第1回ナレッジグラフ推論チャレンジ 2018 応募シート

1. 応募者に関する情報

-チーム名：teamOIF
-チームメンバー
-岡田知輝
-今園真聡
-福島良紀
- 所属：立命館大学ナレッジコンピューティング研究室
- メールアドレス（代表）：is0246if@ed.ritsumei.ac.jp

2. 推論過程の説明

私たちの提案する推論システムでは、犯人の動機をナレッジグラフから検索することで、犯人を特定する。推理小説では、小説ごとに人物の置かれている状況や使用される犯行道具は異なるため、犯行手法は無数に存在するといえる。そのため、推論ルールを決めてナレッジグラフから犯行手法に関わる特定の情報を抽出する、というのは難しい。一方で、犯人が持つ動機は嫉妬や好悪といった人間が普遍的に持ちうる感情に依存するため、犯行手法に注目する場合とは異なり推理小説全般に適応可能な汎用的推論プロセスを定義することができる。よって、私たちは動機の検索が有効だと考えた。

動機推論

まず、動機推論の方法について説明する。推論は「犯人候補の検索」「動機の特定」の 2 段階に分けて行う。この 2 つの処理終了後に、各犯人候補の動機の数をカウントし、推定された動機が一番多かった人物を犯人とする。

犯人候補の検索

はじめに、犯人候補の検索について説明する。「犯人候補の検索」では、「ノックスの十戒」を参考にし、推理小説中から犯人の可能性がある人物を犯人候補として抽出する。推論は以下の 5 つの段階に分けて行った。

1. 全人物の取得
2. 探偵役の除外
3. 前半登場人物に絞り込み
4. 名前が設定されているものに絞り込み
5. 前半に死亡している人の除外

まず「1.全人物の取得」では、犯人は人間であると仮定し、対象の推理小説に登場する人間をすべて取得した。
次に「2.探偵役の除外」では、ノックスの十戒の「The detective must not himself commit the crime.」を参考に、取得した人物からホームズやワトソンなどの探偵役を除外した。
次に「3.前半登場人物に絞り込み」では、ノックスの十戒の「The criminal must be someone mentioned in the early part of the story, but must not be anyone whose thoughts the reader has been allowed to follow.」を参考に、犯人は物語の前半に登場していると仮定して、前半に登場している人間に絞り込んだ。
次に「4.名前が設定されているものに絞り込み」では、犯人は小説中で主要な人物である場合が多いため、名前を持っているものと仮定し、名前を持っている人物に絞り込んだ。
最後に「5.前半に死亡している人の除外」では、物語の前半に死亡している人間は、犯人に殺された被害者または、事件との関連性が薄い人物であると考えられるため（まだらの紐ではジュリアやヘレンの母が該当する）、前半に死亡している人物を除外した。
これらの 5 つ処理によって、推理小説の登場人物から犯人候補を抽出する推論を行った。

動機の特定

次に、「動機の特定」について説明する。動機を、犯人が犯行を行うための理由と定義し、動機を金銭、怨恨、男女関係の 3 種類のカテゴリに分類した。金銭は、お金や資本が関係している動機、怨恨は、過去の行為、特性に対しての嫉妬や嫌悪などの感情に起因する動機、男女関係は、婚姻関係や愛人関係など主に男女間の関係に起因する動機と定義した。また、これらの動機の文に、頻出する単語を動機キーワードとして定義した。「動機の特定」では、犯人候補に対してこれらの動機となる文を持っているか推論を行う。具体的な推論方法としては、以下の処理を行った。

I. 各犯人候補が関連している文の中から動機キーワードを含む、動機文候補を抽出
II. 各動機文候補に対して
    (ア) 動機文の利害関係者を抽出
    (イ) 利害関係者が被害者の場合、確定動機文と判定

具体的にはまず、動機文を検索するために、ホームズシリーズの作品から動機に関連のある名詞と動詞を収集し、それをカテゴリごとに分けた動機キーワード群を作成した。次に、前述した犯人候補の人物に対して、その人物が関わっている文章と動機キーワードが同時に現れる文章を動機文候補として抽出した。次に、各動機文候補に対して、関連する文章とその主語と目的語となっている人物を利害関係者として抽出した。最後に、その利害関係者が対象の小説中で被害に遭っているかどうかを判定し、被害にあっている場合に、確定動機文とした。被害に遭っているかどうかの判定は、まず、動機と同様に、ホームズシリーズの作

