ナレッジグラフ推論チャレンジ2023 応募シート

1. 応募者に関する情報

-氏名またはチーム名：堀田将吾
-所属：大阪電気通信大学情報通信工学部情報工学科
-メールアドレス（代表）：gp20a136@oecu.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：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
  The Speckled Band

- 構築したナレッジグラフの基本情報
  - データサイズ
    llama2による出力:8370ファイル,1.1MB(圧縮)
    その内のトリプル数:686トリプル

  - データ形式
    テキストファイル。JSON形式で記述している。

- 構築したナレッジグラフのデータの入手先
  googledriveの共有URL
  ファイル名: llama_triple.tar.gz
  https://drive.google.com/drive/folders/1ElB_bQccJQe07uLeXFg3w5DRm7SiZO1I?usp=drive_link

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」
  llama2 (量子化モデル)
  種類: llama-2-70b-chat.Q5_K_M.gguf ,size: 48.75GB, 最大RAM: 51.25 GB, 実行: llama.cpp, URL: https://huggingface.co/TheBloke/Llama-2-70B-Chat-GGUF

  BERT
  種類: bert-base-japanese-whole-word-masking, URL: https://huggingface.co/cl-tohoku/bert-base-japanese-whole-word-masking

ナレッジグラフ構築に用いた「データ」

制御プロンプトを作成するための小説

The Speckled Band

ナレッジグラフから「各シーンを説明したテキスト」とそれに対応した「主語、述語、目的語」を抽出し利用した。「各シーンを説明したテキスト」をLLMへの入力として使い、「主語、述語、目的語」を答えとして利用した。但し目的語は「各シーンを説明したテキスト」と直接関係のある['kgc:hasProperty','kgc:to', 'kgc:what', 'kgc:whom', 'kgc:why', 'kgc:where', 'kgc:how','kgc:opposite', 'kgc:on']の合計9種類だけを選んで使った。これらのデータはダウンロード後JSON形式に加工した。このファイルはgoogledriveに「SpeckledBand.json」として共有する。

制御プロンプトはテキストファイルで84個用意した。半分の42個は日本語、残りの半分は英語とした。英語のプロンプトは日本語のプロンプトをgoogle翻訳したものである。プロンプトのファイル名には番号が振られているがこの番号をIDとして参照できるようにしている。制御プロンプト内に「~WORD~」を記述する事でプログラムで入力文章に置換できるようにした。このプロンプトはgoogledriveに圧縮ファイルとして共有する。ファイル名は「prompt.tar.gz」である。

- ナレッジグラフの構築手法の説明

  llama2によって文章を主語、述語、目的語に分割する手法を提案する。本研究では試作したプロンプトをllama2に与え、出力されたトリプルを評価しプロンプトの改良を繰り返す事で文章分割に最適なプロンプトを見つけ出そうとした。llama2による文章分割の手順を図1に示す。処理の制御を行う制御プロンプトは人間が考える。「~WORD~」の文字列を記述する事で、プロンプトにより分割する文章を自動で埋め込む。このプロンプトをllama2に入力するが、この時にループ処理を使い複数の文章、複数の制御プロンプトの組み合わせを自動で入力できるようにした。出力されたトリプルはJSON形式に限定しテキストファイルに保存した。この時に番号を振る事でIDとして参照し易くした。

  llama2により出力されたトリプルをBERTによる評価を行う。この手順を図2に示す。まず、出力されたテキストファイルにはJSON形式のトリプルの他に関係の無い文章が記述される事が殆どである。そのためプログラムにより先頭のJSON記述だけを抽出する。この時、カンマの記述が間違っている事があるのでプログラムによる修正も行う。JSON形式で読み込めたトリプルは正解データと合わせてBERTにより類似度を求める。単語ごとに類似度を求め、全ての単語で類似度が0.82以上の場合を正解として出力する。また、ループ処理により指定した量の評価が可能である。正誤判定の結果はcsvファイルとして出力する。このファイルにはどのプロンプトの結果か判別できるようにIDを付けている。IDは「制御プロンプト - 入力文章 - 回数」と記述されている。

  これらのツールを使い、制御プロンプトを54個、入力文章20個の組み合わせを5回ずつ、合計5400パターンのプロンプトと制御プロンプト30個、入力文章99個の組み合わせを1回ずつの合計2970パターンで推論を行った。

