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：矢野祐貴, 濱松大介, 富田勇希, 光國茜, 井窪奈美, 久保田菜々子, 中本裕大
- 所属：SCSK 株式会社技術戦略本部
- メールアドレス(代表)： yuuk.yano[at]scsk.jp (矢野祐貴)
  - 応募者に学生が含まれる：いいえ
  - 応募者の代表が学生である：いいえ

2. 応募部門：推理小説部門

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

推理小説部門として、下記の小説を生成対象としてナレッジグラフを構築した。

- アーサー・コナン・ドイル, まだらの紐, 翻訳 大久保ゆう and 海野十三. 青空文庫. https://www.aozora.gr.jp/cards/000009/card50717.html

構築したナレッジグラフの基本情報

本投稿では、以下の知識グラフファイルおよびその作成のための中間データを提出する。これらは応募メールの添付ファイルを解凍後、`kgrc2024-scsk-submission/outputs`以下に配置されている。

■ データサイズ

提出する各出力ファイルのサイズは下記の通りである。

- `SpeckledBand.json`: 2,628KB
- `SpeckledBand.xml`: 187KB
- `SpeckledBand.ttl`: 1,015KB

また、`SpeckledBand.ttl` には、合計 3,319 トリプル含まれている。ここでトリプルは `Subject`、`hasPredicate`、`hasProperty`、`what`、`when`、`where`、`why`、`whom`、`how`、`infoSource` のいずれかを持つものを集計した。

■ データ形式

提出する出力ファイルの形式の種類とその内容は下記のとおりである。作成方法は「ナレッジグラフの構築手法の説明」に記述する。

- `SpeckledBand.json`: 提案手法による知識グラフ構築結果である JSON 形式のシーン知識グラフ。後述の「シーン知識グラフの構築」の出力である。
- `SpeckledBand.xml`: 小説の構造的分析結果である XML 形式の構造化テキスト。後述の「発話文の分析」の出力である。
- `SpeckledBand.ttl`: 推論チャレンジで提供されている知識グラフの形式に合わせた TTL 形式のシーン知識グラフ。後述の「最終的に得られるシーン知識グラフ」の出力である。

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」

本投稿のナレッジグラフ構築における、各種サブタスクでは Azure OpenAI Service より提供されている GPT-4o を使用した。

- 言語モデル名: GPT-4o (Version 2024-08-06)
- API Version: `2024-08-01-preview`

ナレッジグラフ構築に用いた「データ」

■ 生成対象とした小説

3 章で示した通り、下記の小説のナレッジグラフを構築した。

- アーサー・コナン・ドイル, まだらの紐, 翻訳 大久保ゆう and 海野十三. 青空文庫. https://www.aozora.gr.jp/cards/000009/card50717.html

■ 大規模言語モデルの学習に用いたデータ

後述するナレッジグラフ構築のサブタスクでは、数件の入出力を In-context Learning で与えている。この数件の入出力の作成には、童話「ももたろう」をベースとして人手で作成した。これらは応募メールの添付ファイルを解凍後、`kgrc2024-scsk-submission/fewshot_samples`以下に配置されている。

■ 大規模言語モデルの推論に用いたプロンプトテンプレート

後述するナレッジグラフ構築のサブタスクに用いたプロンプトテンプレートは、応募メールの添付ファイルを解凍後、`kgrc2024-scsk-submission/src/tasks`以下のコードから確認できる。

ナレッジグラフの構築手法の説明

■ はじめに

推論チャレンジで提供されている知識グラフ（以下、RC 知識グラフ と呼ぶ）は、「犯人の推定」と「推定理由の説明」のために構築されている [古崎ら 2022]。しかし、小説内での種々の事象に対する汎用的な説明生成には、対象とする文章を限定しているため不十分である。例えば、推理に寄与しない、事件の関係者の外見・登場人物の移動・空間の描写などは除かれている事が多い。

