトリプル: 1-hop のサブグラフ) を KG から抽出し, サブグラフを元に LLM が回答を生成するという Information Retrieval (IR) method[10] に基づいた GraphRAG を提案し, 一定の有効性を示した.

一方で, ユーザークエリに含まれてない単語起点のサブグラフ抽出が不可能であり, 加えて現状は LLM によるサブグラフ候補から関連のあるサブグラフの選別を行っているものの, 説明可能性が低く, 選別の精度・速度も向上の余地がある. 前者については, ノードにおける**同義語** (例: レリーズベアリングはあるが, リリースベアリングはない), あるいは**複合語** (例: クラッチ, マスターシリンダーはあるがクラッチマスターシリンダーはない) といった問題がボトルネックである. これは, 想定されるキーワードの数を増やせれば解決が可能であると言える一方で, 抽出されるサブブラフの数が膨大となるため, **サブグラフの選別により工夫が必要**となる.

3 提案手法: EdgeGraphRAG

[9] の手法を踏襲しつつ, 故障 BOM をアップデートした**拡張故障 BOM** の構築, 及び上述の問題点を考慮した **EdgeGraphRAG** を提案する.

仮説として, ユーザークエリから想定される故障の事象と意味的に近いサブグラフを抽出すれば, より適切なサブグラフが得られると考えた.

3.1 拡張ナレッジグラフ

意味的に近いサブグラフの抽出のためには, トリプルの情報による分散表現を各々のエッジに埋め込むことが適切と考えた. 分散表現埋め込みをした KG のうち, 有名なものとして TransE[11] が挙げられる. ノードとエッジそれぞれで個別の分散表現が埋め込まれており, 同じ名前であれば同じ分散表現となっている. しかし, KG 特有の問題としてノードの語義曖昧性が挙げられ, それは部品が複雑に関連している故障 BOM においても同様である. そこで, 各々のエッジ単位で分散表現を埋め込むことができれば, 同じノードやエッジであっても組み合わせによって異なる分散表現を得ることが可能であると考えた.

埋め込みに際しては, BERT[12] に代表される双方向の Encoder を採用したモデルが適切であると考えた. 文脈に即したエンコーディングが可能であり, このような語義曖昧性の問題を回避することができると言える. 本稿では双方向 Encoder を採用したモデルの中で高い性能を誇る DeBERTa[13] を用いる.

継続事前学習済み Sentence-DeBERTa 分散表現獲得のため, メーカー独自のテキストデータで継続事前学習を行った DeBERTa を Sentence-BERT 化 [14] した DeBERTa (**Sentence-DeBERTa**) を構築した.

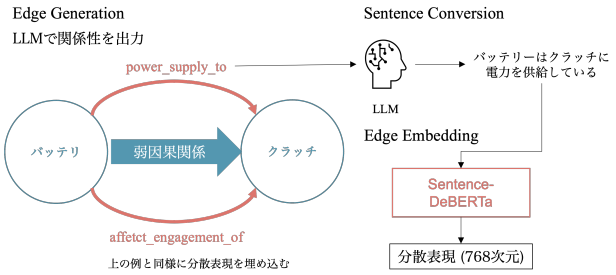

図 2　拡張故障 BOM の構築

専門性の高いテキストデータを最初から学習させてしまうと, 共変量シフト [15] の状況に陥る可能性がある. 対策として, **事前学習を段階的に施し徐々にドメイン適応させる**ことで性能向上を図るアプローチ [16] が適切と考えた. 具体的には, Sakaji ら (2022) における金融領域での継続事前学習 [17] を参考に, 公開されている日本語データで事前学習済みの DeBERTa モデル, izumi-lab/deberta-v2-base-japanese[1] に対して追加で 2 段階の計 3 段階の継続事前学習を行った. 学習リソースの 2 段目には自動車メーカー各社の技報や機械工学系の教科書など自動車産業に関する一般的なテキストを, 3 段目にメーカー独自のマニュアルやハンドブックを用いた.