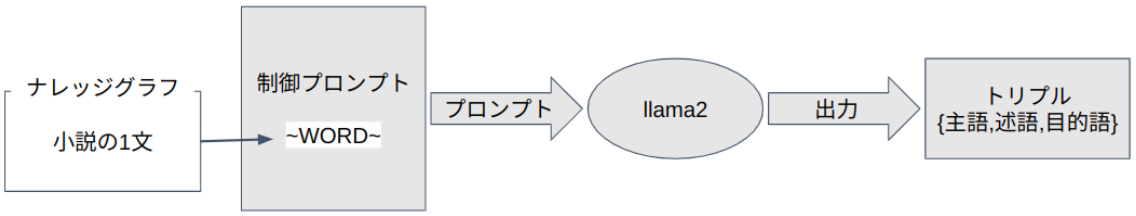

図1 llama2による文章分割

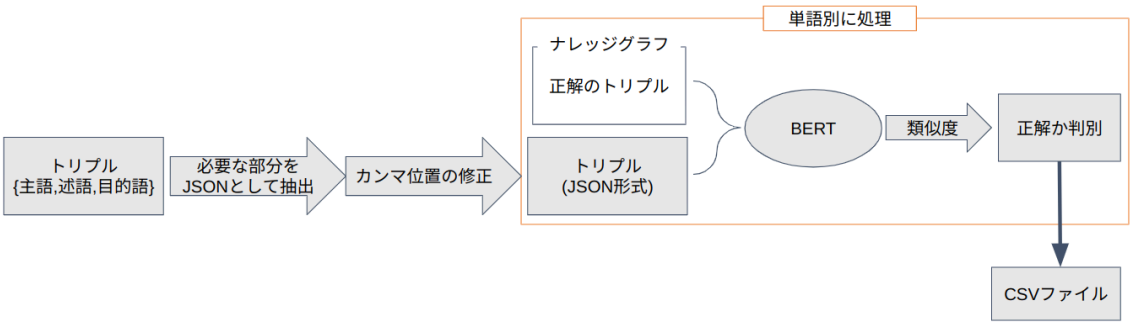

図2 BERTによる評価

- パフォーマンス情報
  OS:Ubuntu 22.04.3 LTS x86_64
  CPU:AMD Ryzen Threadripper PRO 5975WX s (64) @ 3.600GHz
  GPU:NVIDIA RTX A6000
  GPU:NVIDIA RTX A6000
  Memory: 3451MiB / 257565MiB

- 参考情報
  - 公開されている推理小説ナレッジグラフ
  - How to Convert Any Text Into a Graph of Concepts, https://towardsdatascience.com/how-to-convert-any-text-into-a-graph-of-concepts-110844f22a1a
  - GraphGPT, https://github.com/varunshenoy/GraphGPT

5. 構築したナレッジグラフの評価

評価方法は構築手法でも説明した通り、llama2で出力されたJSON形式のトリプルと正解データをBERTで類似度による比較を行い正解かを評価した。これを全てのプロンプトで行った結果をCSVファイルで出力し、さらに正解率等をまとめた結果をgoogle driveにて「正解率」のgoogle spreadsheetとして公開する。表では、1が正解を表し、空白は間違いを表し、「*」はJSON形式として読み込めない事を表す。また、正解率の高かった上位3つの結果を表1に示す。正解率は22.2%とかなり低い。内訳を見ると主語が70.7%と高いが、述語が39.4%、目的語が34.3%と低い。目的語は更に9種類に分類を行っているので正しい項目に分類できない事も正解率が低い要因の1つと考えられる。表3の結果よりJSONとして

読み込めない確率は5%前後の為、少なくとも指定したフォーマットでの回答は行えると考える。一番正解率の高いプロンプト67を以下に示す。2,3番目に高いプロンプト41,57との違いは例題が3つある事、入力文章を記述する箇所には「Actual performance (本番)」と記述している事である。例題が多い方が出力の法則が固定化され正解率が高くなるのではないかと考えられる。また入力文章の手前に「本番」と書く事で例題と入力文章の差別化ができていると感じる。

これらの結果からプロンプトだけでも単語の分割は可能であるが、主に述語と目的語への分類分けは難しいと考えられる。原因として単語の分割と分類の2つの作業を1回の推論で行っているからだと考える。これを踏まえ今後はllama2に1回で出力を行うのでは無く、複数回に別けて推論する事で出力の精度が上がらないかと考えている。

表1 正解率の高いプロンプト結果

プロンプトID	正解率(全体)	正解率(主語)	正解率(述語)	正解率(目的語)
67	22.2%	70.7%	39.4%	34.3%
41	21.0%	64.0%	23.0%	29.0%
57	20.2%	71.7%	42.4%	33.3%

表3 テキストファイルからJSON読み込み失敗率

プロンプトID	読み込み失敗率
67	7.1%
41	2.0%
57	6.1%

プロンプト67

```
Unset
Separate the input sentence into subject, predicate, and object. The output
should be in JSON format.
   'hasProperty',
   'to',
   'what',
   'when',
   'whom',
   'why',
   'where',
   'how',
   'opposite',
   'on'

example)
###Input text
```

Helen comes to Holmes' house
###output
```
{
   "subject":"Helen",
   "hasPredicate":"arrive",
   "hasProperty":"xx",
   "to":"Holmes' house",
   "what":"xx",
   "whom":"xx",
   "why":"xx",
   "where":"xx",
   "how":"xx",
   "opposite":"xx",
   "on":"xx"
}
```
Example 2)
input text
Holmes and Watson think it is unnatural that there is a ventilation hole (a small hole) between the rooms.
output
```
{
 "subject" : "Holmes",
 "subject" : "Watson",
 "subject" : "Watson",
 "hasPredicate" : "doubt",
 "hasProperty":"xx",
 "to":"xx,
 "what" : "vent",
 "whom":"xx",
 "why":"xx",
 "where":"xx",
 "how":"xx",
 "opposite":"xx",
 "on":"xx"
}
}
```
Example 3)
input text
There's a leopard in the mansion
output
```
{
 "subject" : "leopard",
 "hasPredicate" : "exists",
 "hasProperty":"xx",
 "to":"xx,
 "what" : "xx",
 "whom":"xx",
 "why":"xx",
 "where" : "Roilott's mansion",
 "where": "home",
 "where": "mansion",
```

```
 "where" : "land of the mansion",
 "how":"xx",
 "opposite":"xx",
 "on":"xx"
}

Actual performance)
###Input text
~WORD~
###output
```

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）
　構築に用いたプログラム及びデータは以下のgoogle driveに公開する。
https://drive.google.com/drive/folders/1EIB_bQccJQe07uLeXFg3w5DRm7SiZO1I
　llama.cpp.pyがllama2の実行プログラムである。answer.pyとbert.pyが評価のプログラムである。注意としてllama2やBERTはローカルモデルの為、予めモデルをダウンロードする必要がある。特にllama2はllama.cppにより量子化されたモデルを使用している。

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：
　（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（※）：