



國立臺灣大學
National Taiwan University

ADL 2021 Final Project

<Team 15>

組員：黃泰誠、李育倫、王威凱、侯冠哲

指導老師：陳縉儂 教授

目錄

1 Abstract.....	4
2 Introduction.....	5
2.1 Schema-Guided Dialogue	5
2.1.1 The Schema-Guided Dialogue Dataset.....	5
2.1.2 Services and APIs.....	6
2.1.3 Dialogue Simulator Framework	6
2.1.4 Dialogue Paraphrasing	7
2.1.5 Dataset Analysis	7
2.2 SimpleTod-DST.....	8
2.3 BERT-DST.....	9
2.3.1 Dialogue Context Encoding Module	9
2.3.2 Dialogue State Update Mechanism.....	9
2.3.3 Parameter Sharing.....	10
2.3.4 Slot Value Dropout	10
2.4 Add Chit-Chat to Enhance TOD	10
2.4.1 Data Construction.....	11
2.4.2 Candidate Generation	11
2.4.3 Candidate Filtering.....	12
2.4.4 Annotation	12
3 Approaches	13
3.1 The Schema-Guided	13
3.2 SimpleTOD	13
3.2.1 Task-Oriented Dialogue	13
3.2.2 Causal Language Modeling.....	14
3.2.3 Architecture.....	14
3.2.4 Training Details & Action Generation.....	15
3.3 SimpleTOD add chit-chat.....	15
3.3.1 Task Formulations	15
3.3.2 Models	15
4 Experiments.....	17
4.1 DST.....	17
4.1.1 Data preprocess.....	17
4.1.2 Data Embedding.....	19
4.1.3 Pretrain model.....	21
4.1.4 Hyperparameter.....	23
4.1.5 各參數介紹	23

4.1.6 MutiWOZ → SGD.....	24
4.1.7 任務整體流程圖	25
4.1.8 環境設定.....	26
4.1.9 輸出檔案資料處理	27
5. Conclusion	31

1 Abstract

這次的期末任務分成 DST(Dialog State Tracking Challenge)及 NLG(Natural Language Generation)兩個部分。在第一個任務中所著重的部分為預測用戶所要表達的企圖並生成與上下文相關的句子。而第二個任務是將 chit-chat 加到第一個任務所生成對話中來添加整個對話的互動性及擬真性。為了達成上述的目標且讓 chit-chat 的所產生的句子更具多樣性，chit-chat 中的 annotations 是來自兩種 **task-oriented datasets** 的數據集(**Schema-Guided Dialogue and MultiWOZ 2.2**)。由於助教給我們的 dataset 為 SGD 的形式，因此後面章節會特別說明此種數據集的特色，而根據我們所查到的資料，處理 **Task Oriented Dialogues** 的主要 model 有 SimpleTOD-DST 及 BERT-DST 兩種，這兩種 model 都是 seq2seq 的架構。以下分別簡略介紹這兩種 model 的特點。

1. SimpleTOD-DST: 訓練一種單一且具因果關係的 model 來去應付基於句子序列所產生的子任務，通常都是利用有 pre-trained, open domain 及 causal language 特點的 model 來做訓練(ex: GPT-2、DistilGPT2)。其中又可細分出三種子模型來去將 chit-chat 加到 task-oriented dialogues 中，此架構可在 task 和 chit-chat 任務之間進行 codeswitch 的動作，來使句子更有趣味性且人性化。
2. BERT-DST: 特色是聚焦於解決當 state tracker 不知道 ontology 或在訓練時不知道 slot-value 的情況下，透過搜尋對話中的詞句來找出未知的 slot-value，在這個架構中的 state tracker 是採 end-to-end 的形式，它可以直接從對話中提取 slot-value。此外，在 BERT 中的 encoder 的參數是共享的，這麼做有 2 個優點。
 - (1) 在 ontology 中的參數不會呈線性增加。
 - (2) 生成語句的資訊可透過 slot 來進行傳遞。

2 Introduction

2.1 Schema-Guided Dialogue

2.1.1 The Schema-Guided Dialogue Dataset

此種資料集是由 16 個領域中 26 個服務裡頭超過 16000 句對話所構成 (圖 1)。下面會展示用於 **task oriented dialogue** 中模式指引的範例，預測出來的 intents 和 slots 被當作是輸入且存在動態的集合中，在與 **MultiWoz** 這個資料集做比較時可以發現 **MultiWoz** 所涵蓋的 domain 數量沒有 **SGD** 多。此外，它為每個 domain 定義了一個靜態的 API，然而多重的 services 裡頭有著重疊的功能性但有著異構的區域才是比較符合現實情況的。

Metric ↓ Dataset →	DSTC2	WOZ2.0	FRAMES	M2M	MultiWOZ	SGD
No. of domains	1	1	3	2	7	16
No. of dialogues	1,612	600	1,369	1,500	8,438	16,142
Total no. of turns	23,354	4,472	19,986	14,796	113,556	329,964
Avg. turns per dialogue	14.49	7.45	14.60	9.86	13.46	20.44
Avg. tokens per turn	8.54	11.24	12.60	8.24	13.13	9.75
Total unique tokens	986	2,142	12,043	1,008	23,689	30,352
No. of slots	8	4	61	13	24	214
No. of slot values	212	99	3,871	138	4,510	14,139

(圖 1)

這裡簡單舉例一個用於 **task-oriented dialogue** 中 schema-guided 的範例。透過為所有服務和 API 建立一個對話模型，使用 service 的 schema 作為輸入，對話模型就可利用存於 schema 中的動態集合(intents、slots)進行預測。這種設定可以在跨 API 且具相似語意中的 service 進行有效的知識共享，這樣就可以處理那些看不到的 services 和 APIS。

在 eval set 中有 services 是不存在 training set 中的，這樣就可以評估 model 應用在看不到 services 的情況下表現如何。下圖可以看到 20 個 domains 分別隸屬於 train, eval, test 檔案中，且在這些 domains 上建立了 45 個 services。Simulator framework 會與 services 做溝通並生成出對話的大綱。接著利用 crowd-sourcing procedure 將大綱轉成自然語言，crowd-sourcing procedure 還會將從 simulator 中所獲得的所有 annotation 給存下來，這樣就省去在蒐集對話時還要收集 annotation 的這個步驟。

2.1.2 Services and APIs

在 **Schema-Guided Dialogue Dataset** 中是將 schema 視為 intent 及 slot 的組合(圖 2)且有附加的約束，還有值得注意的是所有 service 都是透過 SQL engine 來去實作出來的。接下來我要說明附加約束(domain-specific constraints)是甚麼，在數據集中專門將此種類型的 slots 視為無分類項，這裡頭的 slots 是沒有給 values 的，另外在 eval set 中有部分的 slots-values 不存在於 train set 中以確保存在 new values 的情況下去評估 model。

service_name: "Payment" description: "Digital wallet to make and request payments"	Service
name: "account_type" categorical: True description: "Source of money to make payment" possible_values: ["in-app balance", "debit card", "bank"]	Slots
name: "amount" categorical: False description: "Amount of money to transfer or request"	
name: "contact_name" categorical: False description: "Name of contact for transaction"	
name: "MakePayment" description: "Send money to your contact" required_slots: ["amount", "contact_name"] optional_slots: ["account_type" = "in-app balance"]	Intents
name: "RequestPayment" description: "Request money from a contact" required_slots: ["amount", "contact_name"]	

Figure 1: Example schema for a digital wallet service.

(圖 2)

2.1.3 Dialogue Simulator Framework

此部分是由兩個 system 組成分別是 user system 和 agent system(圖 3)，這兩個 system 透過由 dialogs acts 所組成的有限 actions 的集合來做互動，這些 dialogs acts 可以採用 slot-value 作為它的參數，然而 user 和 system 萃取參數的作法上還是有些許的不同，下面就來說明相異之處。

- (1) User agent: 可透過 service schema 或 SQL 後端來得到 dialog actions。
- (2) System agent: system 在獲得 dialog actions 上就有限制了，它只能在 service 裡頭的 slot 都有對應到 value 這種類型的 service 才可以得到 dialog actions，這麼做是為了確保 system 所產生的 dialog actions 是有效的。

當查完一個 intent 中的所有 dialog actions 後就可以查詢下一個 intent 的資訊了。另外，執行 domain-specific constraints 的編碼時是在 schema 及 scenario 中進行的，這樣就能使 simulator 應用於不同的 domains、services。

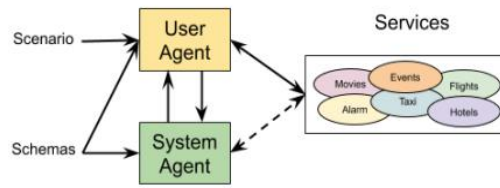


Figure 2: The overall architecture of the dialogue simulation framework for generating dialogue outlines.