構築ステップ

Edge Generation　現時点で存在しているエッジに対して, 考えられる具体的なラベルを LLM によって出力させる. 故障 BOM のエッジのラベルは抽象的であり, LLM に与える情報が乏しいという問題がある. そのため, まずはエッジのラベルをより具体的に記述することが必要であると考えた. 具体的なエッジの記述は, 人手では数が多く対応が難しいので, LLM による生成が適切であると考えた.

Sentence Conversion　得られたトリプルを LLM によって日本語の文章に変換する. これは, 日本語で学習させた Sentence-DeBERTa からより適切な分散表現を得るためである.

Edge Embedding　出力した文章を Sentence-DeBERTa でエンコードし, 得られた分散表現をエッジに埋め込む.

3.2　EdgeGraphRAG

パイプラインの全容を図 3 に示す.

Anchor Identification　サブグラフ抽出の起点 (anchor) となるキーワード郡の抽出を行う. この節

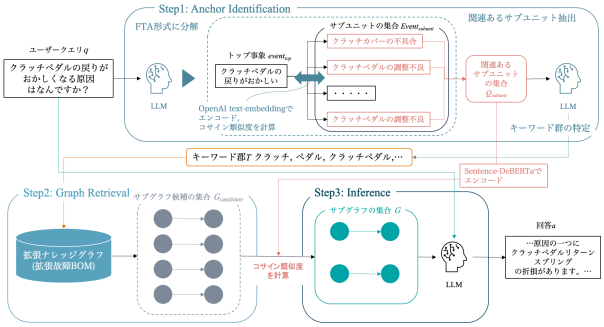

図 3　EdgeGraphRAG のパイプライン

2.2 で記したように, **キーワードの候補を増やす工夫が必要**となる. 具体的には, ユーザークエリ q に対して, **FTA** (Fault Tree Analysis)[18] のアプローチでトップの故障事象 $event_{top}$ とその原因候補 (**サブユニット**) $event_{subunit}$ の集合 $Event_{subunit}$ に分解する. 一般的に故障が起きた際の原因推定の際用いられるアプローチであることに加えて, ユーザークエリをサブクエリに分解することで精度の向上が報告 [19] されていることから, 本研究でも採用した.

$Event_{subunit}$ のうち, 関連あるサブユニットを特定するために, $event_{top}$ とそれぞれの $event_{subunit}$ を OpenAI text-embedding-3-small でエンコードし, $event_{top}$ とそれぞれの $event_{subunit}$ の間でコサイン類似度を求め, 閾値以上[2] のサブユニット $q_{subunit}$ の集合を $Q_{subunit}$ とする.

$Q_{subunit}$ に対して, [9] での Retrieving, すなわち故障に関連する単語 t の抽出を行う. 抽出されたそれぞれの単語 t において, 同義語・複合語の対策として, それぞれの単語から連想される単語 t' を LLM に生成させる. 得られた単語 t, t' の集合をキーワード郡 T とする.

Sub-graph Retrieval　T を元に, 拡張故障 BOM から関連するサブグラフを抽出し, エッジ埋め込みを用いたサブグラフのフィルタリングを行う. サブグラフ候補の集合 $G_{candidate}$ は, [9] での Extracting と同様に, 始点あるいは終点が T に含まれるトリプルを抽出する. $G_{candidate}$ から, 回答に用いるサブグラフの集合 G の選択の際には, それぞれの $q_{subunit}$ を Sentence-DeBERTa でエンコードしたベクトルとのコサイン類似度を計算し, 閾値以上[2] のサブグラフを選択する.