品から被害関係に関連のある名詞と動詞を収集し、被害キーワード群を作成し、それを利用し被害者を検索した。そして、その被害者に利害関係者が含まれているかどうかで被害者判定を行った。

オントロジー

本システムでは、動機のキーワード群や探偵の名前、推論工程などをオントロジーとして定義した。オントロジーデータは法造を用いて作成した。

推論工程について

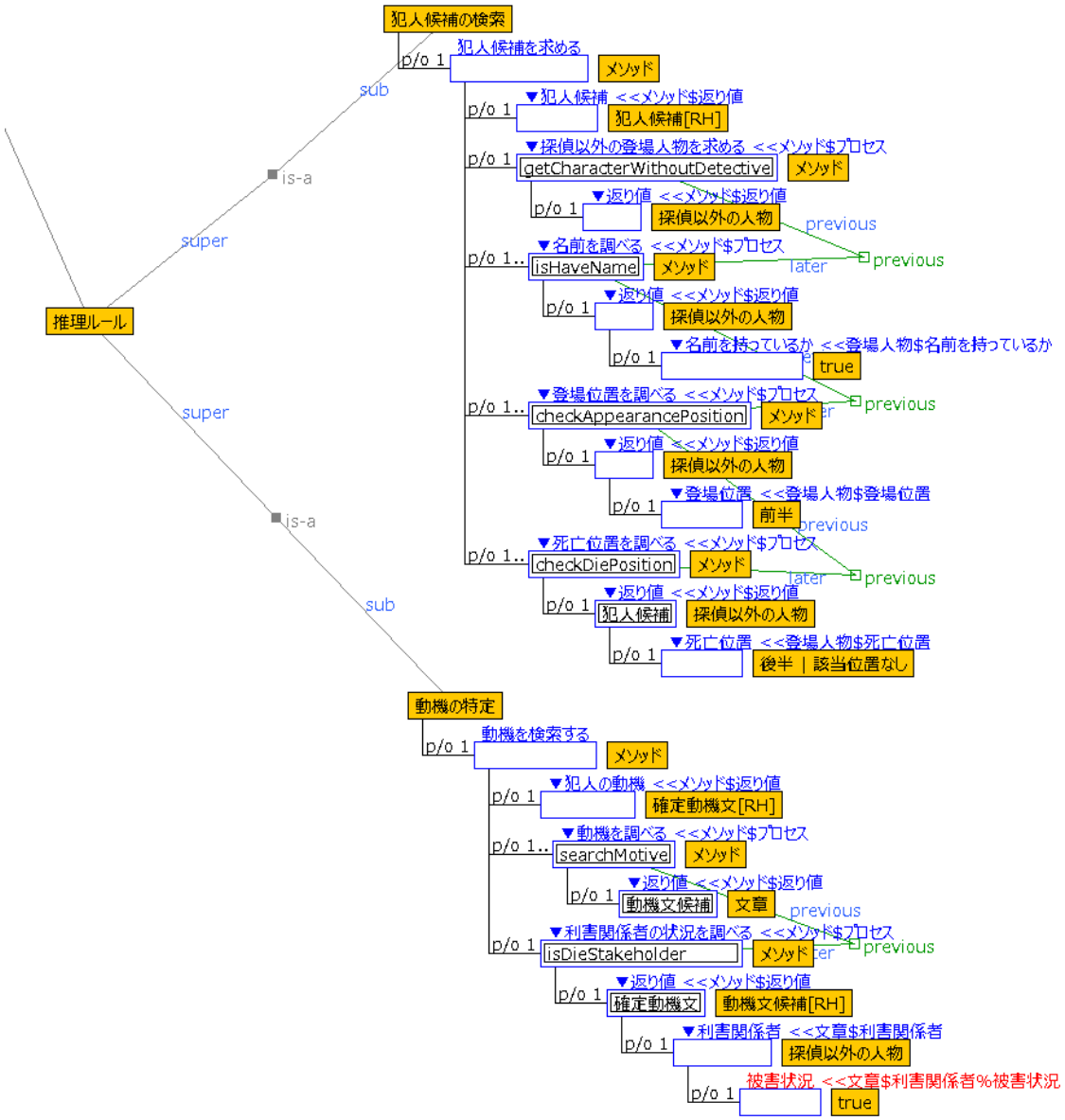

図 1：推理ルール

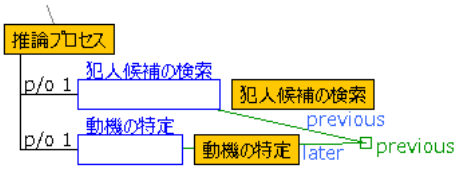

図 2：推論プロセス

推論工程については上の図のように定義される。「推理ルール」概念の下位概念に「犯人候補の検索」「動機の特定」という概念が定義されており、それらのスロットでは推論の過程が示されている。スロットの位置ではメソッドの前後関係は定義されないため、関係概念として previous 概念を定義してメソッド間の前後関係を表現している。さらに、「推論プロセス」概念でも同様の定義を行い、「犯人候補の検索」「動機の特定」の順序関係を定義している。