(圖 3)

2.1.4 Dialogue Paraphrasing

此部分是將 simulator 生成的 outlines 轉換成自然語言。(圖 4)中 a 部分裡頭的 slot-value 都是由 service 取得的，b 部分則是隨機替換 slot-value，要注意的是在用戶回合中 slot-value 要保持一致。c 部分是將每個 dialog action 轉成自然語言並串接在一起。d 部分是找工作人員來確保所有對話的連貫性。在 paraphrasing 這個任務中，工作人員要確保釋義的 slot-value 要保持一致性及正確性，這不僅可以幫助驗證釋義的正確性，還可以通過字符串搜索自動獲取生成的話語中的 slot spans。

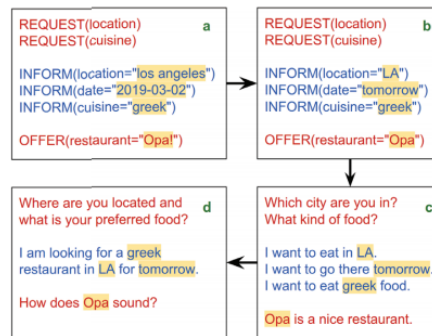
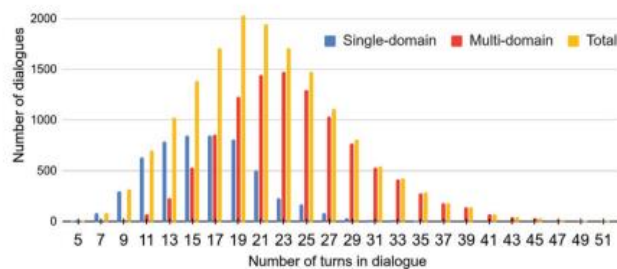


Figure 3: Steps for obtaining paraphrased conversations. To increase the presence of relative dates like tomorrow, next Monday, the current date is assumed to be March 1, 2019.

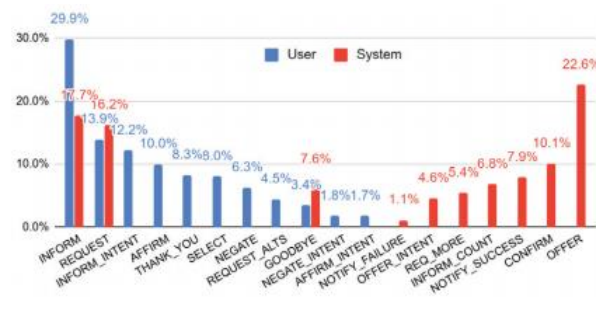
(圖 4)

2.1.5 Dataset Analysis

Annotation 包含了每個用戶對話中的 intent 和對話狀態以及每個系統對話中的系統動作。Schema 是由 APIs 中的語意訊息、intent 及 slot 所組成。(圖 5)標示了 single domain 和 multi domain 中對話回合數的分佈。(圖 6)則是顯示出在 dataset 中各個 dialog actions 所出現的頻率。



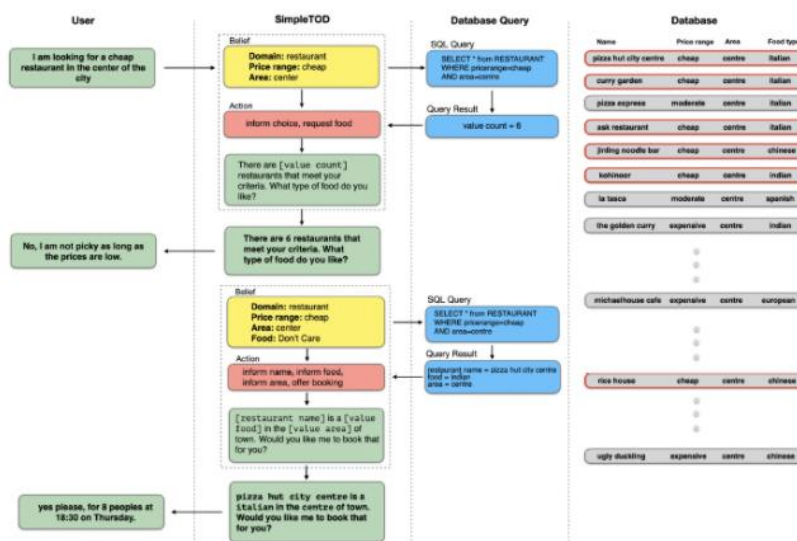
(圖 5)



(圖 6)

2.2 SimpleTod-DST

Task oriented dialogue (TOD)系統由 3 個部分組成，分別是 natural language understanding (NLU)、dialogue management (DM)及 natural language generation (NLG)。NLU 用來追蹤 belief state，DM 根據 belief state 來去決定要採取甚麼 actions，NLG 則是生成自然語言。SimpleTOD 還是第一個在 end-o-end 設置中同時實現對話狀態跟踪、動作決策和響應生成指標的模型。



(圖 7)

2.3 BERT-DST

上下文輸入由基於 BERT 的 encoding module 進行編碼，使上下文有 sentence-level and token-level 的資訊。sentence-level 的資訊用來進行分類，其類別為 3 個分別為 none、dontcare 及來自輸入的 span。Span prediction module 彙整 token-level 的資訊和 slot values 開始及結束的位置。

2.3.1 Dialogue Context Encoding Module

句子的編碼模組是基於 BERT 去訓練的，輸入的資訊分別放置了上一回合系統的對話及這一回合使用者的對話，第一個 token 也是用[CLS]做標記，而用[SEP]分開系統及使用者的對話。讓 $[x_0; x_1; \dots; x_n]$ 表示輸入標記序列。BERT 的輸入層將每個標記 x_i 嵌入到一個 e_i 中， e_i 又可看成 3 種嵌入的組合，其中 E_{tok} 為 wordpiece 的 embedding， E_{seg} 為第一句和第二句話的 embedding， E_{pos} 為第 i 句 token 位置的 embedding。將 embedding 的序列 $[e_1, e_2, e_3, \dots, e_n]$ 通過 BERT 的 bidirectional transformer encoder 變成 hidden state $[t_1, t_2, \dots, t_n]$ 。

$$\begin{aligned} \text{BERTinput}(x_i) &= E_{\text{tok}}(x_i) + E_{\text{seg}}(i) + E_{\text{pos}}(i) \\ &= e_i \in \mathbb{R}^d, \forall 0 \leq i \leq n \end{aligned}$$

$$\begin{aligned} [t_0, \dots, t_n] &= \text{BiTransformer}([e_0, \dots, e_n]) \\ t_i &\in \mathbb{R}^d, \forall 0 \leq i \leq n \end{aligned}$$

其中具有上下文資訊的 t_0 會被傳遞給分類模塊，而具有 token 資訊的 $[t_1, t_2, t_3, \dots, t_n]$ 則會被跨度預測模組來使用。在對話上下文編碼模組中的參數由 BERT 表示，從預訓練的 BERT 檢查點初始化，然後在 DST 數據集上進行微調。

2.3.2 Dialogue State Update Mechanism

在每一回合對話中，如果模型對每該回合預測的 slot 是 dontcare 或指定值（除了 none 之外的任何值），它將用於更新對話狀態。否則，slot 的對話狀態將保持與上一回合相同。

2.3.3 Parameter Sharing

在文獻中可以發現到上下文資訊可以在 slot 之間共享，其資訊是透過上下文編碼模組所產生的。這個發現可以讓所有 slot 中透過對話上下文編碼模組中應用參數共享的方式來減少 BERT 模型參數的數量。此外，它還可以在 slot 之間進行資訊的轉移，這可能有利於上下文關係的理解。在下面的章節中，將特定於 slot 的 BERT-DST 模型聯合架構稱為 BERT-DST SS，將具有編碼模組參數共享的 BERT-DST 模型稱為 BERT-DST PS。

2.3.4 Slot Value Dropout

Slot value dropout 是用來解決 model 在訓練時產生 overfitting 的狀況，overfitting 會使 model 產生 frequency slot 的現象而不是去注意上下文的含意。為了提高對 unseen slot 的穩健性，在訓練階段，以一定的機率使用一個特殊的 [UNK] 標記替換每個目標 slot-value。

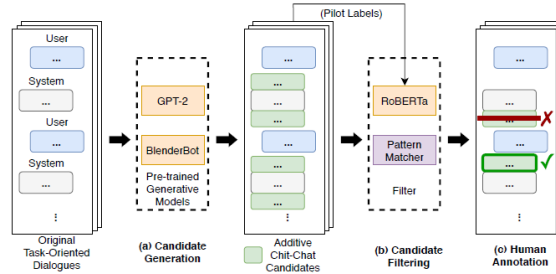
2.4 Add Chit-Chat to Enhance TOD

透過將 Chit-Chat 加到 Task-Oriented dialogues (ACCENTOR-Adding Chit-Chat to Enhance Task-Oriented dialogues) 中來整合兩種類型的系統，希望能讓虛擬助理不僅能夠執行各種複雜的任務，例如檢查天氣、預訂酒店或尋找餐廳，還包括上下文相關的閒聊。為了展示 ACCENTOR 的可行性並為後續研究收集數據，文獻中有提出了一種 Human/AI 協作數據構建的方法，該方法可以有效地將合適的閒聊句子添加到現有的 Task-Oriented dialogues 數據集中的系統響應的開頭或結尾。具體來說，首先使用現成的預訓練語言模型(SimpleTOD)和開放域聊天機器人生成用於增強的閒聊候選者。接下來，使用過濾模型自動過濾掉不太具有良好質量的候選對象。最後，利用人工 annotate 的方式將每個剩餘的候選者標記為好或壞，並附上理由。並使用所提出的方法擴充了模式引導對話 (SGD)和 MultiWOZ 2.1 語料庫。最後採用 ACUTE-Eval 來進行比較，參考的面向有參與度、趣味性、知識和人性。這部分有 3 個可以探討的面向。