Inference　G を元に, LLM が回答 a を生成する. この際, サブグラフは [9] と同様に, グラフを直接記

1)　https://huggingface.co/izumi-lab/deberta-v2-base-japanese

2)　初期値 0.6 とし, 一つも取れなかった場合は 0.1 ずつ下げて抽出を繰り返す

表 1　実験結果

Method	R-1	R-2	R-L	B-Score	S-DeBERTa	text-embedding
IR-based Graph RAG + 故障 BOM	0.4876	0.2302	0.3505	0.7672	0.7766	0.7855
IR-based Graph RAG +拡張故障 BOM	0.4863	0.2303	0.3514	0.7682	0.7662	0.7829
naiveRAG (OpenAI)	0.5218	0.2748	0.3788	0.7825	0.8170	**0.8162**
naiveRAG (Sentence-DeBERTa)	0.5164	0.2652	0.3683	0.7838	**0.8195**	0.8147
EdgeGraphRAG	0.4962	0.2322	0.3615	0.7681	0.7768	0.7790
EdgeGraphRAG+naiveRAG (OpenAI)	**0.5266**	**0.2818**	**0.3920**	**0.7878**	0.8064	0.8078
EdgeGraphRAG+naiveRAG (Sentence-DeBERTa)	0.5161	0.2724	0.3823	0.7851	0.8025	0.8013

述する形式で LLM のプロンプトに渡している.

4　実験

4.1　比較対象

文書データベースを用いた **naiveRAG** と, **IR-based Graph RAG**[9] を比較対象とした. naiveRAG には, Retriever として Sentence-DeBERTa: naiveRAG (Sentence-DeBERTa) 及び Azure OpenAI text-embedding-3-small: naiveRAG (OpenAI) を用いた. 実装に際しては **HyDE** (Hypothetical Document Embeddings) と呼ばれる, 一度ユーザークエリに対して何も情報を与えない状態で LLM から仮の回答を生成 [20] し, 仮の回答とコサイン類似度の高い文書の集合 S を抽出, 文書を元に再度回答を生成するプロセスを実行した.

また, **EdgeGraphRAG と naiveRAG の組み合わせ**も比較対象とした. これは, EdgeGraphRAG の Subgraph Retrieval で得られたサブグラフの集合 G と naiveRAG の Retriever[3] で得られた文書 S を LLM のプロンプトに渡して, 得られた生成文を評価する.

LLM は, 全て Azure OpenAI GPT-4o[4] を用いた.

4.2　評価方法

不具合対応文書を元にしてデータセットを LLM で構築し[5], 生成文と参照文の類似度で性能を評価した. RAG のデータソースに基づいてデータセットを作成, 評価するアプローチは, [21] を参考にした. いすゞ自動車株式会社から提供された文書のうち, "**クラッチ**" が含まれる 43 件を対象にした.

評価指標は, ROUGE1,2,L[22] の F1 値 (R-1,2,L) と BERTScore[23] の F1 値[6] (B-Score), 及び Sentence-DeBERTa (S-DeBERTa), Azure OpenAI text-embedding-3-small (text-embedding) でエンコードした生成文と参照文のコサイン類似度を用いた. 実行回数はそれぞれ 5 回とし, 平均値を評価する.

5　結果

表 1 に実験結果を示す. 提案手法は, IR-based Graph RAG よりはわずかによい結果を残していると言えるものの, naiveRAG には及ばず, naiveRAG (OpenAI) を含む手法が全体的に高いスコアを記録した. これは過去のデータを元にした質問 (Local Question) のデータセットを用いており, その場合, GraphRAG よりも直接情報を参照できる naiveRAG の方が適しているからと考えられる. 2 つの naiveRAG 間では有意差は見られず, Sentence-DeBERTa は Retriever としては OpenAI とほぼ同等の性能にあるといえる. 拡張故障 BOM の効果について, 単にラベルの具体性を増やすだけでは, 有意な性能向上は見られなかった. このことからトリプルでは, ノードにある単語のみが情報として取り入れられており, 関係性は十分に考慮されていない可能性がある.

トリプルではなく Multi-hop でのサブグラフ抽出は候補としてあったが, 提案手法のアプローチではサブグラフ抽出に多くの時間を必要とするため, 本研究では断念した. 現状の故障 BOM においてクラッチなど主要な部品に多くの部品が関連してしまうため, サブグラフの抽出という観点からは相性が悪い可能性がある.