キーワード群について

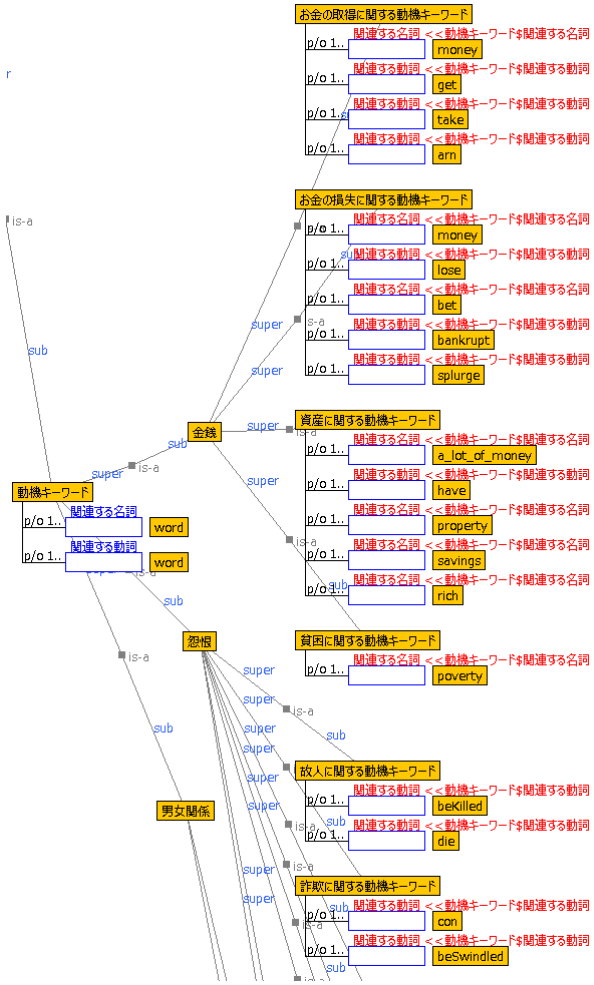

図 3：動機に関する諸概念

動機に関する概念は上の図のように定義される。動機は「金銭」「怨恨」「男女関係」の三種類に分類されている。それぞれの下位概念にはその分類に対応する動機群が定義されており、スロットにはその動機に関連する名詞や動詞が定義されている。例えば、金銭の下位概念には「お金の取得に関する動機」が定義されており、スロットにはクラス制約に「money」「get」「take」「earn」を持つスロットが定義されている。これにより、シス

テムは【「money」「get」「take」「earn」といった単語が出現する文章は、お金の損得に関わる動機を表す文章である可能性が高い】、という推論が可能になる。