推論チャレンジの目的から外れた描写であっても、多くの場面で有用な知識として活用される可能性がある。例えば、小説に対する情報検索や質問応答といった汎用的な説明生成タスクにおいて、また推理システムにおいても、推論に直接寄与しない周辺情報を適切に取捨選択する能力を検証するためには、「犯人の推定」に限定されない知識グラフの構築が求められる。しかし、小説全文から知識グラフを構築するには、作業に多くのコストを要する。

そこで我々は、小説内のすべての描写を対象とした、大規模言語モデルによる知識グラフ構築の仕組みを目指す。本投稿では、[古崎ら 2022] で示された知識グラフ構築手順のうち、「2. 原文を主語や目的語が明確な文（短文）に変更」と「3.自然言語処理技術を用いて、短文に意味役割（5W1H など）を付与」を対象とし、「シーンを表す短文」とそのシーンの知識グラフの構築に取り組んだ。

シーンの知識グラフ構築は、人手であっても複雑な手順を踏むため、一度の生成で実現するのは困難である。そこで、我々はシーン知識グラフ構築を、図 1 に示す複数のサブタスクに分解する。

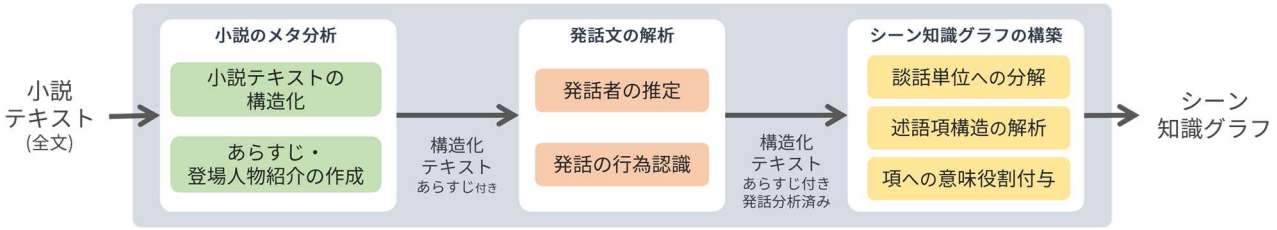


図 1. 提案するシーン知識グラフ構築のフロー図

■ 小説のメタ分析

大規模言語モデルに小説全文を入力することは困難であるため、構築の対象とする小説を処理しやすい単位に分解し、段階的にあらすじを挿入する。（図 2）

小説テキストの構造化

青空文庫から直接ダウンロードした小説テキスト全文を機械的に、章、段落、文の階層状に構造化する。また文は「」(カギ括弧)、『』(二重カギ括弧)を手がかりに、発話と地の文を区別する。

あらすじ・登場人物紹介の作成

ある一定の段落ごとに、そこまでの物語のあらすじと登場人物の紹介を大規模言語モデルによって生成し、構造化した小説テキストに要素として挿入する。今回の設定では、大規模言語モデルが処理する段落は、合計で 20 文以上になった時点とする。

