ナレッジグラフ推論チャレンジ 2024 応募シート

１．応募者に関する情報

- 氏名またはチーム名：棚橋 良将
- 所属：大阪公立大学 創発ソフトウェア研究室
- メールアドレス（代表）：sfb01084@st.osakafu-u.ac.jp
- 応募者に学生が含まれる：　はい
- 応募者の代表が学生である：　はい

２．応募部門：一般部門

３．構築したナレッジグラフについて

- **構築対象としたナレッジグラフ**
  テーマ：論文の引用関係を表現するナレッジグラフ
  　LLM を用いて論文の PDF データから参考文献などの情報を取得し，グラフデータベースである Neo4j で論文の引用関係を表現するナレッジグラフを構築した.

- **構築したナレッジグラフの基本情報**
  - ○ データサイズ
    - ノード数：3341
    - リレーション数：3250
    - ファイルサイズ：45.5MB（data.backup）
  - ○ データ形式
    - Sample1.png（ナレッジグラフのサンプル画像）
    - Sample2.png（ナレッジグラフのサンプル画像）
    - data.backup（ナレッジグラフの Neo4j データ）
    - data.json（ナレッジグラフの JSON データ）

- **構築したナレッジグラフのデータの入手先**
  公開先 URL：https://github.com/tanahashi415/KGRC2024/tree/main/Data

4．ナレッジグラフ構築に用いた「言語モデル」および「構築手法」について

- **ナレッジグラフ構築に用いた「言語モデル」**
  - モデル：gpt-4o-mini（gpt-4o-mini-2024-07-18）
  - URL：https://platform.openai.com/docs/models#gpt-4o-mini

- **ナレッジグラフ構築に用いた「データ」**
  OpenAlex API でランダムに取得した論文 100 件を利用してナレッジグラフを構築した．
  - データ形式：PDF
  - データ量：100 件

- **ナレッジグラフの構築手法の説明**
  ナレッジグラフの構築手法の説明の前に要素技術について説明する．

  - Neo4j

  Neo4j とは，Java で開発されたオープンソースのグラフデータベースである．データはエンティティであるノード（頂点）とノード間の関係を表すリレーション（辺）で表現される．ユーザは Neo4j のグラフクエリ言語である Cypher を使うことで，データベースに対してノードおよびリレーションの作成や検索といった操作をすることができる．各ノードやリレーションには，ラベルやプロパティという情報を与えることができる．

  - PyMuPDF4LLM

  PyMuPDF4LLM は，LLM や RAG 環境で必要な形式で PDF コンテンツを簡単に抽出できるようにすることを目的とする Python ライブラリである．提案するシステムでは，PDF を Markdown 形式に変換するために使用した．

  - Unstructured

Unstructured は PDF のような非構造化ドキュメントを処理するための Python ライブラリである．ドキュメントの情報は「Title」や「Header」などにラベル分けされ，このラベルを用いると任意の情報をまとめて取得することができる．提案するシステムでは，参考文献のリストを取得するために使用した．

次に提案するシステムについて説明する．なお，理解しやすくするために，一部の処理の説明を省略している．

1. 論文データの変換

論文の PDF データを PyMuPDF4LLM で Markdown 形式に変換する．

2. 論文ノードの作成

2.1 Markdown 形式に変換した論文テキストから LLM を用いて論文の情報（ノード情報）を取得する．

図 1 に LLM への問い合わせに用いたプロンプトを示す．「# Instruction 1」では，論文のテキストから [タイトル，著者，分野，作成日，要約] の 5 つの要素を取得するように指示している．「# Instruction 2」では，回答は辞書型で返すように指示している．「# Instruction 3」では，作成日は Y-M-D 形式で回答するように指示している．作成年度しか取得できなかった場合は，その年の 1 月 1 日をデフォルトの出力としている．「# Instruction 4」では，分野を 26 種類のいずれかに分類するように指示している．この 26 種類の分野は参考文献を検索する際に使用した OpenAlex の分野分類に準拠している．「## Notes」では，情報が取得できなかった場合は，その項目を unknown とするように指示している．

```
# Instruction 1
Get the following information from the paper text provided by the user:
- title
- authors
- field
- created
```

- abstract

Instruction 2
Please return the answer as a dictionary.
Example
{"title" : "ChatGPT", "authors" : "OpenAI", ...}

Instruction 3
Output `created` in year-month-day format. If the month and day are unknown, use "year-01-01".