- (1) Data augmentation: 用於為 Task-Oriented dialogues 任務生成不同的聊天數據，利用預先訓練的生成模型和自定義過濾器模型來減少人工 annotate。
- (2) Added chit-chat annotations to 23.8K dialogues。
- (3) Annotated dataset and models: 研究這些數據集和模型在 Task-Oriented dialogues 和 chit-chat 之間進行 code-switching。

2.4.1 Data Construction

(圖 8)為數據建構的總覽圖，數據構建概述：(a) 使用預訓練模型生成各種自由形式的閒聊候選對象，以增強 Task-Oriented dialogues 的任務，以及 (b) 使用自定義過濾器過濾掉不良候選對象以盡量減少註釋工作。(c) 人工標註註釋在上下文中並增強 chit-chat 的應用。



(圖 8)

2.4.2 Candidate Generation

給定一個 Task-Oriented dialogues $D = u_1; s_1; u_2; s_2; \dots$ 的對話集，其中 u_1, u_2, \dots, u_n 為使用者回合， s_1, s_2, \dots, s_n 為系統回合。將對話集丟入預訓練的模型中(ex: SimpleTOD or chit-chat chatbot)，並讓模型在最後一個 s_i 中加入一個 token 來當作這個對話集的結束，接著重複上述步驟來產生很多回合的對話集，再將前面兩邊的輸出視為 chit-chat 的候選者分別加入到系統對話的前面或後面。(圖 9)表說明各訓練 model 生成 chit-chat 的候選者的配置，而執行的平台是 ParlAI。

Generative Model	Beam Size	Minimum Beam Length
BlenderBot (90M)	10	1
BlenderBot (90M)	10	5
BlenderBot (90M)	10	20
BlenderBot (90M)	30	20
BlenderBot (2.7B)	10	1
BlenderBot (2.7B)	10	5
BlenderBot (2.7B)	10	20
BlenderBot (2.7B)	30	20
BlenderBot (9.4B)	10	1
BlenderBot (9.4B)	10	20
GPT-2 (117M)	10	1
GPT-2 (345M)	1	1
GPT-2 (345M)	10	1
GPT-2 (345M)	10	5
GPT-2 (345M)	10	20
GPT-2 (345M)	30	1
GPT-2 (762M)	10	1

Table 6: Employed models and decoding parameters for candidate generation.

(圖 9)

2.4.3 Candidate Filtering

註釋實驗結果顯示，只有大約 1/10 的 annotations 候選者是合適的。因此，建議建立一個過濾模型，而不是直接將候選者進行人工註釋。首先自動過濾掉不太可能具有良好質量的候選者，以減少人工註釋的工作量。過濾器是一個混合模型，基於 RoBERTa 的二位元分類器和規則排序器組成。分類器將 Task-Oriented dialogues 作為輸入，其中用一對特殊標記明確地包圍添加的 chit-chat 候選對象，以幫助模型定位候選對象。用 1700 個候選者訓練分類器，這些候選者從操作者的註釋中被標記為好/壞。排名器最初根據二元分類器輸出的事後機率對每個候選者進行排名，接著它會降低與壞模式列表匹配候選者的等級。大多數不良模式與新引入的假冒信息有關（例如，包含 URL/電子郵件地址、電話號碼、時間或金額等）。其餘的不良模式主要是關於文本類型（例如，包含電子郵件簽收，例如“致以問候”）和格式（例如，標點符號的濫用）。最後，排名器提高 (i) 不常見候選者和 (ii) 與其他候選者不同的候選者的排名，來進行對話和增強的系統響應。我們通過 Levenshtein distance 來衡量相似度。

2.4.4 Annotation

註釋的評分標準由 4 個指標來去評斷，人工標註者可根據評分標準來去標記出糟糕的 chit-chat 候選者。其中 4 個指標分別是，

- (1) **Inappropriate:** 候選者與上下文不適合（例如，重複、不自然），或者與上下文或 AI 助理的角色相互矛盾。
- (2) **Misleading:** 候選人提供了假的或無法立即驗證的信息。
- (3) **Social:** 候選者透過適當地切換主題、詢問隨意自然的後續問題或參雜社交玩笑來保持對話流暢。
- (4) **Useful:** 候選人通過適當地提供意見、評論或相關和真實的信息來增強對話。

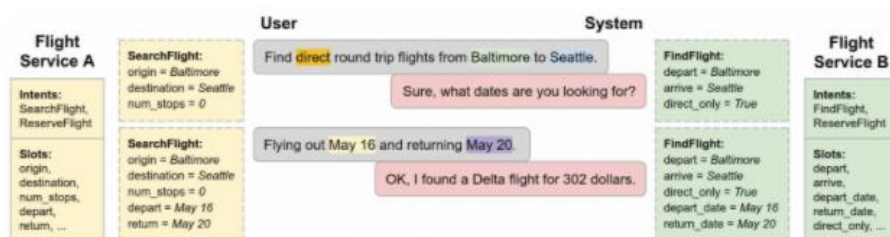
Metric	Value
# of candidates	228, 250
# of unique candidates	68, 406
vocabulary size	10, 005
# of distinct 2-grams	59, 259
# of distinct 3-grams	131, 989
# of distinct 4-grams	195, 508
# of distinct 5-grams	239, 278
average length (in tokens)	8.7
# of good candidates (%)	94, 600 (41.4)
◊ social	86, 324 (37.8)
◊ useful	7, 681 (3.4)
◊ social & useful	577 (0.3)
◊ other (good)	18 (0.0)
# of bad candidates (%)	133, 650 (58.6)
◊ inappropriate	127, 648 (55.9)
◊ misleading	5, 800 (2.5)
◊ inappropriate & misleading	164 (0.1)
◊ other (bad)	38 (0.0)

(圖 10)

3 Approaches

3.1 The Schema-Guided

定義一個包含所有 service 的 schema 並將其整合是一件很困難的事情。因此有文獻提出一種叫做 schema-guide approach 的方法來去用較輕鬆的方式整合 services 和 APIs。使用不包含特定領域或服務特定參數的單一統一模型來根據這些 schema 元素進行預測。(圖 11) 顯示了相同對話的對話狀態表示對於兩個不同的 service 是如何變化的。在這裡，出發和到達城市資訊是透過兩個 service 中功能相似但名稱不同的 slot 來擷取到的。值得注意的是，對話狀態表示取決於所考慮的 schema，該 schema 作為輸入提供，用戶和系統對話也是如此。



(圖 11)

使用單一模型有助於在相關的 services 之間傳遞相似的知識。由於該模型利用語義接口與未經過訓練的看不見的 services 或 APIs。它對諸如向服務添加的 intent 或 slot 之類的更改也能夠應付。

3.2 SimpleTOD

3.2.1 Task-Oriented Dialogue

面向任務的對話 (TOD) 在三個子任務上進行評估：dialogue state (belief state) tracking、對話管理 (action/decision prediction) 和響應生成。SimpleTOD 讀取所有之前的回合作為上下文， $C_t = [U_0; S_0; \dots; U_t]$ 。它生成一個 belief state (B_t)，這是一個記錄特定域中 slot 的三位元組列表：
(domain, slot_name, value) ; belief state 用於查詢數據庫的信息。

$$B_t = \text{SimpleTOD}(C_t)$$

搜索從數據庫中滿足 belief state 狀態條件的行。返回的行稍後可將行作辭彙化，但 SimpleTOD 僅將聚合的數據庫搜索結果作為輸入，然後將 Ct、Bt 和 Dt 條件串接在一起作為單個序列的 decide actions(At)。

$$A_t = \text{SimpleTOD}([C_t, B_t, D_t])$$

這些操作生成另一個三位元組列表：(domain, action_type, slot_name)。一個去詞彙化的響應 St 先前基於單個序列所得到的所有資訊。

$$S_t = \text{SimpleTOD}([C_t, B_t, D_t, A_t])$$

3.2.2 Causal Language Modeling

單個訓練序列上所得到的串聯資訊 $x_t = [C_t; B_t; D_t; A_t; S_t]$ ，將針對序列 x_t 上的聯合概率進行建模。語言建模的目標是學習 $p(x)$ 。使用機率的鍊式法則分析這個分佈是很自然的，並訓練一個帶有參數的神經網絡，以最小化數據集上的相似函數。

$$p(x) = \prod_{i=1}^n p(x_i | x_{<i}) \quad \mathcal{L}(D) = - \sum_{t=1}^{|D|} \sum_{i=1}^{n_t} \log p_{\theta}(x_i^t | x_{<i}^t)$$