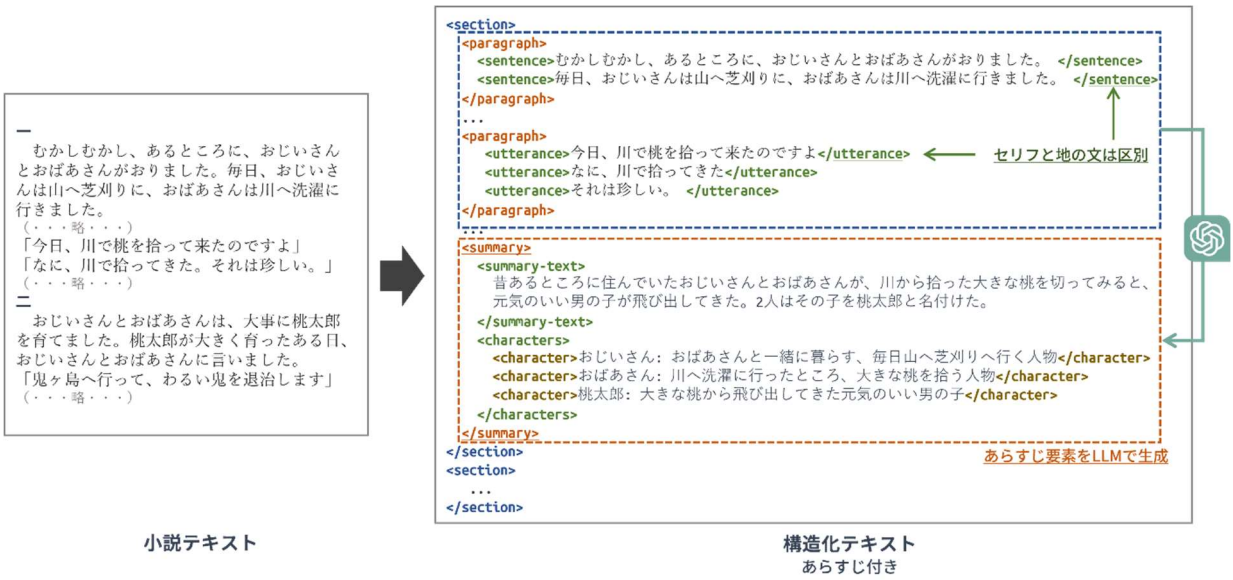


図 2. 小説テキストの構造化とあらすじ・登場人物紹介の作成

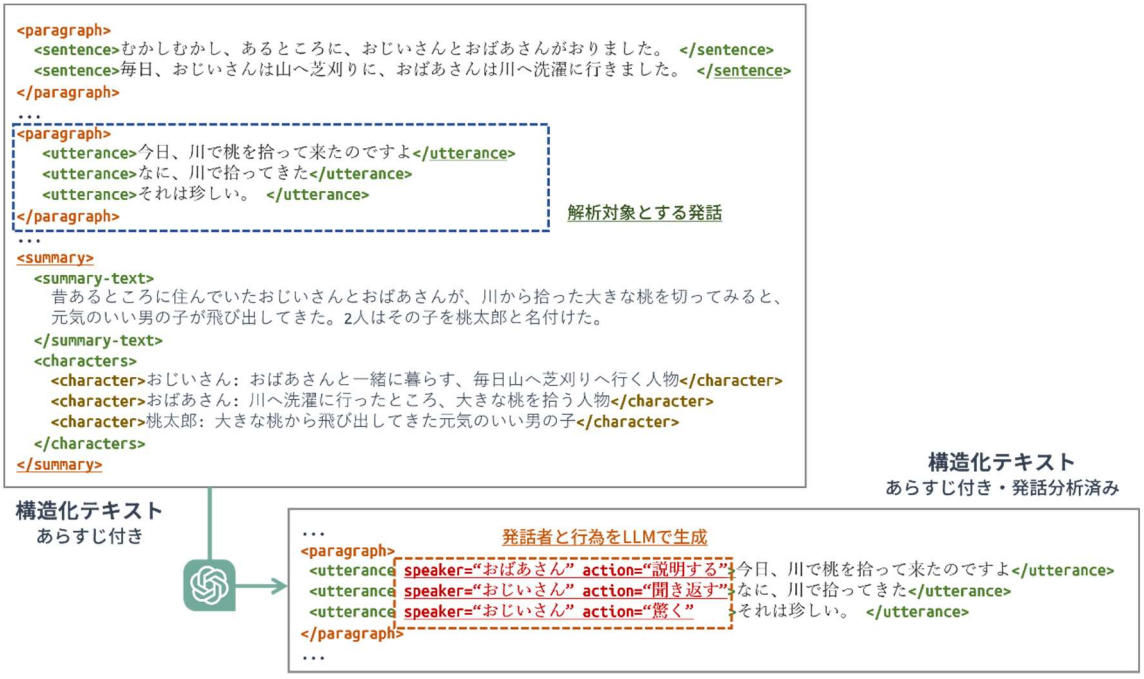


図 3. 発話者の推定と発話の行為認識

■ 発話文の解析

小説テキストのうち、発話文から知識グラフを構築するには、前後の文脈を理解したうえで、誰がどのような機能・目的で発話をしたか読み取る必要がある。そこで、地の文と同じ手法で知識グラフを構築するために、発話文を”[話者]が、「[発話文]」と[行為]する。”という形式に変換する。(図 3)