6　結言

現時点で, 既存の KG を用いる GraphRAG では, 一定の工夫によって RAG としての有効性は示しつつも限界があり, エッジの削減やキャッシュなど KG を GraphRAG のために抜本的に構築し直す工夫が必要であると言える. 対象データの増加や, データソースに含まれないデータを用いたデータセットでの評価, naiveRAG との組み合わせによる効果検証も必要である. これらは今後の課題としたい.

3)　閾値を 0.6, 最大 100 件まで
4)　2024-08-01-preview, tempreture=0
5)　Azure OpenAI GPT-4,1106-preview, tempreture=0
6)　bert-base-multilingual-cased

謝辞

本研究はいすゞ自動車株式会社との共同研究の成果の一部である.

参考文献

[1] 斎藤彰. AI の進化は故障解析に何をもたらすのか ～その期待とリスク～. 日本信頼性学会誌 信頼性, Vol. 40, No. 2, pp. 64–71, 2018.

[2] 原祥太, 小澤秀明, 山下雅己, 山田周歩, 坂地泰紀, 青山和浩. 製品情報のテキストマイニングによる設計開発・保守のナレッジマネジメント. 設計工学・システム部門講演会講演論文集, Vol. 2022.32, p. 1201, 2022.

[3] Bilal Abu-Salih. Domain-specific knowledge graphs: A survey. **Journal of Network and Computer Applications**, Vol. 185, p. 103076, 2021.

[4] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In **Proceedings of the 36th International Conference on Neural Information Processing Systems**, NIPS '22, Red Hook, NY, USA, 2024. Curran Associates Inc.

[5] Linyao Yang, Hongyang Chen, Zhao Li, Xiao Ding, and Xindong Wu. Give us the Facts: Enhancing Large Language Models With Knowledge Graphs for Fact-Aware Language Modeling. **IEEE Transactions on Knowledge and Data Engineering**, Vol. PP, pp. 1–20, 07 2024.

[6] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-Augmented Generation for Large Language Models: A Survey, 2024. arXiv preprint arXiv:2312.10997.

[7] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From Local to Global: A Graph RAG Approach to Query-Focused Summarization, 2024. arXiv preprint arXiv:2404.16130.

[8] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph Retrieval-Augmented Generation: A Survey, 2024. arXiv preprint arXiv:2408.08921.

[9] Yuta Ojima, Hiroki Sakaji, Tadashi Nakamura, Hiroaki Sakata, Kazuya Seki, Yuu Teshigawara, Masami Yamashita, and Kazuhiro Aoyama. Knowledge Management for Automobile Failure Analysis Using Graph RAG. In **2024 IEEE International Conference on Big Data (IEEE BigData 2024)**, 2024.

[10] Yuxin Luo, Bailong Yang, Donghui Xu, and Luogeng Tian. A Survey: Complex Knowledge Base Question Answering. In **2022 IEEE 2nd International Conference on Information Communication and Software Engineering (ICICSE)**, pp. 46–52, 2022.

[11] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, **Advances in Neural Information Processing Systems**, Vol. 26. Curran Associates, Inc., 2013.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In **North American Chapter of the Association for Computational Linguistics**, 2019.

[13] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. In **International Conference on Learning Representations**, 2021.

[14] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, 2019. arXiv preprint arXiv:1908.10084.

[15] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. **Journal of Statistical Planning and Inference**, Vol. 90, No. 2, pp. 227–244, 2000.

[16] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**, pp. 8342–8360, Online, July 2020. Association for Computational Linguistics.

[17] Hiroki Sakaji, Masahiro Suzuki, Kiyoshi Izumi, and Hiroyuki Mitsugi. Gradual Further Pre-training Architecture for Economics/Finance Domain Adaptation of Language Model. In **2022 IEEE International Conference on Big Data (Big Data)**, pp. 2352–2355, 2022.

[18] 鈴木浩一. FMEA, FTA (その 2). 日本航空宇宙学会誌, Vol. 48, No. 558, pp. 438–443, 2000.