Instruction 4
Classify the `field` into one of the following:
- Medicine
- Social Sciences
- Engineering
- Arts and Humanities
- Computer Science
- Biochemistry, Genetics and Molecular Biology
- Agricultural and Biological Sciences
- Environmental Science
- Physics and Astronomy
- Materials Science
- Business, Management and Accounting
- Economics, Econometrics and Finance
- Psychology
- Health Professions
- Chemistry
- Earth and Planetary Sciences
- Neuroscience
- Mathematics
- Immunology and Microbiology
- Decision Sciences
- Energy
- Nursing
- Pharmacology, Toxicology and Pharmaceutics
- Dentistry
- Chemical Engineering
- Veterinary

Notes
If no information is found, please mark it as "unknown"

図 1. 論文から情報を取得するプロンプト

2.2　取得した情報をプロパティとして Neo4j に論文ノードを作成する．

3.　参考文献の論文ノード作成

3.1　Markdown 形式に変換した論文テキストから正規表現あるいは Unstructured を使って参考文献リストを取得する．

正規表現に使用したパターンは次の 2 種類である．パターン 1 が英語用でパターン 2 が日本語用である．

```
pattern1 = '(References|REFERENCES|R E F E R E N C E S|Bibliography|NOTES)(.*)-----'
pattern2 = '(#+ |¥*¥*)(参考文献|参 考 文 献|文献)(.*)-----'
```

上記の正規表現で参考文献リストの取得に失敗した場合，Unstructured を使って参考文献リストを取得する．この方法では箇条書き項目を意味する「ListItem」ラベルの要素を取得することで，参考文献リストを取得する．

3.2　LLM を用いて参考文献リストから論文のタイトルだけを取得する．

図 2 に例を示す．また，図 3 に LLM への問い合わせに用いたプロンプトを示す．

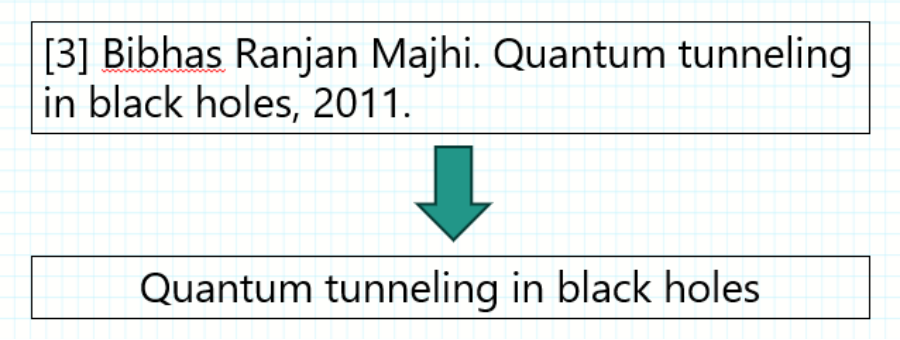

図 2. 参考文献リストからのタイトル抽出例

```
Extract only the titles of the papers from the given text. Please return the answer in bullet points.
### Example
- Learning Crosslingual Word Embeddings without Bilingual Corpora
- Semi-Supervised Translation with MMD Networks
...
```

図 3. 参考文献リストから論文のタイトルだけを取得するプロンプト

3.3　参考文献を OpenAlex API を用いて検索し，参考文献の論文ノードを作成する．

参考文献の論文のタイトルを基に OpenAlex でベクトル検索をする．検索目標の論文のタイトルと検索結果の論文のタイトルのコサイン類似度が 0.9 以上である場合，検索に成功したと見なす．検索に成功した場合，[タイトル，著者，分野，サブ分野，作成日，要約，DOI，被引用数]の 8 つの情報を取得する．この情報をプロパティとして Neo4j に論文ノードを作成する．

