Homework 3

Canonical Knowledge problems

AutoGrader result

```
Starting on 4-15 at 15:06:41

Question q1
===========

*** PASS: test_cases\q1\correctSentence1.test
***     PASS
*** PASS: test_cases\q1\correctSentence2.test
***     PASS
*** PASS: test_cases\q1\correctSentence3.test
***     PASS
*** PASS: test_cases\q1\entails.test
***     PASS
*** PASS: test_cases\q1\entailsLong.test
***     PASS
*** PASS: test_cases\q1\findModelSentence1.test
***     PASS
*** PASS: test_cases\q1\findModelSentence2.test
***     PASS
*** PASS: test_cases\q1\findModelSentence3.test
***     PASS
a.__dict__ is: {'op': 'A', 'args': ()}
*** PASS: test_cases\q1\findModelUnderstandingCheck.test
***     PASS
*** PASS: test_cases\q1\plTrueInverse.test
***     PASS

### Question q1: 10/10 ###


Question q2
===========

*** PASS: test_cases\q2\atLeastOne.test
***     PASS
*** PASS: test_cases\q2\atLeastOneCNF.test
***     PASS
*** PASS: test_cases\q2\atLeastOneEff.test
***     PASS
*** PASS: test_cases\q2\atMostOne.test
***     PASS
*** PASS: test_cases\q2\atMostOneCNF.test
***     PASS
*** PASS: test_cases\q2\atMostOneEff.test
***     PASS
*** PASS: test_cases\q2\exactlyOne.test
***     PASS
*** PASS: test_cases\q2\exactlyOneCNF.test
***     PASS
*** PASS: test_cases\q2\exactlyOneEff.test
```

```
***     PASS

### Question q2: 10/10 ###


Question q3
===========

*** Testing checkLocationSatisfiability
[LogicAgent] using problem type LocMapProblem
Ending game
Average Score: 0.0
Scores:        0.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** PASS: test_cases\q3\location_satisfiability1.test
*** Testing checkLocationSatisfiability
[LogicAgent] using problem type LocMapProblem
Ending game
Average Score: 0.0
Scores:        0.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** PASS: test_cases\q3\location_satisfiability2.test
*** Testing pacphysicsAxioms
*** PASS: test_cases\q3\pacphysics1.test
*** Testing pacphysicsAxioms
*** PASS: test_cases\q3\pacphysics2.test
*** PASS: test_cases\q3\pacphysics_transition.test
***     PASS

### Question q3: 10/10 ###


Question q4
===========

[LogicAgent] using problem type PositionPlanningProblem
Path found with total cost of 3 in 0.0 seconds
Nodes expanded: 0
Pacman emerges victorious! Score: 508
Average Score: 508.0
Scores:        508.0
Win Rate:      1/1 (1.00)
Record:        Win
*** PASS: test_cases\q4\positionLogicPlan1.test
***     pacman layout:          maze2x2
***     solution score:         508
***     solution path:          West South
[LogicAgent] using problem type PositionPlanningProblem
Path found with total cost of 999999 in 0.2 seconds
Nodes expanded: 0
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win
*** PASS: test_cases\q4\positionLogicPlan2.test
```

```
***     pacman layout:          tinyMaze
***     solution score:         502
***     solution path:          South South West South West West South West
[LogicAgent] using problem type PositionPlanningProblem
Path found with total cost of 999999 in 31.8 seconds
Nodes expanded: 0
Pacman emerges victorious! Score: 491
Average Score: 491.0
Scores:        491.0
Win Rate:      1/1 (1.00)
Record:        Win
*** PASS: test_cases\q4\positionLogicPlan3.test
***     pacman layout:          smallMaze
***     solution score:         491
***     solution path:          East East South South West South South West West South West West

### Question q4: 10/10 ###

Question q5
===========

[LogicAgent] using problem type FoodPlanningProblem
Path found with total cost of 9 in 0.1 seconds
Nodes expanded: 0
Pacman emerges victorious! Score: 513
Average Score: 513.0
Scores:        513.0
Win Rate:      1/1 (1.00)
Record:        Win
*** PASS: test_cases\q5\foodLogicPlan1.test
***     pacman layout:          testSearch
***     solution score:         513
***     solution path:          West East East South South West West East
[LogicAgent] using problem type FoodPlanningProblem
Path found with total cost of 999999 in 7.6 seconds
Nodes expanded: 0
Pacman emerges victorious! Score: 573
Average Score: 573.0
Scores:        573.0
Win Rate:      1/1 (1.00)
Record:        Win
*** PASS: test_cases\q5\foodLogicPlan2.test
***     pacman layout:          tinySearch
***     solution score:         573
***     solution path:          South South West East East East East North North North North West

### Question q5: 10/10 ###


Finished at 15:07:22

Provisional grades
==================
Question q1: 10/10
Question q2: 10/10
Question q3: 10/10
Question q4: 10/10
```

```
Question q5: 10/10
------------------
Total: 50/50

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

Q1. Logic Warm-up

- sentence 1:

```
# 透過 conjoin 及 disjoin 及相關 operator 組成 expression
conjoin(A | B, (~A % (~B | C)), disjoin(~A, ~B, C))
```

- sentence 2:

```
# 透過 conjoin 及相關 operator 組成 expression
conjoin(C % (B | D), A >> (~B & ~D), (~(B & ~C) >> A), (~D >> C))
```

- sentence 3:

  依據描述提供相關 expression

  1. Pacman is alive at time 1 if and only if Pacman was alive at time 0 and it was not killed at time 0 or it was not alive at time 0 and it was born at time 0.

     ```
     PacmanAlive_1 % ((PacmanAlive_0 & ~PacmanKilled_0) | (~PacmanAlive_0 & PacmanBorn_0))
     ```

  2. Pacman cannot both be alive at time 0 and be born at time 0.

     ```
     ~(PacmanAlive_0 & PacmanBorn_0)
     ```

  3. Pacman is born at time 0.

     ```
     PacmanBorn_0
     ```

  4. 透過 conjoin 結合，完整 expression 如下

     ```
     # PacmanAlive_0: 表示在時間 t=0 時，Pacman 是否活著
     # PacmanAlive_1: 表示在時間 t=1 時，Pacman 是否活著
     # PacmanKilled_0: 表示在時間 t=0 到 t=1 期間，Pacman 是否被殺死
     # PacmanBorn_0: 表示在時間 t=0 到 t=1 期間，Pacman 是否出生

     conjoin([PacmanAlive_1 % ((PacmanAlive_0 & ~PacmanKilled_0) |
     (~PacmanAlive_0 & PacmanBorn_0)),
     ~(PacmanAlive_0 & PacmanBorn_0), PacmanBorn_0])
     ```

- findModelUnderstandingCheck:

  透過一個 dummyClass 來進行 expression 中的字串符號以及變量的處理，確保在 expression 中能夠正確處理

```
# code snippet
class dummyClass:
    def __init__(self, variable_name: str = 'A'):
        self.variable_name = variable_name
    def __repr__(self):
        return self.variable_name
    return {dummyClass('a'): True}
```

- entail:

  主要判斷 premise (前提) 跟 conslusion (結論) 是否為蘊含 (entailment) 關係。這裡透過 conjoin 的方式判斷 premise 跟 ~conslusion 的關係是否為 entail (return True) or not (return False)

  ```
  # code snippet
  findModel(conjoin(premise, ~conclusion)) is False
  ```

- plTrueInverse:
  - 取 inverse_statement 的否定 ⇒ ~inverse_statement
  - 透過 to_cnf 轉換成 conjuctive normal form
  - 透過 pl_true 判斷 statement 及 assignment 真假

