

使用 Simple Transformers 建立多元情緒分類模型

楊德倫

資策會數位教育研究所數位人才培育中心講師

摘要

Transformers 是一種深度學習的神經網路模型架構，主要用於自然語言處理（NLP，Natural Language Processing），完成文本分類、資訊擷取、問答、摘要、翻譯、文本生成等任務，提供 BERT、GPT-2、RoBERTa、T5 等支援 100 種以上語系的預訓練模型，讓開發者能夠直接使用，或透過微調（fine-tune）來進行客製化。

Simple Transformers 是基於 Transformers 的一種函式庫，可以讓使用者快速地訓練出自定義的語言模型。本文將使用 Simple Transformers 來建立一個多元（multi-class）情緒分類模型，協助開發者完成未知語料的情緒分類任務。

前言

過去在處理序列（Sequence）型態的資料（如文字、圖片、影音等）上，最常見的是遞歸神經網路（RNN，Recurrent Neural Network）架構，按照序列的順序來處理資料，例如常見的 LSTM、GRU 等變化型態，然而 RNN 架構在處理距離過長的序列資料，可能會有系統資源耗費的問題，此時需要一個具有平行處理序列資料的神經網路架構，於是 Transformers 在 2017 年橫空出世，在處理序列資料的效能上，有了大幅的提升與改變。

為了讓開發者能夠使用更簡便的語法來進行訓練，阿姆斯特丹大學（University of Amsterdam）的學生 Thilina Rajapakse 對 Transformers API 進行封裝，開發了一個容易上手的函式庫 Simple Transformers，在 Transformers 的使用上，語法簡單卻很有力，程式碼結構一致卻富有彈性，對初學者也很友善。

本文中，我們將整合 NTCIR 提供的中文情緒對話生成（CECG，Chinese Emotional Conversation Generation）語料，該語料提供了對話、回話文字與相對應的情緒標記，適合幫助我們進行訓練，建立客製化的情緒分類語言模型。

中文情緒對話生成（CECG，Chinese Emotional Conversation Generation）

NTCIR 第 14 屆的短文對話任務（STC-3，Short Text Conversation 3），是資訊檢索評估任務之一，其中的子任務「中文情緒對話生成」（CECG，Chinese Emotional Conversation Generation）提供了 2019 年 600,000 組與 2017 年

1,100,00 組有情緒標記的單輪對話之訓練資料集，給大家使用。原先的資料集使用簡體中文，可以考慮使用 OpenCC (Open Chinese Convert) 將其轉換成正體中文。

```
[
  [
    ["明天又要去旅行了, 超開心!", "5"],
    ["你又去哪兒啊?", "3"]
  ],
  [
    ["今天終於睡的飽飽的了.....哈哈", "5"],
    ["睡睡睡。。就知道睡", "0"]
  ]
]
```

圖 1 CECG 資料集的部分內容

如圖 1 所示，每一組單輪對話間，分為 [post, label] 與 [reply, label]。post 指的是使用者輸入的發話，reply 是指對 post 所作出的回應，label 是對應 post 或 reply 的情緒標記，值域在 0 到 5 之間。0 是指其它 (Other)，1 是指喜歡 (Like)，2 是指悲傷 (Sadness)，3 是指噁心、厭惡 (Disgust)，4 是指憤怒 (Anger)，5 是指開心、幸福、高興 (Happiness)，在圖 2 中可以清楚了解資料集的結構。



圖 2 CECG 資料集的結構

系統環境建置

本文測試環境使用 Python 3.8.8 版本，作業系統是 Windows 10，支援 GPU，NVIDIA (R) Cuda compiler driver 版本為 v11.0，所以需要安裝支援 GPU 的 PyTorch 相關套件：

安裝套件 (有 GPU)

<pre>pip install torch==1.7.1+cu110 torchvision==0.8.2+cu110 torchaudio==0.7.2 -f https://download.pytorch.org/whl/torch_stable.html</pre>
--

若是電腦沒有 GPU，則可以直接安裝 CPU 版本的 PyTorch 相關套件，缺點是訓練需要花費非常久的時間：

安裝套件 (無 GPU)