3.2.3 Architecture

我們訓練 Transformer 的變型來學習這些條件分佈。包含 n 個 tokens 的序列作為 n 個向量的序列嵌入到 \mathbb{R}^d 中。每個向量是學習的 tokens 嵌入和正弦位置嵌入的總和。向量序列堆疊成矩陣 X 屬於 $\mathbb{R}^{n \times d}$ 並由 L 層注意力層處理。第 i 層由兩個塊組成，用來保留模型維度 d 。第一個 block 使用具有 k 個 multi-head attention。第二個 block 使用帶有 ReLU 激活的前饋網絡，將輸入投影到內部維度 f 。每個 block 在層標準化之前的核心功能，並在它之後使用 residual connection，用最後一層來計算分數。

$$\begin{aligned} \text{Attention}(X, Y, Z) &= \text{softmax} \left(\frac{\text{mask}(XY^T)}{\sqrt{d}} \right) Z \\ \text{MultiHead}(X, k) &= [h_1; \dots; h_k] W_o \\ \text{where } h_j &= \text{Attention}(XW_j^1, XW_j^2, XW_j^3) \\ \hline \text{This operation is parameterized by } U &\in \mathbb{R}^{d \times f} \text{ and } V \in \mathbb{R}^{f \times d}; \\ FF(X) &= \max(0, XU)V \\ \text{Scores}(X_0) &= \text{LayerNorm}(X_L) W_{vocab} \end{aligned}$$

在訓練期間，這些分數是交叉熵損失函數的輸入。在生成 token 期間，對應於最終標記的分數使用 softmax 進行正規化，從而產生用於採樣新 token 的分佈。

3.2.4 Training Details & Action Generation

根據實驗結果，SimpleTOD 的實驗在 Huggingface Transformers 中使用 GPT-2 和 DistilGPT2 的 hyperparameters，且長度超過 1024 個 tokens 的序列將被截斷。包括完全忽略數據庫搜索結果的實驗，(圖 12)展示沒有數據庫搜索信息的 SimpleTOD 其有效性不輸有進行數據搜索的 model。

Belief State	DB Search	Action	Inform	Success	BLEU	Combined
generated	oracle	generated	79.3	65.4	16.01	87.36
generated	dynamic	generated	83.4	67.1	14.99	90.24
generated	-	generated	85	70.5	15.23	92.98

Table 3: Action and response generation on MultiWOZ 2.1 for SimpleTOD.

(圖 12)

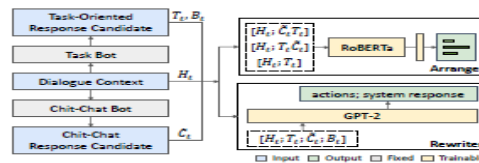
3.3 SimpleTOD add chit-chat

3.3.1 Task Formulations

由於 oracle 的資料庫（即 oracle belief state、oracle action decisions）在實際使用中不可用，並且 SGD 數據集沒有發布有關 chit-chat 的數據庫，因此生成一個去辭彙化且不使用 oracle 資訊和數據庫搜索結果的 task oriented responses 的 end-to-end 設置。

3.3.2 Models

在這個章節中，為了能添加 chit-chat 的功能，我們找了有一篇文獻在處理這個功能的程序。該文獻提出了一個包含 chit-chat 的 SimpleTOD 以及兩個新模型（圖 13），可以更明確地在 chit-chat and task oriented responses 之間進行代碼切換。



(圖 13)

- (1) **SimpleTOD+:** 我們透過在訓練期間引入一個特殊的 new dialogue action chit-chat 和好的 chit-chat candidates 來構建輸入序列，從而擴展 SimpleTOD。具體來說，讓 C^t 表示該對話回合中良好的 chit-chat 候選者。如果 C^t 為空，就當作與一般 SimpleTOD 相同的訓練序列。否則，標記為要預先添加到下一回合對話的候選對象，我們使用 H_t 、 B_t 、[cit-chat]、 A_t 、 C_t 和 T_t 作為訓練序列。
- (2) **Arranger:** 為了每系統回合 t 產生響應，該模型將(i)對話歷史 H_t ，(ii) 閒聊響應 C_t 作為輸入，由基於 H_t 生成 chit-chat，以及 (iii) task oriented response 基於 H_t 的 task oriented response 模型生成的 T_t 。該模型透過 RoBERTa 編碼器對 H_t 和這三個響應中的每一個的串接進行編碼，並將結果表示通過線性加 softmax 層進行選擇。
- (3) **Rewriter:** 此模型複寫了現成的 task-oriented dialogue 模型和現成的 chit-chat 模型的輸出。它直接透過未經修改過 task-oriented model 的 belief state 及經由因果語言模型所生成的 action decisions 和系統響應。因果語言模型與 SimpleTOD+ 的不同之處在於，它在每個訓練序列中在 H_t 和 B_t 之間添加了兩個額外的參數 T_t 和 C_t 。

4 Experiments

4.1 DST

4.1.1 Data preprocess

(1) TA Test data format

```
[
  {
    "dialogue_id": "PMUL0320",
    "services": [
      "restaurant",
      "taxi",
      "hotel"
    ],
    "turns": [
      {
        "turn_id": 0,
        "speaker": "USER",
        "utterance": "Hi, I'm looking for a hotel to stay in that includes free wifi. I'm looking to stay in a hotel, not a guesthouse."
      },
      {
        "turn_id": 1,
        "speaker": "SYSTEM",
        "utterance": "Yes I have many options. What is your preferred price point?"
      },
      {
        "turn_id": 2,
        "speaker": "USER",
        "utterance": "I would like one in the moderate price range and with free parking."
      },
      {
        "turn_id": 3,
        "speaker": "SYSTEM",
        "utterance": "I would recommend the Lovell Lodge. It is indeed a hotel despite it's name and is very cute."
      },
      {
        "turn_id": 4,
        "speaker": "USER",
        "utterance": "Okay, please book that for 3 people and 2 nights starting from Friday."
      },
      {
        "turn_id": 5,
        "speaker": "SYSTEM",
        "utterance": "Booking was successful. Reference number is : 9HMD04UW . anything else?"
      },
      {
        "turn_id": 6,
        "speaker": "USER",
        "utterance": "I would love to find a restaurant in the same price range as the Lovell Lodge."
      },
      {
        "turn_id": 7,
        "speaker": "SYSTEM",
        "utterance": "In what area?"
      },
      {
        "turn_id": 8,
        "speaker": "USER",
        "utterance": "I would like the west please."
      }
    ]
  }
]
```

(2) After doing data preprocess

```
[
  {
    "dialogue_id": "PWUL0320",
    "services": [
      "restaurant",
      "taxi",
      "hotel"
    ],
    "turns": [
      {
        "frames": [
          {
            "actions": [],
            "service": "restaurant",
            "slots": [],
            "state": {
              "active_intent": [],
              "requested_slots": [],
              "slot_values": {}
            }
          },
          {
            "actions": [],
            "service": "taxi",
            "slots": [],
            "state": {
              "active_intent": [],
              "requested_slots": [],
              "slot_values": {}
            }
          },
          {
            "actions": [],
            "service": "hotel",
            "slots": [],
            "state": {
              "active_intent": [],
              "requested_slots": [],
              "slot_values": {}
            }
          }
        ],
        "speaker": "USER",
        "turn_id": "0",
        "utterance": "Hi, I'm looking for a hotel to stay in that includes free wifi. I'm looking to stay in a hotel, not a guesthouse."
      },
      {
        "frames": [
          {
            "actions": [],
            "service": "SYSTEM",
            "slots": [],
            "state": {
              "active_intent": [],
              "requested_slots": [],
              "slot_values": {}
            }
          }
        ],
        "speaker": "SYSTEM",
        "turn_id": "1",
        "utterance": "Yes I have many options. What is your preferred price point?"
      }
    ]
  }
]
```

4.1.2 Data Embedding

目的:為了避免每次執行程式都要重做資料預處理

(1) Dialogues embedding

名稱	修改日期	類型	大小
 dstc8_all_train_examples.tf_record	2021/6/21 下午 05:53	TF_RECORD 檔案	779,873 KB

```
NFO:tensorflow:Processed 0 dialogs.
0623 17:07:40.876499 14352 data_utils.py:120] Processed 0 dialogs.
NFO:tensorflow:Processed 1000 dialogs.
0623 17:07:46.116821 14352 data_utils.py:120] Processed 1000 dialogs.
NFO:tensorflow:Writing example 0 of 30288
0623 17:07:51.054855 14352 data_utils.py:644] Writing example 0 of 30288
NFO:tensorflow:Writing example 10000 of 30288
0623 17:07:54.565835 14352 data_utils.py:644] Writing example 10000 of 30288
NFO:tensorflow:Writing example 20000 of 30288
0623 17:07:58.315589 14352 data_utils.py:644] Writing example 20000 of 30288
NFO:tensorflow:Writing example 30000 of 30288
0623 17:08:01.928922 14352 data_utils.py:644] Writing example 30000 of 30288
NFO:tensorflow:Finish generating the dialogue examples.
0623 17:08:02.234810 14352 predict_seen.py:773] Finish generating the dialogue examples.
NFO:tensorflow:Start generating the schema embeddings.
```