[19] Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. KG-GPT: A general framework for reasoning on knowledge graphs using large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, **Findings of the Association for Computational Linguistics: EMNLP 2023**, pp. 9410–9421, Singapore, December 2023. Association for Computational Linguistics.

[20] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise Zero-Shot Dense Retrieval without Relevance Labels, 2022. arXiv preprint arXiv:2212.10496.

[21] Angels Balaguer, Vinamra Benara, Renato Luiz de Freitas Cunha, Roberto de M. EstevÃ£o Filho, Todd Hendry, Daniel Holstein, Jennifer Marsman, Nick Mecklenburg, Sara Malvar, Leonardo O. Nunes, Rafael Padilha, Morris Sharp, Bruno Silva, Swati Sharma, Vijay Aski, and Ranveer Chandra. RAG vs Fine-tuning: Pipelines, Tradeoffs, and a Case Study on Agriculture, 2024. arXiv preprint arXiv:2401.08406.

[22] Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In **Text Summarization Branches Out**, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[23] Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating Text Generation with BERT. In **International Conference on Learning Representations**, 2020.

A　利用したプロンプト

A.1　Anchor Identification

ユーザークエリを FTA で分解

あなたは、自動車の不具合・故障情報に関して大きな専門性を持つエンジニアです。ユーザーから渡される質問は、自動車の故障の原因を問うものになっています。渡された質問から対象となる故障事象を抜き出したうえで、その故障事象が一番上にくる FTA を考えるのがあなたのタスクです。その故障事象を踏まえたうえで、サブユニットとして考えられる故障事象を限界まで挙げてください。

指示
- 回答は json 形式で、一番上の故障事象を top というキーに、サブユニットの故障事象は subunits というキーに格納してください。

サブユニットからキーワード特定

あなたは、自動車の不具合・故障情報に関して大きな専門性を持つエンジニアです。故障の原因を特定するために、キーワードとなる単語を特定し、そのキーワードに関連する情報をナレッジグラフから抽出する必要があります。あなたのタスクは、次に渡される文章の中で、自動車の不具合・故障に関連する部品の単語を抜き出すことです。

次の注意点とナレッジグラフの特徴を考慮しつつ、タスクを遂行してください。

注意点
- 不具合という単語やそれに似た単語は抜き出さないできるだけ細かい単語にする
例: クラッチペダル→クラッチペダル, クラッチ, クラッチ
- 抜き出した単語は json 形式で、words というキーに格納する

ナレッジグラフの特徴
{ 故障 BOM の特徴を記述 }

キーワードから連想される単語を生成

あなたは、自動車の不具合・故障情報に関して大きな専門性を持つエンジニアです。故障の原因を特定するために、キーワードとなる単語を特定し、そのキーワードに関連する情報をナレッジグラフから抽出する必要があります。
ユーザーから渡される単語は、自動車の不具合・故障に関連する単語です。あなたのタスクは、与えられた単語を、同じ意味だけど違う単語に置き換えることです。変換に際して、複数の単語の出力を推奨します。
例: 排気ブレーキ→エキゾーストブレーキ

注意点
- 変換した単語は json 形式で、words というキーに配列で全て格納する。

A.2　Inference

抽出したサブグラフから回答生成

あなたは、自動車の不具合・故障情報に関して大きな専門性を持つエンジニアです。ユーザーから渡される質問は、故障の原因を問うものになっています。以下の関係性とそのラベルを参照して、故障の原因を特定して、その原因における部品の構成・故障の伝播を回答してください。また、以下の注意点を考慮してください。

注意点
- 回答に謝罪を含めない
- 回答に質問の内容を繰り返さない
- 回答が質問に即した内容になっている
- 回答には、具体的な数字・日にち・場所のような、自動車の知識とは関係ない情報を含まない
- あなたがわからない知識を回答に含めない
- 箇条書きを用いない
- 原因となる部品を特定する

関係性
{抽出したサブグラフをここに記述}