3.4　2.2 で作成した論文ノードと 3.3 で作成した論文ノードとのリレーションを作成する．

リレーションの方向は (引用論文)-[Refer]->(被引用論文) である．

4.　著者ノードの作成

4.1　著者を OpenAlex API を用いて検索し，著者ノードを作成する．

2.1 で取得した著者情報を基に OpenAlex でベクトル検索をする．検索目標の著者名と検索結果の著者名のコサイン類似度が 0.9 以上である場合，検索に成功したと見なす．検索に成功した場合，[名前，ORCID，論文数，論文の総引用数，専門分野，h 指数]の 6 つの情報を取得する．この情報をプロパティとして Neo4j に著者ノードを追加する．

4.2　2.2 で作成した論文ノードと 4.1 で作成した著者ノードとのリレーションを作成する．

リレーションの方向は (著者)-[Write]->(論文) である．

- **パフォーマンス情報**

使用した PC のスペックは以下の通りである．

- RAM：32.0 GiB

- CPU：11th Gen Intel® Core™ i7-11700 @ 2.50GHz×16

表 1 に提案するシステムのパフォーマンスを示す．参考文献を検索する際の検索数 search_n= 3 で実行した．

表 1.システムの実行時間

論文	参考文献数	実行時間（秒）
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding	56	215.88
Deep Residual Learning for Image Recognition	50	211.30
Long Short-Term Memory	41	176.49
Attention Is All You Need	40	165.61
Adam: A Method for Stochastic Optimization	23	99.36

- **参考情報**

5．構築したナレッジグラフの評価

- **評価方法**

OpenAlex API を使って条件に合致する論文を無作為に 100 件取得し，その論文を用いてシステムの性能を検証した．

論文の取得条件は以下の通りである．
・PDF のダウンロードが可能であること
・言語が英語または日本語であること
・分野が明確である（OpenAlex にて分野が定められている）こと
・要約があること
・参考文献があること

システムの性能は，論文 PDF から情報を取得する正確さを表す抽出精度と

OpenAlex での検索成功率を表す検索精度に分けて検証した．

抽出精度は，事前に OpenAlex API から取得したデータを正解データとして，PDF から取得したデータと比較することによって精度を検証した．各論文に対して，タイトル，分野，作成日，要約は正誤判定を行い，著者と参考文献は全体の内何割の情報が取得できたかを調べた．最後に各論文の精度の平均値を取ることでシステム全体の抽出精度とした．

検索精度も同様にして，各論文に対して，抽出できた参考文献の内何割の論文が検索に成功したかを調べた．最後に各論文の精度の平均値を取ることでシステム全体の検索精度とした．

- **評価結果**

表 2 にシステムの性能検証をした結果を示す．詳細な検証データは次のリンクに示す（https://github.com/tanahashi415/KGRC2024/blob/main/Data/verification.pdf）．左の 6 つの項目は抽出精度を示しており，右の 2 つの項目は検索精度を示している．

先行研究が無いため主観的な評価になってしまうが，抽出精度については概ね高水準であると言える．分野の分類精度が少し低くなってしまったのは，分野の類似性が原因だと考えられる．例えば，Medicine（医学）と Biochemistry, Genetics and Molecular Biology（生化学、遺伝学、分子生物学）と Immunology and Microbiology（免疫学と微生物学）と Veterinary（獣医学）は互いに似たような分野領域であり，私たち人間から見ても論文がどの分野に属するか判断することが難しい場合がある．そのために分野の分類精度が低くなってしまったことが考えられる．作成日の精度が少し低くなってしまったのは，論文によっては提出日（submitted）と受理日（accepted）など複数の日付情報が記載されている場合があり，それによって誤った回答をしてしまったことが原因だと考えられる．

検索精度については，あまり良くない値に見えるかもしれないが妥当な値だと考えられる．なぜなら，検索に失敗した論文の内 OpenAlex に登録されているものの検索できなかったという事例は 1 割以下であり，多くの場合は OpenAlex に登録されていない論文であるため検索できなかったという事例で

あったからである．特に 2000 年代以前の古い論文は OpenAlex に登録されていない場合が多く，精度が落ちる原因になった．

表 2 ：システムの性能検証結果

	抽出						検索	
項目	title	field	created	abstract	authors	references	authors	references
精度	0.990	0.790	0.840	0.950	0.995	0.916	0.791	0.729