(2) Schema embedding

名稱	修改日期	類型	大小
 train_pretrained_schema_embedding....	2021/6/14 下午 08:06	NPY 檔案	159,017 KB

```
NFO:tensorflow:Generating embeddings for service Alarm_1.
0623 17:40:30.029751 14732 extract_schema_embedding.py:383] Generating embeddings for service Alarm_1.
NFO:tensorflow:Generating embeddings for service Banks_1.
0623 17:40:30.089739 14732 extract_schema_embedding.py:383] Generating embeddings for service Banks_1.
NFO:tensorflow:Generating embeddings for service Banks_2.
0623 17:40:30.149718 14732 extract_schema_embedding.py:383] Generating embeddings for service Banks_2.
NFO:tensorflow:Generating embeddings for service Buses_1.
0623 17:40:30.244721 14732 extract_schema_embedding.py:383] Generating embeddings for service Buses_1.
NFO:tensorflow:Generating embeddings for service Buses_2.
0623 17:40:30.385662 14732 extract_schema_embedding.py:383] Generating embeddings for service Buses_2.
NFO:tensorflow:Generating embeddings for service Buses_3.
0623 17:40:30.529655 14732 extract_schema_embedding.py:383] Generating embeddings for service Buses_3.
NFO:tensorflow:Generating embeddings for service Calendar_1.
0623 17:40:30.717597 14732 extract_schema_embedding.py:383] Generating embeddings for service Calendar_1.
NFO:tensorflow:Generating embeddings for service Events_1.
0623 17:40:30.769568 14732 extract_schema_embedding.py:383] Generating embeddings for service Events_1.
NFO:tensorflow:Generating embeddings for service Events_2.
0623 17:40:30.957509 14732 extract_schema_embedding.py:383] Generating embeddings for service Events_2.
NFO:tensorflow:Generating embeddings for service Events_3.
0623 17:40:31.102463 14732 extract_schema_embedding.py:383] Generating embeddings for service Events_3.
NFO:tensorflow:Generating embeddings for service Flights_1.
0623 17:40:31.291418 14732 extract_schema_embedding.py:383] Generating embeddings for service Flights_1.
NFO:tensorflow:Generating embeddings for service Flights_2.
0623 17:40:31.573372 14732 extract_schema_embedding.py:383] Generating embeddings for service Flights_2.
NFO:tensorflow:Generating embeddings for service Flights_3.
0623 17:40:31.810398 14732 extract_schema_embedding.py:383] Generating embeddings for service Flights_3.
NFO:tensorflow:Generating embeddings for service Flights_4.
0623 17:40:32.090269 14732 extract_schema_embedding.py:383] Generating embeddings for service Flights_4.
NFO:tensorflow:Generating embeddings for service Homes_1.
0623 17:40:32.327613 14732 extract_schema_embedding.py:383] Generating embeddings for service Homes_1.
NFO:tensorflow:Generating embeddings for service Homes_2.
0623 17:40:32.516916 14732 extract_schema_embedding.py:383] Generating embeddings for service Homes_2.
NFO:tensorflow:Generating embeddings for service Hotels_1.
0623 17:40:32.707872 14732 extract_schema_embedding.py:383] Generating embeddings for service Hotels_1.
NFO:tensorflow:Generating embeddings for service Hotels_2.
```

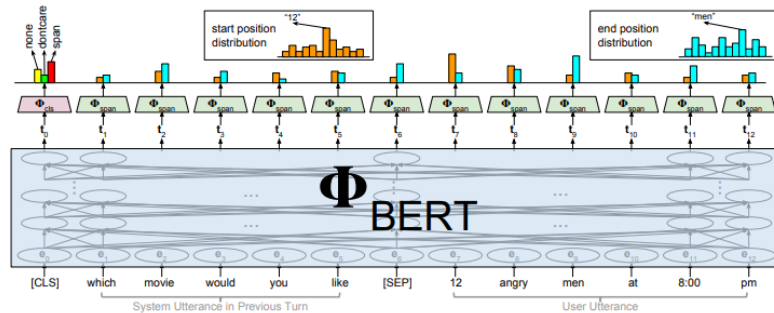
(3) 在 config.py 檔中，也有提供實作不同任務如下所示：

本次使用 dstc8_all 來時做 DST 任務，在實作過程中，因為 max_num_cat_slot、max_num_noncat_slot、max_num_value_per_cat_slot、max_num_intent 參數設太小所以報錯，debug 了許久。而 config 檔中也有紀錄各 data dialogue 的數量，如果要改 train data 或其他 dialogue 的數量需要透過修改 range 中的值，例如 TA 給的 train data 中有 dialogue001~dialogue138，因此我們將 "train": range(1, 139) 設定。

```
DATASET_CONFIG = {
    "dstc8_single_domain":
        DatasetConfig(
            file_ranges={
                "train": range(1, 139),
                "dev": range(1, 21),
                "test_seen": range(1, 17),
                "test_unseen": range(1, 6)
            },
            max_num_cat_slot=6,
            max_num_noncat_slot=12,
            max_num_value_per_cat_slot=12,
            max_num_intent=4),
    "dstc8_multi_domain":
        DatasetConfig(
            file_ranges={
                "train": range(1, 139),
                "dev": range(1, 21),
                "test_seen": range(1, 17),
                "test_unseen": range(1, 6)
            },
            max_num_cat_slot=20,
            max_num_noncat_slot=20,
            max_num_value_per_cat_slot=20,
            max_num_intent=20),
    "dstc8_all":
        DatasetConfig(
            file_ranges={
                "train": range(1, 139),
                "dev": range(1, 21),
                "test_seen": range(1, 17),
                "test_unseen": range(1, 6)
            },
            max_num_cat_slot=20,
            max_num_noncat_slot=20,
            max_num_value_per_cat_slot=20,
            max_num_intent=20),
```

4.1.3 Pretrain model

簡介：將 [CLS] + system utterance + [SEP] + user utterance + [SEP] 做 tokenize 放入 model (有點類似 squad task)，第一個 token [CLS] 的 output 做分類 (none、dontcare、span 三類)，如果輸出 span 則判斷哪個 token 的 start position 及 end position 輸出的機率分布比較高 (start position 彼此共享參數 / end position 彼此共享參數，具有參數較少的優點)，進而找出使用者所說的重要資訊。



➤ Bert-based




(1) Config.json :

```
{
  "attention_probs_dropout_prob": 0.1,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "max_position_embeddings": 512,
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "type_vocab_size": 2,
  "vocab_size": 28996
}
```

(2) Vocab.txt:

```
cat
exception
convinced
executed
pushing
dollars
replacing
soccer
manufacturing
##ros
expensive
kicked
minimum
Josh
coastal
Chase
ha
Thailand
publications
deputy
Sometimes
Angel
effectively
##illa
criticism
conduct
Serbian
landscape
NY
```

(3) Ckpt (size : 425520 KB):

 bert_model.ckpt.data-00000-of-00001	2018/10/19 上午 06:38	DATA-00000-OF...	425,520 KB
 bert_model.ckpt.index	2018/10/19 上午 06:38	INDEX 檔案	9 KB
 bert_model.ckpt.meta	2018/10/19 上午 06:38	META 檔案	885 KB

➤ Roberta-base:





(1) Config.json:

```
{
  "architectures": [
    "RobertaForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "eos_token_id": 2,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-05,
  "max_position_embeddings": 514,
  "model_type": "roberta",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 1,
  "type_vocab_size": 1,
  "vocab_size": 50265
}
```

(2) Vocab.txt:

```
[["<s>": 0, "<pad>": 1, "</s>": 2, "<unk>": 3, ".": 4, "Gthe": 5, ",": 6, "Gto": 7,
8, "Gthere": 89, "t": 90, "GHe": 91, "Gnew": 92, "Gack": 93, "Glast": 94, "Gjust":
95, "Ggovernment": 168, "Gway": 169, "We": 170, "Gmany": 171, "Gthen": 172, "Gwork": 1
45, "3": 246, "Gcountry": 247, "GR": 248, "Gpolice": 249, "A": 250, "Glong": 251,
"Gsupport": 323, "ie": 324, "Gbillion": 325, "Gt": 326, "Gshares": 327, "I": 328,
398, "Gdidn": 399, "Glocal": 400, "6": 401, "Gsomething": 402, "Gcase": 403, "Gall
470, "Ghead": 471, "Gplayers": 472, "Gdoes": 473, "Ghealth": 474, "Gm": 475, "Gpo
3, "Gservice": 544, "G16": 545, "Glooking": 546, "Gheld": 547, "ve": 548, "Gwethe
ow": 614, "Genough": 615, "GWhile": 616, "Gfurther": 617, "Gpost": 618, "Gfeel": 6
683, "Gknown": 684, "Glost": 685, "Gsure": 686, "us": 687, "Gweeks": 688, "Gfood":
Gfact": 754, "GHigh": 755, "Gcareer": 756, "im": 757, "Ginternational": 758, "GNov
"Gg": 821, "Gfilm": 822, "Gnearly": 823, "Gcenter": 824, "Gvisit": 825, "Ggroup":
: 894, "ah": 895, "GCanada": 896, "Gla": 897, "Gresult": 898, "Gaccess": 899, "Gvc
died": 962, "Ginvolved": 963, "Gfriends": 964, "Gisn": 965, "Gworth": 966, "ik": 9
ve": 1031, "Gfight": 1032, "ings": 1033, "Ghope": 1034, "Gsummer": 1035, "Gofficer
"50": 1096, "anø": 1097, "Ghospital": 1098, "Gbad": 1099, "Gaddress": 1100, "Gkore
```