発話者の推定

前節のあらすじ作成に用いた段落を単位として、その段落に含まれる発話文の話者を大規模言語モデルにより推定する。大規模言語モデルには、あらすじと登場人物紹介、および段落の全体を文脈として与える。また、推定する話者は登場人物紹介で出現した人物に限定するように指示を行った。

発話の行為認識

発話者の推定と同様に、あらすじ要素の入力段落を単位として、その段落中の発話文が物語上でどのような機能を果たすかを認識する。大規模言語モデルへの指示としては、脚本などに用いられる「ト書き」を考えるように指示を行った。

■ シーン知識グラフの構築

「小説のメタ分析」および「発話文の分析」によって整理された、小説の個々の文からシーン知識グラフを構築する。RC 知識グラフでは、原文を主語や目的語が明確な短文に人手で変更し、その短文に自然言語処理技術によって意味役割付与を施している。我々の手法では、まず原文から述語を中心とした、それ以上分解のできない最小単位の文へ分解し、これを「シーン」とする。続いて、このシーンの文に対して述語項構造の解析を行い、述語と項それぞれに RC 知識グラフと同じような意味役割ラベルを付与することで、シーンの知識グラフを構築する。(図 4)

シーンの分解

日本語の 1 文は、1 つ以上の述語から構成されており、そのまま RC 知識グラフの 1 シーンを作成しようとすると、`hasPredicate` や `Subject` といった必須かつ単一の要素が重複してしまう。そのため、シーン知識グラフの構築の前に、小説中のそれぞれの文を、大規模言語モデルを用いて述語を主辞とするそれ以上分けられない最小単位の文（シーン文）へと分解する。さらに大規模言語モデルへの指示として、後続の意味役割付与に役立てるため、述語の種類である「名詞述語」「動詞述語」「形容詞述語」も識別するように指示した。

述語項構造の解析 と 意味役割の付与

述語を主辞としたシーン文に対して、複雑な文の構造や表層の違いを正規化する為に、述語と項の構造を大規模言語モデルによって解析する。

得られた述語と項の関係を入力として、それぞれに意味役割ラベルを付与する。今回は評価時に RC 知識グラフとの比較を行うため、述語に対して `hasPredicate` と `hasProperty` のいずれかを、項に対しては `Subject` または 5W1H のいずれかを大規模言語モデルを用いて付与する。また発話文については、「発話文の解析」で推定した話者を `infoSource` として登録する。