- **システムで工夫した点**

- ノードの大きさ

論文ノードは被引用数に応じて大きさが 11 段階に変化するようにした．また，著者ノードも同様に h 指数と論文の平均引用数から算出される値 α に応じて大きさ 11 段階にが変化するようにした．これによって，影響力のある論文・著者が一目で分かるようになっている．

なお，$\alpha = \text{h 指数} + \sqrt{\frac{\text{論文の総引用数}}{\text{論文数}}}$ である．h 指数は「発表した論文のうち、被引用数が h 回以上ある論文が h 本以上ある場合、これを満たす数値 h がその研究者の h 指数となる」と定義されており，当該研究者の論文の量（論文数）と論文の質（被引用数）とを同時に 1 つの数値で表すことができるという利点を持つ。しかし，h 指数の算出方法では被引用数の絶対値の情報が抜け落ちてしまい，優秀であっても論文数がまだ少ない若手研究者の数値が低くなってしまうという問題がある．そこで，h 指数の値をなるべく維持しつつ，その問題を改善する方法として上記のような値を定義した．

- 論文ノードのプロパティ

論文ノードのプロパティである created（作成日）は Date 型として保存している．これによって，日付による条件検索が可能である．また，参考文献の論文ノードには要約や DOI が保存されているので，気になった論文にはすぐにアクセスできるようになっている．アクセス先で論文の PDF を入手すれば，再びシステムを利用することができるようになり，データベースを拡張していくことが可能である．

- 分野の分類精度

初期のシステムでは分野の分類精度が 50%前後と非常に低くなってしまった．そこで十分な参考文献がある場合，参考文献の分野の多数決に基づき論文の分野を補正する機能を追加した．これによって精度が 30%近く向上した．

- **他のサービスとの違い**

提案するシステムのように論文の引用関係を表すグラフを構築するサービスとして，Connected Papers（https://www.connectedpapers.com/）や，ScholarPlanets（https://scholarplanets.skillupai.com/） などが挙げられる．提案するシステムとこれらの既存サービスとの違いは，対応する論文の幅広さにあると考える．既存のサービスではデータベース上に登録してある論文しか対応していないため，最新の論文や世間一般に公開されていない論文のグラフ構築は不可能である．しかし，提案するシステムは PDF データさえあればグラフを構築することが可能であるため，先ほど述べたような論文にも対応することができる．そのため，提案するシステムの有効的な使用方法としては，最新の論文の内容や引用関係を手軽かつ迅速に把握するために使用することや，研究室内の卒業論文などをデータベースに登録して，研究室の特色を把握したり研究室で注目されている論文を確認したりすることが考えられる．

- **今後の発展**

本システムは，興味ある分野の先行研究を俯瞰し，研究の方向性を見極めるための強力なツールとなる．被引用数に基づく関連論文の重要度の可視化が可能で，新たな知見の発見に役立つ．また，論文執筆時には参考文献の整理に利用でき，研究者の支援ツールとしても有用である．

本提案は研究者が新たなアイデアを生み出し，知的探索を効率化する基盤技術としての発展が期待される．

6．ナレッジグラフの構築に利用したプログラム（オプション）

- 公開先 URL：https://github.com/tanahashi415/KGRC2024/blob/main/PDF2NG.py
- 実行方法の説明：

https://github.com/tanahashi415/KGRC2024/blob/main/Use/PDF2NG.ipynb

７．資料の共有について

応募フォーム

- 公開の可否：
  （〇）公開してよい
  （　）非公開とする

応募したナレッジグラフ

- 公開の可否：
  （〇）公開してよい
  （　）非公開とする
- 公開形式：
  （〇）ナレッジグラフ推論チャレンジのサイトで公開
  （　）独自のサイトで公開してリンクを希望
  →公開先 URL（※）：

応募したプログラム等

- 公開の可否：
  （〇）公開してよい
  （　）非公開とする
- 公開形式：
  （　）ナレッジグラフ推論チャレンジのサイトで公開
  （〇）独自のサイトで公開してリンクを希望
  →公開先 URL（※）：https://github.com/tanahashi415/KGRC2024