(3) Ckpt(size:486897KB):

 checkpoint	2021/6/9 下午 10:39	檔案	1 KB
 roberta_base.ckpt.data-00000-of-000...	2021/6/9 下午 10:39	DATA-00000-OF...	486,897 KB
 roberta_base.ckpt.index	2021/6/9 下午 10:39	INDEX 檔案	9 KB
 roberta_base.ckpt.meta	2021/6/9 下午 10:39	META 檔案	494 KB

4.1.4 Hyperparameter

```
flags.DEFINE_integer(
    "max_seq_length", 128,
    "The maximum total input sequence length after WordPiece tokenization. "
    "Sequences longer than this will be truncated, and sequences shorter "
    "than this will be padded.")

flags.DEFINE_float("dropout_rate", 0.2,
    "Dropout rate for BERT representations.")

# Hyperparameters and optimization related flags.
flags.DEFINE_integer("train_batch_size", 8, "Total batch size for training.")

flags.DEFINE_integer("eval_batch_size", 8, "Total batch size for eval.")

flags.DEFINE_integer("predict_batch_size", 8, "Total batch size for predict.")

flags.DEFINE_float("learning_rate", 5e-5, "The initial learning rate for Adam.")

flags.DEFINE_float("num_train_epochs", 100.0,
    "Total number of training epochs to perform.")

flags.DEFINE_float(
    "warmup_proportion", 0.2,
    "Proportion of training to perform linear learning rate warmup for. "
    "E.g., 0.1 = 10% of training.")

flags.DEFINE_integer("save_checkpoints_steps", 10000,
    "How often to save the model checkpoint.")
```

4.1.5 各參數介紹

- Bert_ckpt_dir: initial fine-tune model 所擺放的位置 (可以是 bert cased / Roberta-base

```
flags.DEFINE_string("bert_ckpt_dir", "./cased_L-12_H-768_A-12/",
    "Directory containing pre-trained BERT checkpoint.")
```

- Task_name: 任務名稱包含了 dstc8_multi_domain、dstc8_single_domain、dstc8_all、multiwoz21_all 這四類。

```
flags.DEFINE_enum("task_name", 'dstc8_all', config.DATASET_CONFIG.keys(),
    "The name of the task to train.")
```

- Dstc8_data_dir: train / dev / test_seen / test_unseen 要在同一個資料夾下，Dstc8_data_dir 為此資料夾位置。

```
flags.DEFINE_string(
    "dstc8_data_dir", "./data/",
    "Directory for the downloaded DSTC8 data, which contains the dialogue files"
    " and schema files of all datasets (eg train, dev)")
```

- Run_mode: do train 或 do predict。

```
flags.DEFINE_enum("run_mode", "train", ["train", "predict"],
    "The mode to run the script in.")
```

- Output_dir: training 過程中 ckpt 存放位置，以及 predict 出來的資料擺放位置。

```
flags.DEFINE_string(
    "output_dir", "./output_ft/",
    "The output directory where the model checkpoints will be written.")
```

- Schema_embedding_dir / dialogues_example_dir : dialogues 和 schema 做完 embedding 檔案輸出位置。

```
flags.DEFINE_string(
    "schema_embedding_dir", "./output_schema_embedding/",
    "Directory where .npy file for embedding of entities (slots, values,"
    " intents) in the dataset_split's schema are stored.")

flags.DEFINE_string(
    "dialogues_example_dir", "./output_dialogues_example/",
    "Directory where tf.record of DSTC8 dialogues data are stored.")
```

- Dataset_split: 使用哪個資料做 train 或 predict，可選擇 train /dev/ test_seen/test_unseen 共四種 與 Run_mode 做搭配。

```
flags.DEFINE_enum("dataset_split", "train", ["train", "dev", "test_seen", "test_unseen"],
    "Dataset split for training / prediction.")
```

- Eval_ckpt: 填入 int(ckpt 步數)，如果要連續使用不同 ckpt 做 predict，用逗號座分隔。

```
flags.DEFINE_string(
    "eval_ckpt", "105000",
    "Comma separated numbers, each being a step number of model checkpoint"
    " which makes predictions.")
```

4.1.6 MutiWOZ → SGD

- MutiWoz 2.1 data 如下所示:

.DS_Store	2020/4/22 下午 06:09	DS_STORE 檔案	7 KB
.README.swp	2019/11/15 下午 06:58	SWP 檔案	12 KB
attraction_db.json	2019/11/15 下午 06:58	JSON 檔案	35 KB
data.json	2020/4/22 下午 05:35	JSON 檔案	365,649 KB
hospital_db.json	2019/11/15 下午 06:58	JSON 檔案	8 KB
hotel_db.json	2019/11/15 下午 06:58	JSON 檔案	18 KB
ontology.json	2019/11/15 下午 06:58	JSON 檔案	40 KB
police_db.json	2019/11/15 下午 06:58	JSON 檔案	1 KB
README	2019/11/15 下午 06:58	檔案	4 KB
restaurant_db.json	2019/11/15 下午 06:58	JSON 檔案	56 KB
slot_descriptions.json	2019/11/15 下午 06:58	JSON 檔案	5 KB
system_acts.json	2019/11/15 下午 06:58	JSON 檔案	23,779 KB
taxi_db.json	2019/11/15 下午 06:58	JSON 檔案	1 KB
testListFile.txt	2019/11/15 下午 06:58	文字文件	14 KB
tokenization.md	2019/11/15 下午 06:58	MD 檔案	1 KB
train_db.json	2019/11/15 下午 06:58	JSON 檔案	732 KB
valListFile.txt	2019/11/15 下午 06:58	文字文件	14 KB

➤ 轉換成 SGD 形式如下所示:

(1) Train

dialogues_001.json	2021/6/7 下午 07:22	JSON 檔案	8,692 KB
dialogues_002.json	2021/6/7 下午 07:22	JSON 檔案	8,685 KB
dialogues_003.json	2021/6/7 下午 07:22	JSON 檔案	8,517 KB
dialogues_004.json	2021/6/7 下午 07:22	JSON 檔案	8,794 KB
dialogues_005.json	2021/6/7 下午 07:22	JSON 檔案	8,904 KB
dialogues_006.json	2021/6/7 下午 07:22	JSON 檔案	8,573 KB
dialogues_007.json	2021/6/7 下午 07:22	JSON 檔案	8,667 KB
dialogues_008.json	2021/6/7 下午 07:22	JSON 檔案	8,741 KB
dialogues_009.json	2021/6/7 下午 07:23	JSON 檔案	8,700 KB
dialogues_010.json	2021/6/7 下午 07:23	JSON 檔案	8,392 KB
dialogues_011.json	2021/6/7 下午 07:23	JSON 檔案	8,381 KB
dialogues_012.json	2021/6/7 下午 07:23	JSON 檔案	8,681 KB
dialogues_013.json	2021/6/7 下午 07:23	JSON 檔案	8,499 KB
dialogues_014.json	2021/6/7 下午 07:23	JSON 檔案	8,752 KB
dialogues_015.json	2021/6/7 下午 07:23	JSON 檔案	8,576 KB
dialogues_016.json	2021/6/7 下午 07:23	JSON 檔案	8,827 KB
dialogues_017.json	2021/6/7 下午 07:23	JSON 檔案	4,195 KB
schema.json	2021/6/7 下午 07:21	JSON 檔案	16 KB

(2) Dev

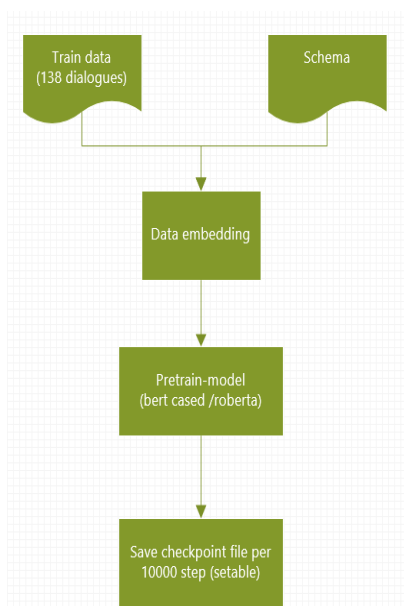
名稱	修改日期	類型	大小
dialogues_001.json	2021/6/7 下午 07:21	JSON 檔案	9,826 KB
dialogues_002.json	2021/6/7 下午 07:21	JSON 檔案	8,926 KB
schema.json	2021/6/7 下午 07:21	JSON 檔案	16 KB