<pre>pip install torch==1.7.1+cpu torchvision==0.8.2+cpu torchaudio==0.7.2 -f https://download.pytorch.org/whl/torch_stable.html</pre>
--

最後別忘記安裝 Simple Transformers 套件（官方網站也有提供 conda 的安裝方法，可自行前往參考）：

安裝套件

<pre>pip install simpletransformers</pre>

語料下載與轉換

可至 [NTCIR-14 Short Text Conversation Task \(STC-3\)](#) 的網站下載 CECG 資料集，或是到這裡[下載](#)（連結可能隨時失效）。之後轉換格式的資料集，會以整合 2019 與 2017 年共 170 萬組單輪對話（340 萬句符合 [text, label] 格式的資料），檔案名稱為 `stc3_cecg_2017_and_2019_170w.json`，讀者可自行定義：

```
() stc3_cecg_2017_and_2019_170w.json
```

```
1 [[["現在刷朋友圈最大的快樂就是看代購們各種直播。。。", "5"], ["臥槽我也是", "4"]], [{"什麼時候可以一覺到天亮~", "0"}, [{"可憐}加油! ~~, "1"]], [{"1200 差點永遠睡在跑道上阿 [淚]", "2"}, [{"為什麼跑那麼多? 會死人的! ", "4"]], [{"各種不爽。。。", "3"}, [{"咋了??", "3"}], [{"不活成自己想活的樣子, 就不算活著吧。", "3"}, [{"存在即是一切, 一切為了存在 [壞笑]", "0"}], [{"香頰的芒果慕斯不錯", "1"}, [{"呢家希臘餐廳, 求具體地址呀, 找不到嘍。 [淚] [淚]", "2"}], [{"倫敦奧運撐杆跳的杆在跳的過程斷了 [汗] 還好運動員沒受傷", "5"}, [{"。。。斷成三大節了。", "0"}], [{"我收回我之前那句卸妝水廢雞卸妝油好用的話", "2"}, [{"你幹了什麼", "4"}], [{"遍地是冰, 好糾結", "3"}, [{"那更不該一樣了。我看的都是小包包跟襪子的是的昨天想一個都沒想出來 4點才睡", "3"}], [{"不愁吃, 不愁穿, 不愁住, 不愁行, 還愁啥呢?", "3"}, [{"愁個老婆 [嘻嘻]", "5"}], [{"哇哇哇...向利和包場誰電影~爽~ [哈哈]", "5"}, [{"你能再奮鬥點?", "3"}], [{"出差報銷全都到賬啦~ [歡笑] [樂樂] 要去還信用卡了 [淚] [生病]", "5"}, [{"有木有老王的, 有木有?", "4"}], [{"好想回到沉迷書海的年代...", "2"}, [{"現在靜得下心看一小時書已屬難得", "5"}], [{"[抓狂] [抓狂] [抓狂] [抓狂] [抓狂] [抓狂]", "2"}, [{"腫麼這麼早啊上班麼 [偷笑]", "5"}], [{"今天看見ales了, 很紳士。", "5"}, [{"哪位啊", "3"}], [{"你美了 [嘻嘻] [離嘴] [離嘴]", "5"}, [{"really? 好誘惑, 抽時間一定要去 [離嘴]", "1"}], [{"六點幾集合, 點個名解散, 七點幾再集合開飯, 好玩咩 [惡罵]", "3"}, [{"好過3點幾集合啊", "3"}], [{"為何每個同學剪完前劉海 [碎發] 都那麼傻逼", "3"}, [{"是你, 好不好?", "3"}], [{"死扛吧, 能扛多久算多久", "3"}, [{"5555555555555555", "2"}], [{"賈玲和翟穎的動力火車我笑吐血了", "5"}, [{"這期我也笑抽了。。。