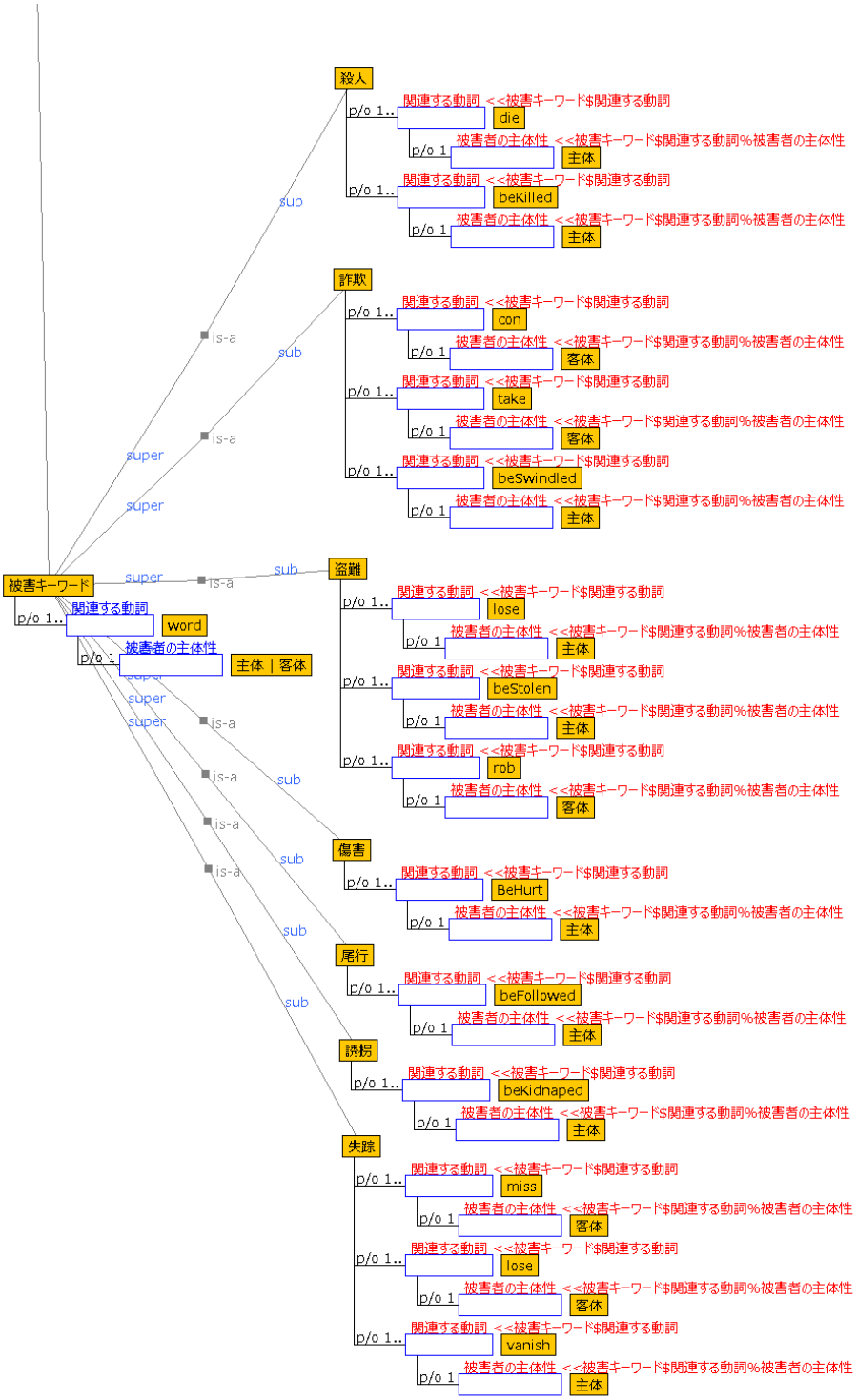

図 4：被害に関する諸概念

被害に関する概念は上の図のように定義される。被害キーワード概念の下位概念には、具体

的な被害内容を表す概念が定義されており、スロットにはその被害に関連する動詞が定義されている。関連する動詞のスロットには「主体」あるいは「客体」をクラス制約とする概念が定義されており、これに基づいて被害者の特定が可能になる。例えば、「殺人」という概念は、スロットの 1 つに「die」をクラス制約とするものがあり、そのスロットは「主体」をクラス制約とするサブスロットを持つ。これにより、【die の主語は殺人の被害者である】、という推論が可能になる。

フレームワーク

図 5 に本システムのフレームワークを示す。本システムは法造で作成された、探偵のデータ、動機キーワード群、推論過程などが定義されているオントロジーデータを読み込んで動作する。オントロジーデータ内の定義内容は法造 API を使用して、参照する。まず、ユーザは、本システムに対して、小説前半部の最後に該当する ID（以下、前半 ID と呼ぶ）を入力する。その後、本システムがオントロジーデータと小説のナレッジグラフを探索することで、犯人とその動機を推論し、出力する。