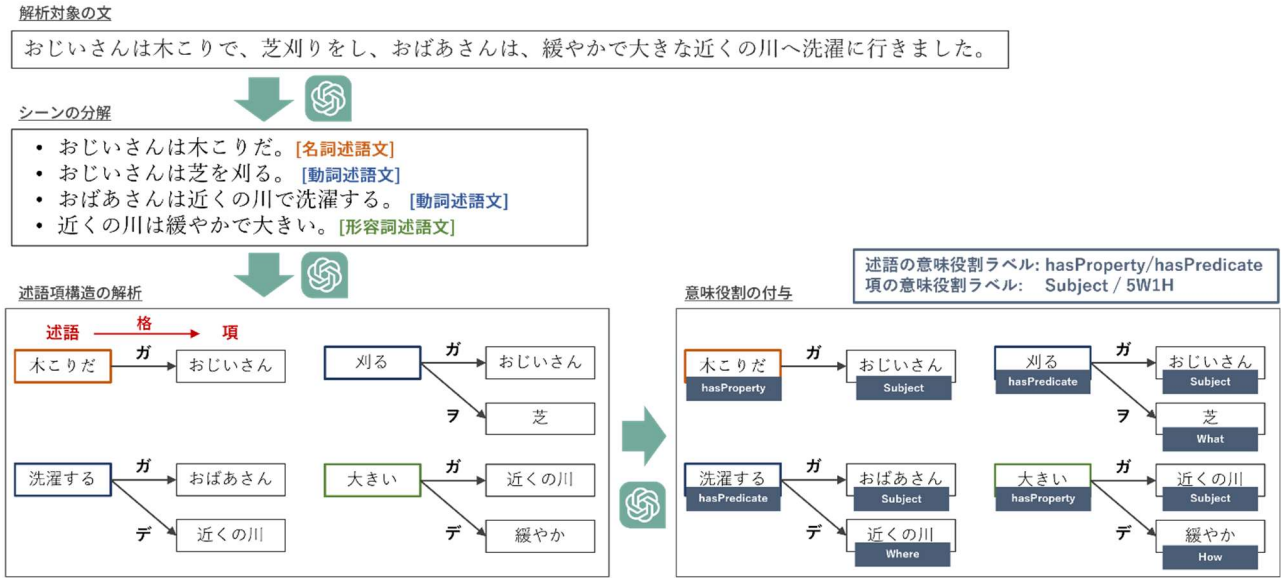


図 4. シーンの分解と述語項構造の解析および意味役割の付与

■ 最終的に得られるシーン知識グラフ

小説全文に対して、意味役割の付与まで実施後、それぞれのシーンに先頭から ID を割り当て RC 知識グラフのシーンと同じ構造に変換した。また、RC 知識グラフと比較するため、述語および項をそれぞれ Google 翻訳を用いて英語表記にした。最終的に得られた TTL 形式の知識グラフの一部をいかに示す。

```
<http://kgc.knowledge-graph.jp/data/SpeckledBand/1145>
    kgc:hasPredicate <http://kgc.knowledge-graph.jp/data/predicate/explain> ;
    kgc:how <http://kgc.knowledge-graph.jp/data/SpeckledBand/brief> ;
    kgc:source "ホームズが起こった事を手短に説明する。"@ja ;
    kgc:source_discourse "とホームズは言い、起こった事を手短に説明した。"@ja ;
    kgc:subject <http://kgc.knowledge-graph.jp/data/SpeckledBand/holmes> ;
    kgc:what <http://kgc.knowledge-graph.jp/data/SpeckledBand/What_happened> .
```

ここで`kgc:source`は小説から抜き出した、このシーン知識グラフを構築するための原文であり、`kgc:source_discourse`は「シーンの分解」によって得られたシーン文である。

パフォーマンス情報

本投稿のナレッジグラフ構築では、各サブタスクとも Azure API で生成している。したがって、各サブタスクの平均入出力トークン数および平均処理時間とそれぞれの処理単位について下記に報告する。

サブタスク	平均入力トークン数 [token]	平均出力トークン数 [token]	平均処理時間 [sec]	処理単位
あらすじ・登場人物紹介の作成	3161.84	450.80	9.48	20 文以上の段落(※)
発話者の推定	3917.80	825.48	19.15	20 文以上の段落
発話の行為認識	3550.01	783.62	14.23	20 文以上の段落
シーンの分解	1376.95	70.97	2.29	1 文
述語項構造の解析	944.85	67.18	1.88	1 シーン文
意味役割の付与	932.18	80.10	2.10	1 シーン文

(※)「小説のメタ分析」で述べた、最低 20 文以上で構成される段落もしくは段落の集合

参考情報

- 古崎晃司，江上周作，松下京群，鵜飼孝典，川村隆浩 (2022). "説明生成のための知識グラフ構築ガイドラインの考察 - ナレッジグラフ推論チャレンジを例にして." 人工知能学会全国大会論文集.
- 笹野遼平，飯田龍，奥村学 (2017). “文脈解析 - 述語項構造・照応・談話構造の解析. ” コロナ社.
- Azure OpenAI Service モデル – Microsoft Learn
  Available: https://learn.microsoft.com/ja-jp/azure/ai-services/openai/concepts/models