  ```
  # code snippet
  pl_true(to_cnf(~inverse_statement), assignments)
  ```

Q2. Logic Workout

- atLeastOne: 即 disjoin literals
- atMostOne:
  - 透過 combiniations 取得 literals 的組合，例如: [x1, x2, x3] ⇒ (x1, x2), (x1, x3), (x2, x3)
  - 透過 for loop 針對每個 pair 生成 expression，判斷 ~a | ~b，例如: ~x1 | ~x2, ~x1 | ~x3, ~x2 | ~x3
  - conjoin 所有的結果: (~x1 | ~x2) & (~x1 | ~x3) & (~x2 | ~x3)
  - 透過以上可取得最多一個為真的狀況
- exactlyOne:
  - conjoin(atLeastOne, atMostOne)
  - 原因：atLeastOne: 確保至少有一個變數是真的，atMostOne: 確保最多只有一個變數是真的。結合後即為剛好只有一個，即 exactlyOne

Q3. Pacphysics and Satifiability

- pacmanSuccessorAxiomSingle:
  - 表示 pacman 從 t-1 to t 的可能移動 expression。
  - PropSymbolExpr(pacman_str, x, y, time=now): 定義 pacman 在 now 的時間點於 (x, y) 的 expression
  - disjoin(possible_causes): 列出所有可能的先前位置
  - PropSymbolExpr(pacman_str, x, y, time=now) % disjoin(possible_causes): 表示 if and only if 的關係，表示 pacman 的目前位置是基於所有 possible causes 的先前位置而來
- pacphysicsAxioms
  - 按照題目中的提示步驟進行實作

  ```
  # for all (x, y) in all_coords:
  #     If a wall is at (x, y) --> Pacman is not at (x, y)
  for x, y in all_coords:
      wall_x_y = PropSymbolExpr(wall_str, x, y)
      pacman_x_y_t = PropSymbolExpr(pacman_str, x, y, time=t)
      pacphysics_sentences.append(wall_x_y >> ~pacman_x_y_t)

  # Pacman is at exactly one of the squares at timestep t.
  pacphysics_sentences.append(
      exactlyOne(
          [PropSymbolExpr(pacman_str, x, y, time=t) for x, y in non_outer_wall_coords]))
  ```

```
# Pacman takes exactly one action at timestep t.
pacphysics_sentences.append(
    exactlyOne([PropSymbolExpr(direction, time=t) for direction in DIRECTIONS]))

# Results of calling sensorModel(...), unless None.
if sensorModel is not None:
    pacphysics_sentences.append(sensorModel(t, non_outer_wall_coords))