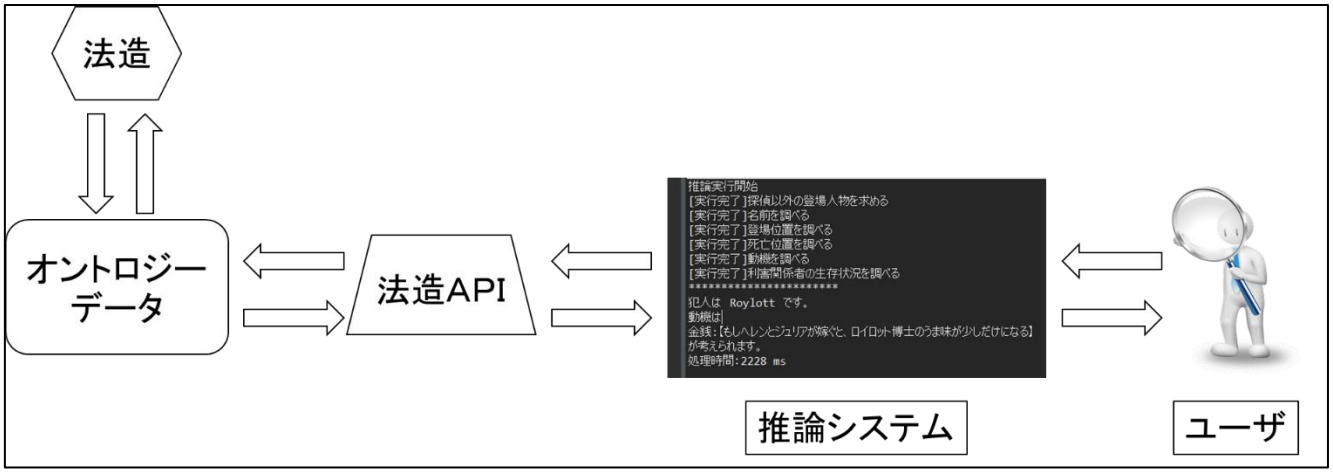

図 5：フレームワーク

システムの出力

次に推論システムを、ログを用いて説明する。システムのログとしてはシステム実行状況、犯人名、犯行動機、処理時間の 4 つの情報が出力される。システム実行状況は各推論ステップの開始と終了のタイミングに、「[実行開始 or 終了]推論ステップ名」が表示され、終了のタイミングに推論の状況として「現在の犯人候補の人数と名前」or「各犯人候補に存在する動機」が表示される。

まず、システムを実行すると、オントロジーに記述されている推論過程を読み込み、「犯人候補の検索」ステップの「探偵以外の登場人物を求める」ステップが実行され、図 6 が表示される。

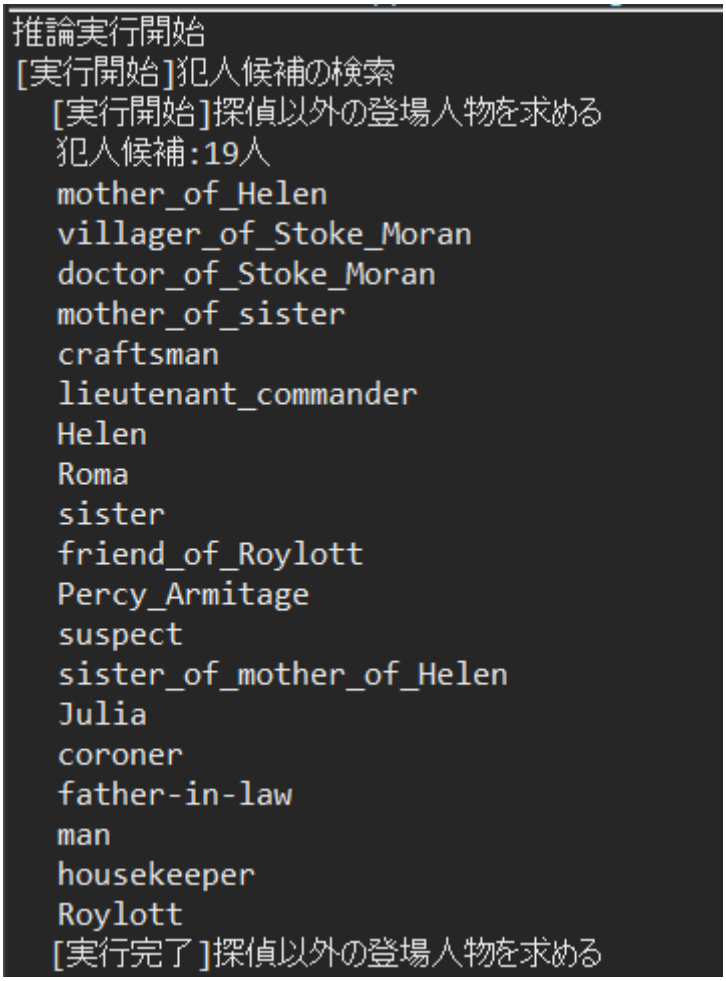

図 6：「探偵以外の登場人物を求める」ステップのシステムログ

「探偵以外の登場人物を求める」ステップでは、ナレッジグラフに対して、下記の sparql 文を実行し、人物を犯人候補として取得する。その後、取得した人物からオントロジーに記述されている探偵役の人物を除外する。