(3) Test

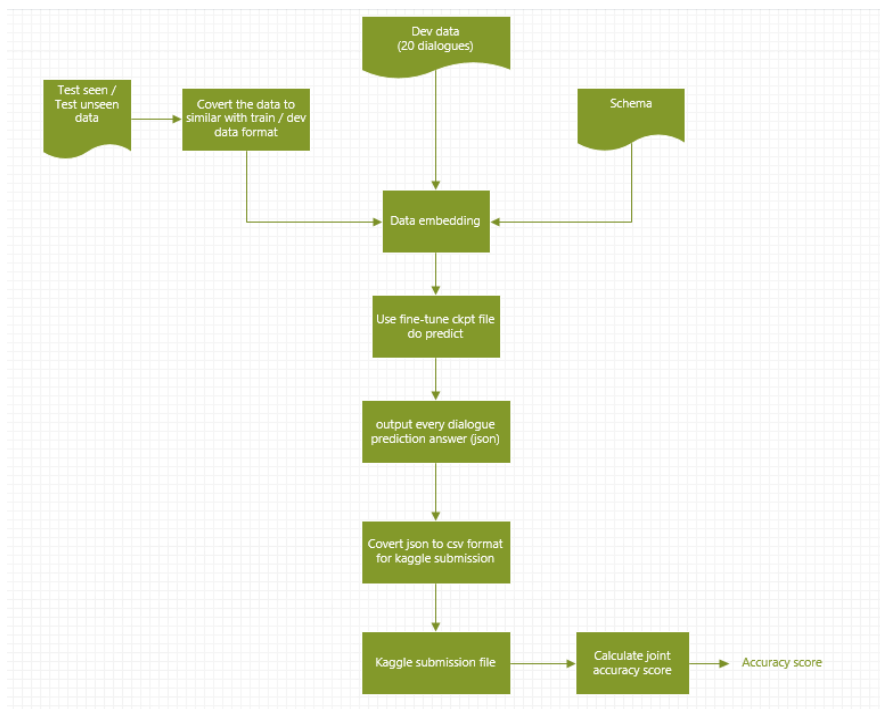
名稱	修改日期	類型	大小
dialogues_001.json	2021/6/7 下午 07:21	JSON 檔案	9,865 KB
dialogues_002.json	2021/6/7 下午 07:21	JSON 檔案	9,000 KB
schema.json	2021/6/7 下午 07:21	JSON 檔案	16 KB

4.1.7 任務整體流程圖

➤ Train:



➤ **Predict on dev/test:**



4.1.8 環境設定

因我們使用的範例程式是 tensorflow，以下是本次期末專題會用到的套件與相對應版本如下：

absl-py>=0.12.0

fuzzywuzzy>=0.17.0

numpy>=1.16.1

six>=1.12.0

transformers == 2.11.0

datasets == 1.6.2

tqdm == 4.49.0

pandas == 1.2.4

首先，使用 command，cd 到該.txt 位置，輸入 pip install -r requirements.txt。

接著要安裝本次所使用的 tensorflow_GPU，因版本問題，所以需要使用到 **python3.7**。

如果有 anaconda 可創立一個新環境，並且使用以下指令：

conda install tensorflow-gpu=1.14.0

如果使用 conda 指令安裝 tensorflow，只要對好 python 版本，會自動安裝相對應的 cudnn 與 cudatoolkit。

如果使用 conda 安裝失敗，就需要自行使用 pip 指令安裝相關套件。

安裝完成之後可使用下列程式碼進行測試：

```
import tensorflow as tf
tf.__version__
tf.test.is_gpu_available()
```

4.1.9 輸出檔案資料處理

Dialogues → .csv 格式

下圖為 model 預測出來的檔案格式，都是以.json 所組成

 dialogues_001.json	2021/6/9 下午 07:58	JSON 檔案	2,879 KB
 dialogues_002.json	2021/6/9 下午 07:58	JSON 檔案	2,466 KB
 dialogues_003.json	2021/6/9 下午 07:58	JSON 檔案	2,426 KB
 dialogues_004.json	2021/6/9 下午 07:58	JSON 檔案	2,798 KB
 dialogues_005.json	2021/6/9 下午 07:58	JSON 檔案	3,563 KB
 dialogues_006.json	2021/6/9 下午 07:58	JSON 檔案	3,211 KB
 dialogues_007.json	2021/6/9 下午 07:58	JSON 檔案	2,427 KB
 dialogues_008.json	2021/6/9 下午 07:58	JSON 檔案	3,515 KB
 dialogues_009.json	2021/6/9 下午 07:58	JSON 檔案	2,486 KB
 dialogues_010.json	2021/6/9 下午 07:58	JSON 檔案	3,400 KB
 dialogues_011.json	2021/6/9 下午 07:58	JSON 檔案	2,181 KB
 dialogues_012.json	2021/6/9 下午 07:58	JSON 檔案	3,187 KB
 dialogues_013.json	2021/6/9 下午 07:58	JSON 檔案	3,541 KB
 dialogues_014.json	2021/6/9 下午 07:58	JSON 檔案	3,422 KB
 dialogues_015.json	2021/6/9 下午 07:59	JSON 檔案	2,524 KB
 dialogues_016.json	2021/6/9 下午 07:59	JSON 檔案	2,464 KB
 dialogues_017.json	2021/6/9 下午 07:59	JSON 檔案	2,416 KB
 dialogues_018.json	2021/6/9 下午 07:59	JSON 檔案	2,212 KB
 dialogues_019.json	2021/6/9 下午 07:59	JSON 檔案	2,257 KB
 dialogues_020.json	2021/6/9 下午 07:59	JSON 檔案	456 KB

每個 dialogue 檔案中的格式如下圖，因為 System 與 User 會來回對話，在選取答案上，我們會取**最後一個 state 的 slot 與 value**，作為這個 dialogue 的輸出。

```
[
  {
    "dialogue_id": "11_00000",
    "services": [
      "Media_2",
      "Music_1"
    ],
    "turns": [
      {
        {
          "beginning": [],
          "end": [],
          "frames": [
            {
              {
                "actions": [
                  {
                    "act": "REQUEST",
                    "canonical_values": [],
                    "slot": "genre",
                    "values": []
                  }
                ],
                "service": "Media_2",
                "slots": []
              }
            ],
            "speaker": "SYSTEM",
            "utterance": "What genre do you want?"
          },
          {
            "frames": [
              {
                {
                  "actions": [
                    {
                      "act": "AFFIRM_INTENT",
                      "canonical_values": [],
                      "slot": "",
                      "values": []
                    }
                  ],
                  "service": "Music_1",
                  "state": {
                    "active_intent": "NONE",
                    "requested_slots": [],
                    "slot_values": {
                      "song_name": [
                        "Adorn"
                      ]
                    }
                  }
                }
              ]
            }
          ]
        },
        1,
      ]
    ]
  }
]
```

將輸出資料夾的所有 json 檔讀取，並加每個 dialogue 的格式轉換成下列格式：

```
ans = {
    dialogue_id1: {service1-slot1: value1, service1-slot2: value2, ...},
    dialogue_id2: {service2-slot3: value3, service3-slot4: value4, ...},
}
```

之後透過助教提供的 state_to_csv.py 轉換成 kaggle 的繳交格式：

{ id: state }，state 之間使用”|”隔開。

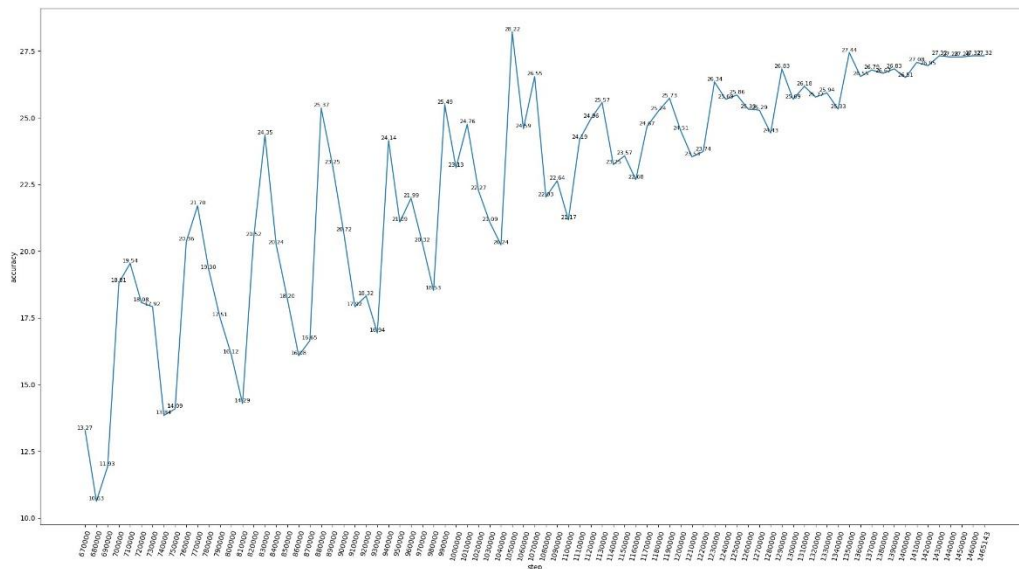
以下是自行取出的答案格式，其中，可以看到有{}，也就是模型沒有預測出答案，會再經過助教給的程式碼，將所有的{}都轉成 None。

```
{
  '11_00000': {'Music_1-song_name': 'Adorn'},
  '11_00001': {},
  '11_00002': {'Music_1-song_name': 'Adorn'}
}
```