# Results of calling successorAxioms(...),
# describing how Pacman can end in various locations on this time step.
# Consider edge cases. Don't call if None.
if t > 0 and successorAxioms is not None:
    pacphysics_sentences.append(successorAxioms(t, walls_grid, non_outer_wall_coords))
```

- checkLocationSatisfiability

  判斷 pacman 在 t = 1 的時候在 x1, y1 的可能性與否，演算法主要拆成兩部分

  1. **初始化知識庫 (KB)**
     - 將 pacman 在 t=0 和 t=1 的 pacphysicsAxioms 添加到知識庫。
     - 添加表示 pacman 在特定時間和位置的 PropSymbolExpr。
  2. **模型生成**:
     - findModel 函數用於尋找一個模型，使得與知識庫中的所有規則一致且 pacman 位於指定位置 (x1, y1) 或不在該位置的情況下為真。
     - model_pacman_at_x1_y1 模型表明在 (x1, y1) 位置的可能性。
     - model_pacman_not_at_x1_y1 模型表明不在 (x1, y1) 位置的可能性。

  回傳兩個 model，分別表示在 x1, y1 的可能及不在的可能

Q4. Path Planning with Logic

計算 pacman 的可能路徑

1. 初始化知識庫 (KB)
2. 使用一個無窮迴圈來模擬每一個時間步 t 的狀態
3. 透過 exactlyOne 確保每個時間點的唯一行動和位置
4. 當 t ≠ 0，將所有非牆位置在 KB 中記錄 pacmanSuccessorAxiomSingle
5. 檢查是不是已經完成 goal，若完成則 return 目標位置路徑，否則 t += 1，繼續下一個時間點的 loop

Q5. Eating all the Food

1. 初始化 KB
2. 使用一個無窮迴圈來模擬每一個時間步 t 的狀態
3. 將所有非牆位置紀錄至 KB
4. 透過 findModel 判斷是否有找到一個模型表示 foodList 是否全部都被吃掉作為終止 loop 條件。
5. 若沒有找到則 t += 1，繼續下一個 loop 的 iteration

Modern knowledge problems

Q1.

人工通用智能（AGI）在 AWS 上的定義是一個理論性的 AI 研究領域，旨在創建具有類似人類智慧和自學能力的軟體。這種軟體的研發目標，是使其能夠執行那些它可能未必接受過相關訓練或專門開發來完成的任務。

從我自身專注於自然語言處理（NLP）的研究領域來看，不論是過去 Transformer-based 的 Language model（如 BERT-based model），還是從前年開始逐漸興起的 GPT 3 至今日的 GPT 4。這些類型的語言模型雖然已經具備了對上下文（contextual）的理解能力，但它們仍然高度依賴訓練資料集。無論是基於 BERT 的模型還是生成模型，儘管它們的架構存在差異，但最大的區別仍在於

數據的獲取和收集方面，目標是通過更大量的數據訓練出更加完善的模型。然而，模型始終存在邊界。例如，國科會的台德計畫，旨在將語言模型本地化，使用繁體中文的本地化數據進行訓練，這表明數據在很大程度上影響了模型的泛化和理解能力。