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX kgc: <http://kgc.knowledge-graph.jp/ontology/kgc.owl#>
PREFIX kd: <http://kgc.knowledge-graph.jp/data/SpeckledBand/>
SELECT DISTINCT ?s WHERE {?s rdf:type kgc:Person.}"
```

このステップ時点では、犯人候補として 19 人の名前が表示されている。

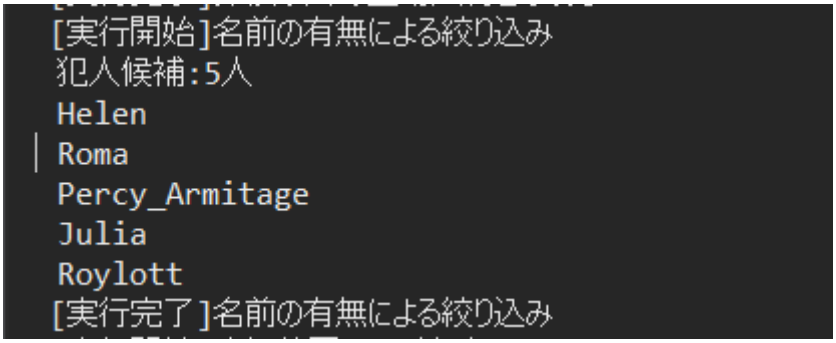

図 7:「名前の有無による絞り込み」ステップのシステムログ

次に、「名前の有無による絞り込み」ステップが実行され、図 7 が表示される。このステップ時点では、犯人候補として 5 人の名前が表示されている。

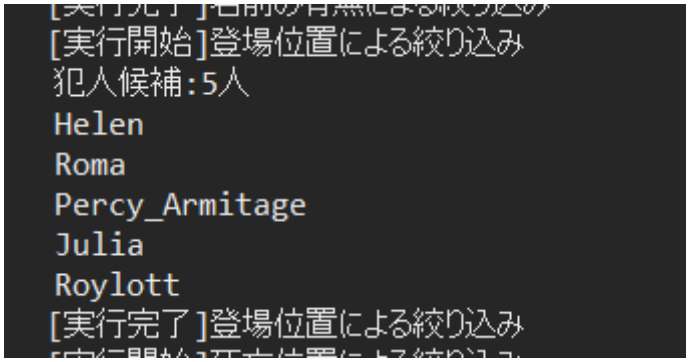

図 8:「登場位置による絞り込み」ステップのシステムログ

次に「登場位置による絞り込み」ステップが実行され、図 7 が表示される。このステップではナレッジグラフに対して、下記の sparql 文を実行し、対象の人物が出現している ID を取得する。その後、取得した ID が前半 ID と比較し、前半に出現していない場合は犯人候補から除去する。

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX kgc: <http://kgc.knowledge-graph.jp/ontology/kgc.owl#>
PREFIX kd: <http://kgc.knowledge-graph.jp/data/SpeckledBand/>
SELECT DISTINCT ?id WHERE {?id kgc:subject kd:人物名.}
```

このステップ時点では、犯人候補として 5 人の名前が表示されている。

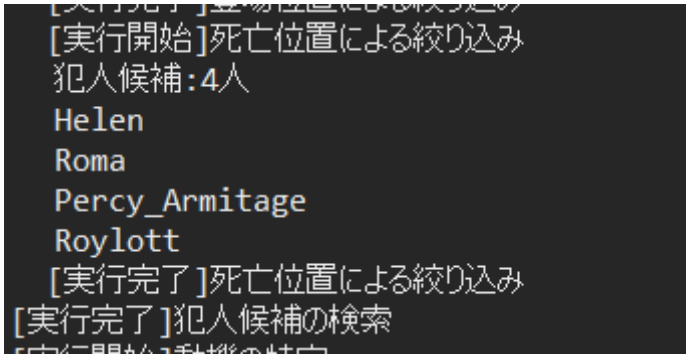

図 9:「死亡位置による絞り込み」ステップのシステムログ

次に、「死亡位置による絞り込み」では、下記の sparql 文を実行し、対象の人物が死亡したと考えられる ID を取得した。その後、取得した ID と前半 ID を比較し、前半に出現している場合は犯人候補から除去する。