	A	B	C	D	E	F	G	H	I
1	id	state							
2	10_00000	weather_1-city=palo alto	weather_1-date=14th of this month						
3	10_00001	weather_1-city=cupertino							
4	10_00002	weather_1-city=brisbane							
5	10_00003	weather_1-city=long beach	weather_1-date=7th of this month						
6	10_00004	weather_1-city=healdsburg							
7	10_00005	weather_1-city=lafayette							
8	10_00006	weather_1-city=albany							
9	10_00007	weather_1-city=duncans mills	weather_1-date=next tuesday						
10	10_00008	weather_1-city=montara	weather_1-date=march 14th						
11	10_00009	weather_1-city=point reyes station							
12	10_00010	weather_1-city=eldridge	weather_1-date=march 14th						
13	10_00011	weather_1-city=marshall	weather_1-date=4th of march						
14	10_00012	weather_1-city=pittsburg							
15	10_00013	weather_1-city=vancouver_bcl	weather_1-date=next wednesday						
16	10_00014	weather_1-city=woodside	weather_1-date=next tuesday						
17	10_00015	weather_1-city=south san francisco							
18	10_00016	weather_1-city=paris							
19	10_00017	weather_1-city=vallejo							
20	10_00018	weather_1-city=los gatos	weather_1-date=march 7th						
21	10_00019	weather_1-city=pescadero	weather_1-date=the 14th						
22	10_00020	weather_1-city=burlingame	weather_1-date=12th of march						
23	10_00021	weather_1-city=burlingame	weather_1-date=4th of this month						
24	10_00022	weather_1-city=pleasanton	weather_1-date=next wednesday						
25	10_00023	weather_1-city=oakland							
26	10_00024	weather_1-city=fremont	weather_1-date=thursday next week						
27	10_00025	weather_1-city=valley ford	weather_1-date=the 14th						
28	10_00026	weather_1-city=san jose	weather_1-date=2nd of march						
29	10_00027	weather_1-city=belmont	weather_1-date=march 6th						
30	10_00028	weather_1-city=brisbane	weather_1-date=friday next week						
31	10_00029	weather_1-city=toronto	weather_1-date=11th of march						
32	10_00030	weather_1-city=atlanta							
33	10_00031	weather_1-city=pleasant hill	weather_1-date=march 1st						
34	10_00032	weather_1-city=cotati							
35	10_00033	weather_1-city=el sobrate							
36	10_00034	weather_1-city=marshall							
37	10_00035	weather_1-city=santa rosa							
38	10_00036	weather_1-city=santa clara	weather_1-date=6th of this month						
39	10_00037	weather_1-city=atlanta_gal	weather_1-date=march 9th						
40	10_00038	weather_1-city=richmond	weather_1-date=the 11th						
41	10_00039	weather_1-city=stanford	weather_1-date=today						
42	10_00040	weather_1-city=hercules	weather_1-date=the 3rd						
43	10_00041	weather_1-city=los altos	weather_1-date=the 5th						
44	10_00042	weather_1-city=alamo	weather_1-date=march 2nd						
45	10_00043	weather_1-city=pacifica	weather_1-date=march 8th						
46	10_00044	weather_1-city=point reyes station	weather_1-date=march 9th						
47	10_00045	weather_1-city=forestville							
48	10_00046	weather_1-city=hayward	weather_1-date=the 9th						
49	10_00047	music_1-playback_device=tv	music_1-song_name=wonderful life						
50	10_00048	music_1-playback_device=bedroom speaker	music_1-song_name=barefoot in the park						
51	10_00049	music_1-playback_device=kitchen speaker							

Training curve:

在訓練過程中，我們設定每 10000 steps 會存一個 model 的 ckpt 檔案，並且由這個 ckpt 去 predict 我們的 dev 檔案，作為評估模型的好壞，在 670000 steps 之前因為效果都很糟，所以我們從 670000 開始繪圖至訓練結束，並且使用 matplotlib 將訓練曲線繪製出來(X 軸:steps, Y 軸:accuracy)。訓練前期，學習的曲線震盪比較大，但經過數小時的訓練之後，有慢慢收斂跡象。



Hyper parameters:

Warmup = 0.2

Len = 256

Batch size = 8

Epoch = 25

Dev score:

Dev 計算分數的方式，事先將 Dev ground true 透過程式轉成 csv 的方式，再將預測的 csv 兩者進行 joint accuracy 的計算。

利用程式判斷 id 相同，並且 state 完全正確，才將此 dialogue 視為預測正確。

上課中，雖然還有提到其他的計算準確率方法，我們最後選擇與 kaggle 上計算方式作為我們模型評估的依據。

5. Conclusion

總結來說本次的任務我們完成了 DST 的實作，但是我們因為時間的關係並沒有完成 Chit Chat 的部分，在 model 上我們使用 bert-cased 來作為我們的微調模型，雖然有想過要使用 Roberta-base 來加強準確率但是因為硬體方面的限制 (CUDA out of memory)，所以最後還是使用 bert-cased 來 finetune，在完成本次的實作時我們發現只要從每一個 system 跟 user 的 frames 中注重 slot-value 和 service 其實就足夠當作我們的訓練輸入，然而在我們起初的訓練資料中觀察到 MultiWOZ 與 SGD 對 multi-domain 的答案整理有落差，MultiWOZ 會在每個 turn 保留以往所有 domain 的 state，而 SGD 會在每個 turn 只更新當前 domain 的 state，並丟掉其他 domain 的紀錄。所以為了將這個問題排除就必須更新 SGD 的資料並讓他保留所有 domain 的 state，有看到某些組別解決了資料的不完整後整個準確率有大幅的提升，但是我們重新訓練後的感受並沒有很明顯的提升，我想其實是因為我們這組在使用最一開始的資料時就有用一些方法來排除資料的不統一性。雖然會捨去訓練的資料量不過這樣卻能夠讓我們在最短的時間讓模型開始訓練。

這次的期末專題我們用的是 tensorflow 來完成的，相較於前幾次的作業我們所使用的 pytorch 有些許不同，舉例來說我們的 pretrain model 和 finetune model 就必須是 ckpt 檔而非 bin 檔，還有我們參考的 github 是使用 tensorflow 1 版本來撰寫，這使得前幾次作業所使用的 python3.8 不支援，當然我們有嘗試使用 `import tensorflow.compat.v1 as tf` 等程式碼來修正，但發現有些函式庫還是無法正常轉換，所以我們改用 python3.7 版本來進行訓練，然而 PyTorch 和 TensorFlow 的關鍵差異是它們執行代碼的方式，相同的是兩個框架都基於基礎數據類型張量 (tensor) 來工作的。分布式訓練 PyTorch 和 TensorFlow 的一個主要差異特點是數據並行化。PyTorch 優化性能的方式是利用 Python 對異步執行的本地支持。而用 TensorFlow 時，我們必須手動編寫代碼，並微調要在特定設備上運行的每個操作，以實現分布式訓練。

6. Reference

- K. Shuster, D. Ju, S. Roller, E. Dinan, Y. Boureau, and J. Weston. 2020. [The Dialogue Decathlon: Open-Domain Knowledge and Image Grounded Conversational Agents](#), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2453–2470. Association for Computational Linguistics. *arXiv preprint arXiv:1911.03768*.
- E. Hosseini-Asl, B. McCann, C. S. Wu, S. Yavuz, and R. Socher. 2020. [A Simple Language Model for Task-Oriented Dialogue](#). *arXiv preprint arXiv:2005.00796*.
- K. Sun, S. Moon, P. Crook, S. Roller, B. Silvert, B. Liu, Z. Wang, H. Liu, E. Cho, and C. Cardie. 2021. [Adding Chit-Chat to Enhance Task-Oriented Dialogues](#), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1570–1583. Association for Computational Linguistics. DOI:10.18653/v1/2021.naacl-main.124
- P. Xu, and Q. Hu. 2018. [An End-to-end Approach for Handling Unknown Slot Values in Dialogue State Tracking](#), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1448–1457. Association for Computational Linguistics. DOI:10.18653/v1/P18-1134
- P. Shah, D. Hakkani-Tur, G. Tur, A. Rastogi, A. Bapna, N. Nayak, and L. Heck. 2018. [Building a Conversational Agent Overnight with Dialogue Self-Play](#). *arXiv preprint arXiv: 1801.04871*.
- G. L. Chao, and I. Lane. 2019. BERT-DST: [Scalable End-to-End Dialogue State Tracking with Bidirectional Encoder Representations from Transformer](#). *arXiv preprint arXiv: 1907.03040*.