以下是一些在發展 AGI 的領域中可能碰到的瓶頸與挑戰：

1. 理解與生成方式：LLM 的理解能力是通過上下文的來回閱讀（embedding）來實現的，而在文本生成時，它採用機率方式輸出下一個最有可能的詞；然而 AGI 追求的理想可能包括真正的 “理解”、“自我意識” 和 “情感”，這些是目前LLM所無法達成的。
2. 幻覺問題：目前 LLM 在回應過程中仍可能產生幻覺，雖然像 RAG 等研究已經能夠在一定程度上降低這一問題，但要讓 LLM 解決幻覺問題，以滿足 AGI 所需解決的挑戰，仍然存在空間。
3. 跨領域泛化能力：AGI 的目標是在各種領域中都能提供更通用的解決方案和適用性。然而，儘管LLM 目前已在收集數據，推出了許多不同數據量級的模型，例如 meta 的 llama-2 7b、13b、70b等，但在不同領域和文化中獲取文本數據，或是獲取用於訓練的影像或語音數據（例如 YouTube已經發布聲明不允許使用影音數據進行訓練）的困難，仍然是存在的瓶頸。
4. 倫理及社會：AGI 的目標是解決包括人類情感或認知任務在內的通用問題。然而，正如之前提到的，生成型 AI 無法真正理解情感等狀況，因此可能會在各種面向上引起衝突（隱私、安全性、文化性等），這需要被謹慎防範。

Q2.

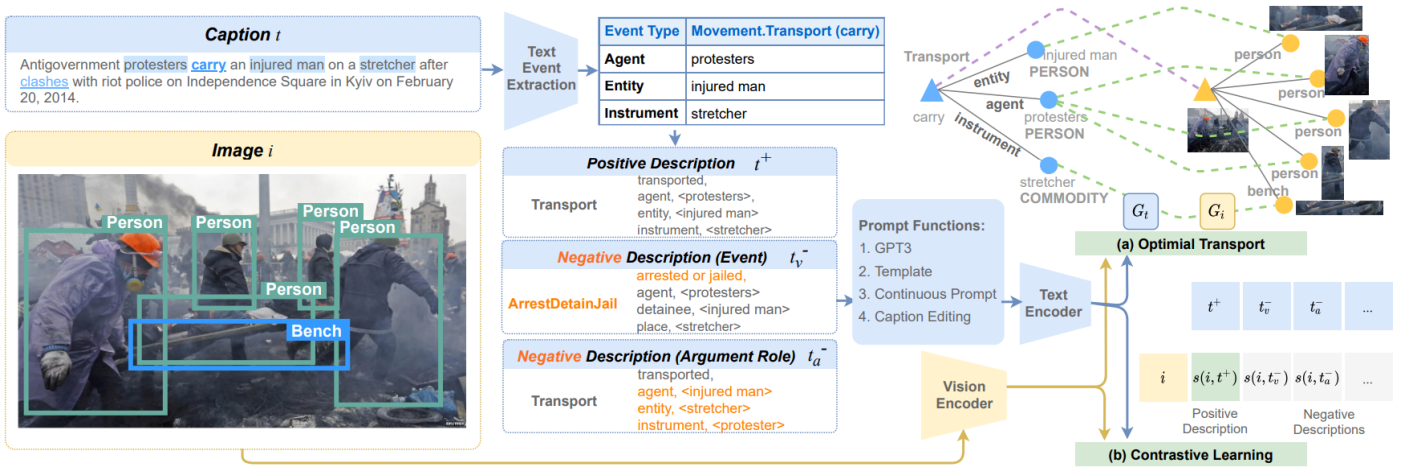


Figure 2. Architecture of CLIP-Event. We take advantage of event structural knowledge in captions to contrast hard negatives about event types and argument roles (in blue), which is then used to supervise image event understanding (in yellow) as a cross-media transfer of event knowledge. The negative event structures are highlighted in orange. The events and objects are from automatic system output.

Constrastive Learning 的概念主要是透過正負樣本的蒐集進行訓練，在訓練過程中，期望透過各種相似度比對（最簡單的例如 cosine similarity），將相近的 embedding 在空間中距離拉近，並將 negative sample 距離拉遠。從 CLIP 的架構圖可知此篇 paper 的整體流程，其中與 constrastive 相關的步驟如下：

1. Event structure extraction and image recognition

   首先針對 Caption 進行資訊擷取（Information extraction，類似 Named entity recognition 提取出相關實體），並且轉化為 Event structure。並且同時針對 Image 透過 objective detection 的方式擷取出相關圖形物件。

2. 正負樣本取得

   - Positive description: 直接從圖像及擷取出的文字產生相關描述
   - Negative description: 透過 event structure 的參數更換進行處理