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX kgc: <http://kgc.knowledge-graph.jp/ontology/kgc.owl#>
PREFIX kd: <http://kgc.knowledge-graph.jp/data/SpeckledBand/>
SELECT DISTINCT ?id WHERE {?id kgc:hasPredicate kd:die ;
                                kgc:subject kd:人物名.}
```

このステップ時点では、犯人候補として 4 人の名前が表示されている。また、このステップが終了することで、「犯人候補の検索」ステップが完了する。

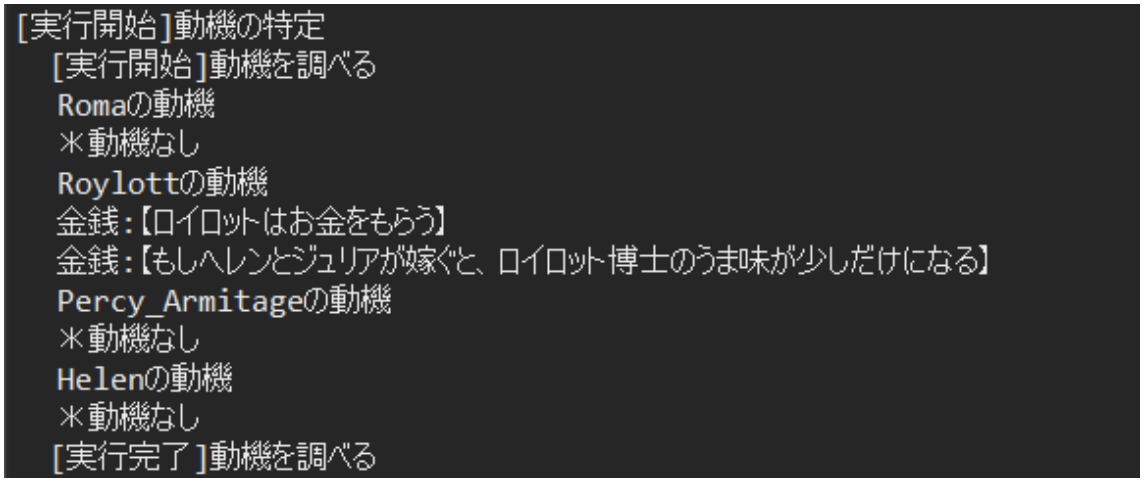

図 10:「動機を調べる」ステップのシステムログ

「犯人候補の検索」ステップ終了後、「動機の特定」ステップの中の「動機を調べる」ステップが開始される。このステップでは、オントロジーで定義されている動機キーワードに基づき、下記の sparql 文を実行する。これにより取得した文を動機文候補として、人物とカテゴリとセットで保存する。

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX kgc: <http://kgc.knowledge-graph.jp/ontology/kgc.owl#> PREFIX kd: <http://kgc.knowledge-graph.jp/data/SpeckledBand/> "WHERE {?s kgc:subject kd:***人物名***; kgc:hasPredicate kd:***動機キーワード（動詞）*** kgc:what kd:***動詞キーワード（名詞）***.}

このステップ時点では、ロイロットに 2 つの動機文候補があることが表示されている。

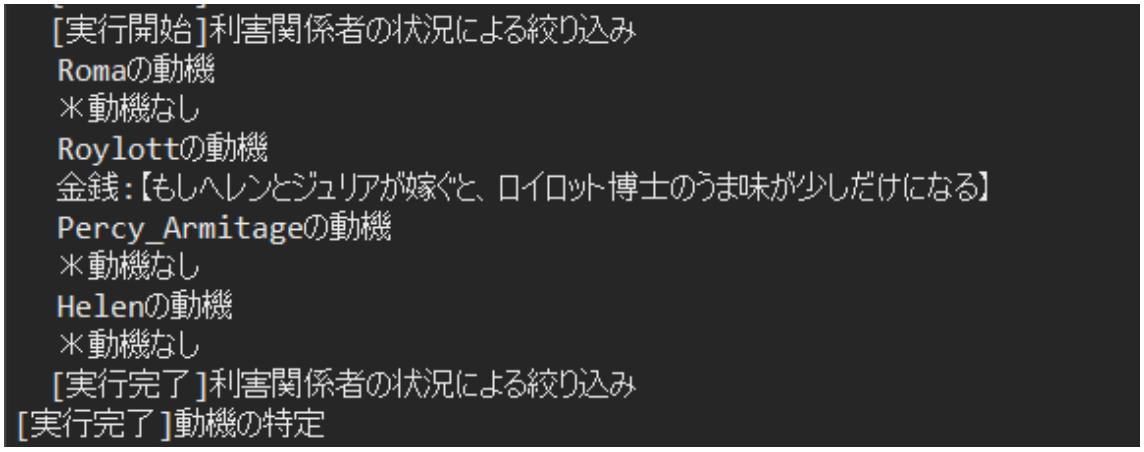

図 11:「利害関係者の被害状況による絞り込み」ステップのシステムログ

次に、「利害関係者の被害状況による絞り込み」ステップが実行される。このステップでは、まず、オントロジーで定義されている被害キーワード群に基づき、下記の sparql 文を実行し、小説中に何らかの被害を受けていると推測できる人物を取得し、被害者群として定義する。

被害キーワードの主体スロットのクラス制約が「主体」の場合
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX kgc: <http://kgc.knowledge-graph.jp/ontology/kgc.owl#> PREFIX kd: <http://kgc.knowledge-graph.jp/data/SpeckledBand/> SELECT DISTINCT ?id ?s WHERE {?id kgc:hasPredicate kd:***被害キーワード***; kgc:subject ?s.}
被害キーワードの主体スロットのクラス制約が「客体」の場合
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX kgc: <http://kgc.knowledge-graph.jp/ontology/kgc.owl#>