5. 構築したナレッジグラフの評価

評価の概要

RC 知識グラフを参照して、本投稿のシステムによって生成されたデータについて人手評価を実施する。特に、本投稿システムの動機である、推理に寄与しない小説内で記述された情報が追加できているのか、RC 知識グラフと比較した際の情報の欠落度合いなどを分析する。そのため、RC 知識グラフと生成した知識グラフで対応するシーンのみを抜き出し、それらの知識グラフ同士を比較した際、片方のみに出現するトリプルを分析する。(図 5)

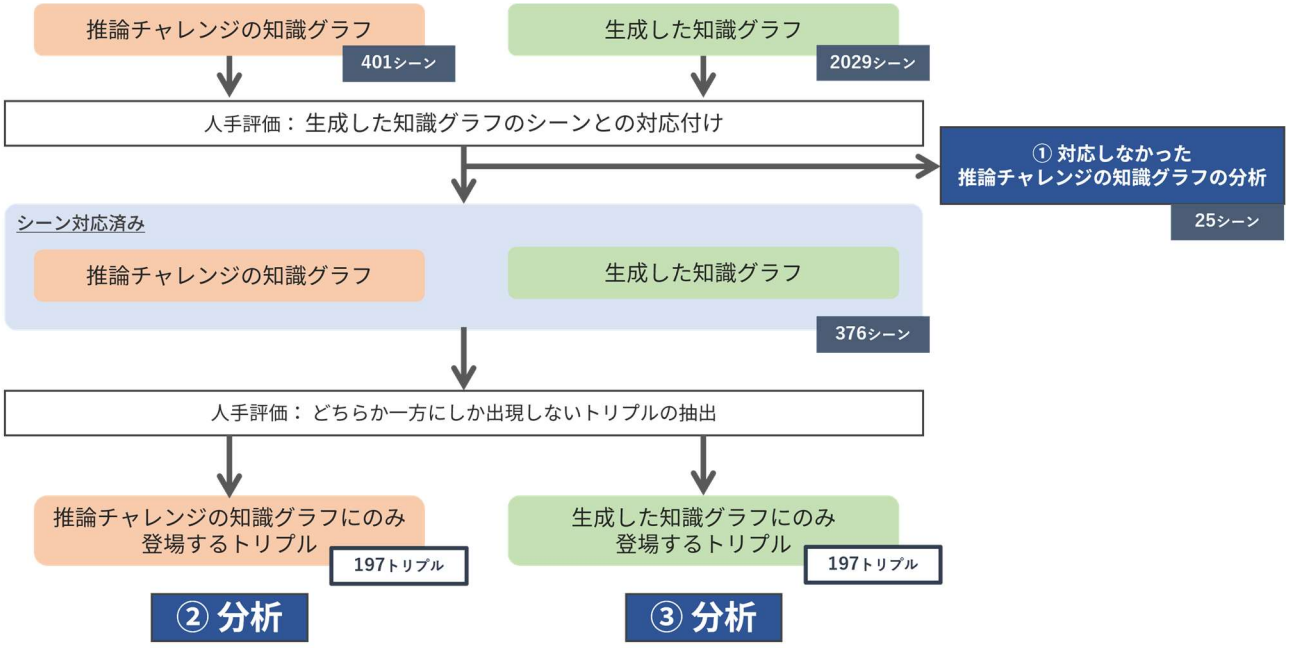


図 5. 評価分析の概要

① 対応しなかった推論チャレンジの知識グラフの分析

まず、RC 知識グラフの各シーンに対し、本システムのどのシーンが対応するかを人手関連付ける作業を行った。具体的に、RC 知識グラフの `source` や時系列的な情報から、小説原文のどこから作成されたシーンかを捉え、同じ原文から生成した知識グラフ側のシーンと対応付ける。シーンによっては、RC 知識グラフ側の 2 シーンと生成した知識グラフ側の 1 シーンが対応付けられたり、反対に RC 知識グラフ側の 1 シーンと生成した知識グラフ側の 2 シーンが対応したりするケースが見受けられた。