     - Event 類型的更換：更改 event 的類型來建立 negative sample，例如將 transport 變更為 arrested or jailed。
     - Argument role 更換：通過更改或調動參數，例如調換 agent 為 injured man，entity 改為 stretcher，instrument 更改為 protester。

3. Contrastive learning

   透過上述所 collect 的 positive / negative samples 透過 vision encode 及 text encoder 轉換為 embedding，並藉由 cosine similarity 計算相似度以進行文字與圖形間的 alignment。在 alignment 中，此 paper 導入一種叫做 optimal transport 強化 alignment。

Q3.

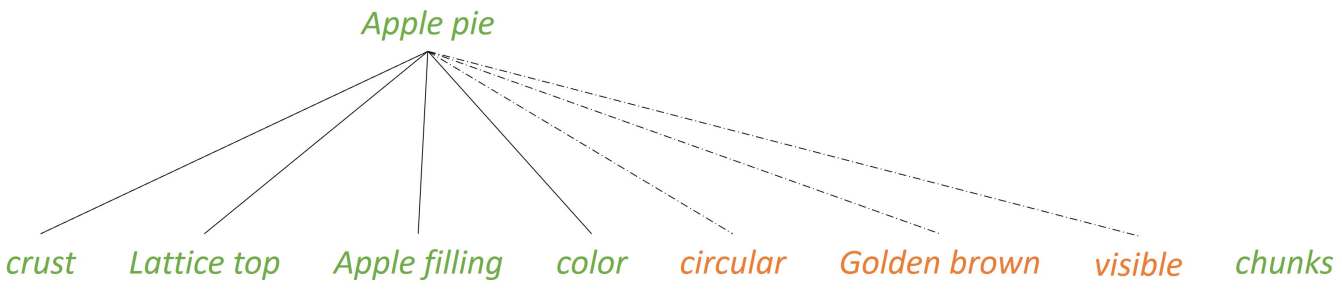
這篇 paper 只要透過結構化資訊來嘗試解決一般透過類別名稱及描述作為 context prompting input，無法有效表示特定類別以及屬性間的關係。HPT 透過定義並結構化相關實體以及屬性間的關聯來提升 VLM 的 performance。結構化的步驟如下：

1. 透過人工預先定義的 instruction T 讓 LLM 生成針對類別的相關描述，例如：”What does [water lily] look like among all [types of flowers]?”。LLM 將根據 T 生成相關關鍵描述。

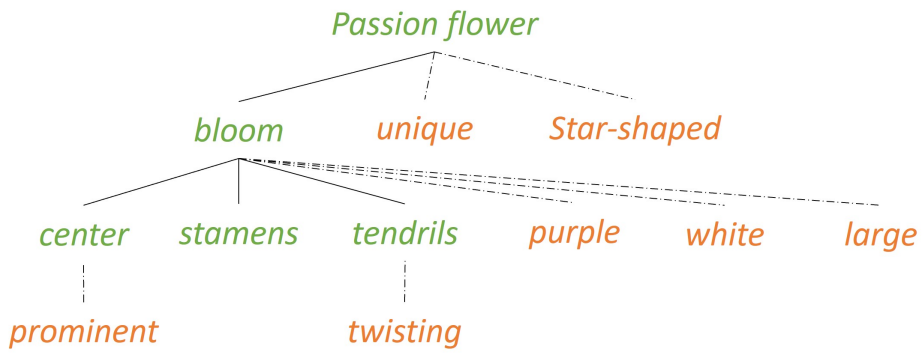
2. 將上述的生成結果透過 T′ 轉換成 structure information，包含 entity、attribute、relation 等
3. 透過將 entity, attribute 及 relation 藉由 triplet 建立 structured graph (類似 knowledge graph)。

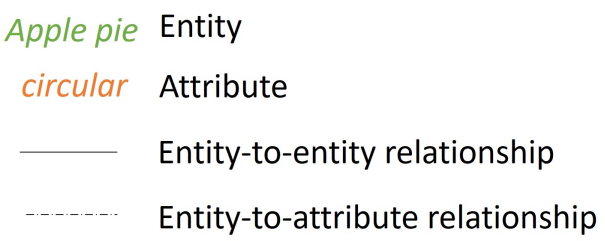
Q4.

Description:
Apple pie is recognizable by its circular crust, lattice top, golden brown color, and visible chunks of apple filling.



Description:
The passion flower has a unique star-shaped purple and white bloom with a large center, prominent stamens, and twisting tendrils.





Q5.

除了透過結構化的資訊定義類別 (entity) 本身的 attribute 及 relation 外，仍有其他資訊可以試著強化 VLM 的生成能力。

- sentiment analysis：在欣賞畫作時，通常會同時感受到繪圖者的情緒，目前 VLM 或許能臨摹各種不同世代的畫風，但尚未能將情感融入進行理解。
- 基礎常識理解：當 VLM 接收到水流出來了的指令，對於人類而言可以清楚之類因為地心引力的作用，常態性在不特別說明下水應會往低處流，如何讓 VLM 可以更好的理解基礎常識，以避免導致影像上的幻覺或是錯誤。
- 時序關係：若希望繪製下雨的街道，對於 VLM 來說，如何理解：下雨 → 地濕等則是對於 VLM 來說的時序理解問題。