```
PREFIX kd: <http://kgc.knowledge-graph.jp/data/SpeckledBand/>
SELECT DISTINCT ?id ?s
WHERE {?id kgc:hasPredicate kd:被害キーワード;
          kgc:whom ?s.}
```

次に、下記の sparql 文を実行し、動機文候補の利害関係者を取得し、先ほど取得した被害者群と比較する。動機候補文から取得した利害関係者が一人でも被害者群に属している場合、対象の動機文候補を確定動機文とする。

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX kgc: <http://kgc.knowledge-graph.jp/ontology/kgc.owl#>
PREFIX kd: <http://kgc.knowledge-graph.jp/data/SpeckledBand/>
SELECT ?s WHERE {kd:動機文候補の ID kgc:if ?id.
                    ?id kgc:subject ?s}
```

このステップ時点では、ロイロットに確定動機文が 1 つあることが表示されている。また、このステップが終了することで、「動機の特定」ステップが完了し、すべての推論ステップが完了する。

```
**********
犯人は 『Roylott』 です。
動機は
金銭:【もしヘレンとジュリアが嫁ぐと、ロイロット博士のうま味が少しだけになる】
が考えられます。
処理時間:4601 ms
```

図 12: 犯人の名前、犯行動機、処理時間のシステムログ

最後に推論の結果として図 12 が表示される。図 12 では、犯人の名前、犯行動機、処理時間が表示されている。犯行動機は「動機カテゴリ:【確定動機文】」の形式で表示され、犯人が何のために犯行を行ったのかが表示される。処理時間では、プログラム実行から犯人出力までの時間が表示されている。まだらの紐を対象とした本システムの平均処理時間は 4786ms であった。

もし、推論システムが犯人を推定できなかった場合は、図 13 のようにエラーメッセージが表示される。

```
**********
推定不可能です
動機の推定では犯人はわかりません
処理時間:2741 ms
```

図 13:犯人推論不可能な場合のシステムログ

課題

本システムの課題としては、大きく 2 つある。1 つ目は、小説の前半の定義をユーザがしないといけないことである。本システムの入力として、ユーザから前半 ID 番号を受け取り、ナレッジグラフを使う部分を指定している。そのため、物語の進行を全登場人物の出現割合や、探偵役の行動から推測する必要があると考えられる。2 つ目は探偵役が動的に取得できないことである。本システムでは、探偵役の取得を小説のナレッジグラフからではなく、オントロジーから取得している。そのため、探偵役がホームズやワトソンでない推理小説に適応させることができない。出現回数での絞り込みや登場人物との関連性などを抽出し、探偵役となっている人物を動的に生成できるようにする必要があると考える。

使用した知識の範囲

ID:1-288

計算機のスペック

ThinkPad X250, Memory 8GB, Core i7 2.6Ghz, Windows10

3. 実行プログラム

- git の URL

https://github.com/teamOIF2018/CriminalDetector

- 実行方法の説明

上記サイトからプログラあムをダウンロードし、main クラスを実行する。引数として小説前半部最後の ID（288）を入力する。

- 使用したツール
  - Java1.8
  - 法造 Version 5.6(オントロジー構築利用環境)