結果として、RC 知識グラフの全 401 シーン中、376 シーンが生成した知識グラフ側の 1 つ以上のシーンと対応付けられた。対応付けられなかった RC 知識グラフの 25 シーンについては、そもそも”source”がないシーンや、when にタイムスタンプのみのケースがみられた。

- **対応付けられなかったシーンについて**：

  `source` があったが対応付けられなかったシーンとして「*姉妹とはヘレンとジュリアである。*(scene ID: 035)」がある。前後のシーンを考慮すると、このシーンの原文とされるのは全く関係のない次の 1 文となる。

  > *ジュリアは二年前のクリスマスにそこへ参りまして、休職中の海軍少佐の方と出会い、婚約の運びとなったのでございます。*

  同様に、生成した知識グラフと関連付けられなかったシーンとして「*ロマは楽団である*(scene ID: 401)」があり、これも原文中から読み取れることはできない。このように、生成した知識グラフのシーンの中には、時系列的に不自然かつ原文中から読み取ることのできないシーンが存在し、これがどういう機序で発生したかを特定することは困難である。

② RC 知識グラフにのみ出現するトリプルの分析

RC 知識グラフにのみ出現するトリプルについて分析を行った結果、原因として以下 3 つの要因が考えられる。

I. source の内容に差異がある:
II. RC 知識グラフの内容に、source のみでは作成が難しいものが存在する
III. 生成した知識グラフで、 source に記載があるが知識グラフに内容が反映されていない

■ I. source の内容に差異がある

本システムでは、原文に登場する 1 文から複数の文に要素を分けてシーンを作成する。結果として、RC 知識グラフでは 1 つのシーンとして作成される内容が、複数のシーンとして作成されるケースが見られた。以下に、原文に登場する 1 文から作成されたシーンの例を示す。

原文

今のわたくしではお礼も十分には致しかねますが、あとひと月ふた月のあいだに結婚して、お金を自由にできるようになりますので、そうすれば相応のお礼もできるかと思います

推論チャレンジの知識グラフ

source	ヘレンは２ヶ月以内にお金を得る
subject	Helen
hasPredicate	get
infoSource	Helen
what	money
when	1883-06-01, within_2_months
how	

生成した知識グラフ

source	若いご婦人はお金を自由にできるようになる。
subject	Young_lady
hasPredicate	Be
infoSource	Young_lady
what	
when	
how	free_your_money

上記の内、when の within_2_months という内容が RC 知識グラフのみに出現する要素となるが、本システムの場合、「２か月以内」という要素は別のシーンに分けられている。そのため、該当単語は RC 知識グラフのみに登場することとなった。

■ II. RC 知識グラフの内容に、source のみでは作成が難しいものが存在する

以下の例に示すように、source から読み取れない内容が知識グラフに含まれているケースが存在する。

原文

ただいま義理の父親のロイロット博士と一緒に住んでおります。

推論チャレンジの知識グラフ

source	へレンはロイロットと住んでいる
subject	Helen
hasPredicate	live
infoSource	
when	1883-04-01
where	mansion_of_Roylott
whom	
how	

生成した知識グラフ

source	へレン・ストーナがただいまロイロット博士と一緒に住んでいる。
subject	Helen_Stona
hasPredicate	live
infoSource	Helen_Stona
when	I'm_home
where	
whom	Dr._Loylot
how	together

上記のシーンの内、RC 知識グラフの where に mansion_of_Roylott という表現が出現している。mansion_of_Roylott は、元となった原文の 1 文後の文からわかる内容であるが、これは RC 知識グラフの source には記載されておらず、同様に生成した知識グラフの source にも記載はない。このように、RC 知識グラフでは、 source に記載されていない情報も知識グラフ化する傾向があるため、上記のようなケースが発生する。

■ III. 生成した知識グラフで、 source に記載があるが知識グラフに内容が反映されていない

以下の例に示すように source に記載がある情報が、生成した知識グラフに反映されないケースが存在する。

原文

でも、あたし、姉さんと違って、安眠しちゃうから。

推論チャレンジの知識グラフ

source	へレンは安眠している
subject	Helen
hasPredicate	
hasProperty	asleep
infoSource	Helen
when	1881-04-01T03
how	deeply

生成した知識グラフ

source	あたしは安眠してしまう。
subject	I
hasPredicate	Fall_asleep
hasProperty	
infoSource	Helen_Stona
when	
how	

上記のシーンでは互いの source 文に「安眠」と記載されている。RC 知識グラフでは how に deeply という要素が作成されるが、生成した知識グラフでは対応する表現が何も生成されていない。 このように同じ表現ではあるが、RC 知識グラフのみで追加されるケースが見受けられた。

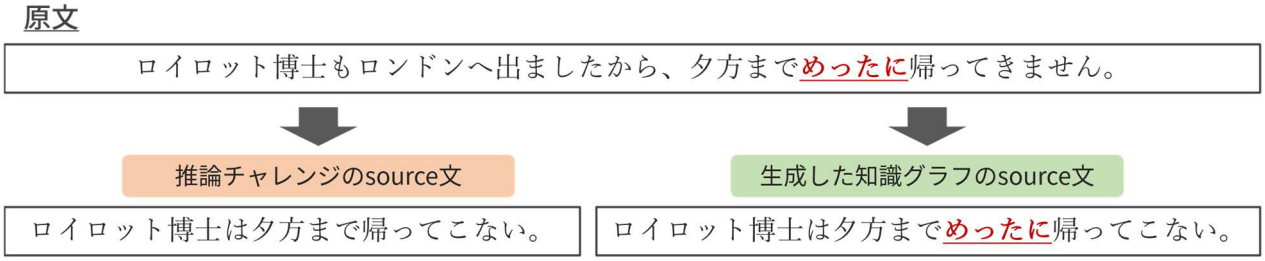
③ 生成した知識グラフにのみ出現するトリプルの分析

生成した知識グラフのみに出現するトリプルについて分析を行った結果、原因として以下 2 つの要因が考えられる。

I. source の内容に差異がある
II. RC 知識グラフの source が原文の表現から変化している

■ I. source の内容に差異がある

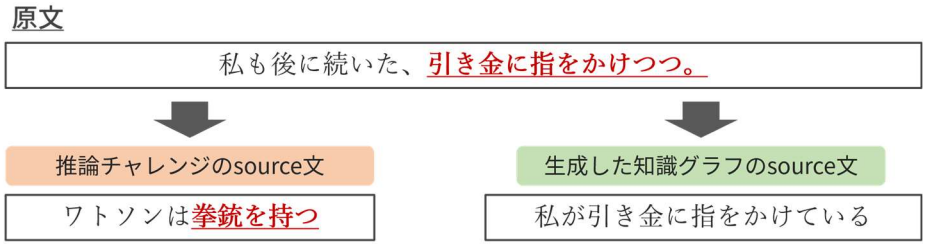
原文から source を作成する際、RC 知識グラフでは修飾語が省略されるケースが見受けられる。



上記の例では、「めったに」という要素が RC 知識グラフでは消失している。 以上のように、一部の修飾語が source で省略されることで、生成した知識グラフのみに出現する要素が発生する。

■ II. RC 知識グラフの source が原文の表現から変化している

以下に例を示すように、RC 知識グラフの source が原文の表現から変化しているケースが存在する。



上記の通り「引き金に指をかける」という表現を、RC 知識グラフでは「拳銃を持つ」という表現に直して知識グラフ化している。 以上の通り、 RC 知識グラフの source が原文の表現を変化させているケースで生成した知識グラフのみに出現するトリプルが作成された。

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）

※本応募で使用した各種プログラムは、後日整理した状態でご連携させていただきます。

7. 資料の共有について

応募フォーム

- **公開の可否：**
  （◯）公開して良い
  （　）非公開とする

応募したナレッジグラフ

- **公開の可否：**
  （◯）公開して良い
  （　）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
  （　）独自のサイトで公開してリンクを希望

応募したプログラム等

- **公開の可否：**
  （◯）公開して良い
  （　）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
  （　）独自のサイトで公開してリンクを希望