迪克牛仔啊", "5"}], [{"齊啊! 幫別人下東西, 還把人家mp3搞壞了。。。 (t ^ t)", "3"}, [{"那要賠人家啊!", "4"}], [{"不急才怪!", "3"}, [{"媽逼的你是有事才得聯繫我。 明天我來貴陽", "4"}], [{"收拾行囊", "0"}, [{"拜拜", "0"}], [{"在眉州東坡長虹橋店, 陪老公參加同學會 [兔子]", "0"}, [{"你在我家門口~, "1"}], [{"週五啦 [嘻嘻] [奧特曼] [鼓掌]", "5"}, [{"嘿嘿, 你換小米了嗎?", "2"}], [{"停水了就不吃冰箱裏放好幾天的蘑菇了。煮點白粥算了。", "3"}, [{"這是在大連租房的時候吧, 俺猜滴對不 [思考]", "1"}], [{"最近事好多", "0"}, [{"忙並快樂著。。。", "1"}], [{"鄺視 ab 的服務員 [鄺視] [鄺視] [可憐] [可憐]", "2"}, [{"你開到裏子?", "0"}], [{"手機散熱不太理想, 才玩了一個小時手掌就感覺到熱了", "3"}, [{"g 幾?", "0"}], [{"早上好, 我奔四的第一天", "5"}, [{"啊啊啊, 不是十一月七號嗎??? 生日快樂!!!!!!", "4"}], [{"mubamboo 好難哦。。。 [失望]", "2"}, [{"嘻嘻] 開發的人就在樓下", "5"}], [{"受不鳥了。。。打一晚上海上飽了; [抓狂]; [抓狂]; [抓狂]", "3"}, [{"衰] 我都餓不行了。。。", "2"}], [{"媽的這空調漏水就像尿不禁一樣, 真想給它塞片前列康", "3"}, [{"憑啥這空調是公的!?", "3"}], [{"如果有錢的話, 我會選擇做個未婚爸爸。", "0"}, [{"你爹都催婚了 還使咩做未婚爸爸 [懶得理你]", "3"}], [{"直接給我一把刀算了 鍵盤這種時候鬧脾氣 想買張機票回國了", "0"}, [{"外接鍵盤肯定有賣的阿, 不行就去 it, 加油", "1"}], [{"明天去三中千人報告廳開家長會, 各種無奈", "2"}, [{"哈哈? 特邀嘉賓?", "0"}], [{"好天氣! 想去爬爬山~ 呼吸下新鮮空氣! 清涼腦袋! [可愛]", "1"}, [{"去爬山沒有?", "0"}], [{"今天大暴雨。。。 [抓狂]", "2"}, [{"剛下車就大暴雨啊!! 有傘也沒用啊!!!!", "4"}], [{"突然好想吃麥旋風!!!!", "1"}, [{"淚] 我也想吃, 吃奧利奧吧, 自己弄", "2"}], [{"個個都話又濃又好味! 不錯吧! [離嘴] [離嘴]", "1"}, [{"難道系你落廚的? [思考] [偷笑]", "5"}], [{"第一夫人頒獎給主旋律影片 [話筒] 恭喜阿狗! [話筒]", "5"}, [{"她說得挺好的", "1"}], [{"井柏然真的是華誼的?", "0"}, [{"微笑] 可能抱來的", "0"}], [{"我唯一的希望剩下作文了操。語文試卷我表示無力吐槽", "2"}, [{"淚] 一頁? 我還一本啊擦!", "2"}], [{"導師對論文還是不滿意, 我給自己點根 [蠟燭]", "3"}, [{"悲傷] 那怎麼辦", "2"}], [{"蛇足在確認信息中", "0"}, [{"謝謝! 加油!", "1"}], [{"平安夜 [攤手] 把眉毛修缺了一個口子 [攤手] 心疼自己", "2"}, [{"不能信任你的修眉技術了?", "0"}], [{"準備睡覺。。。 = 大家晚安★, "5"}, [{"晚安!", "5"}], [{"2011 年最後一天領到新筆記本電腦 x220i [鼓掌] [偷笑] [嘻嘻]", "5"}, [{"靠, 沒有我的。。。", "2"}], [{"久違的早晨... 夢想不遠!", "0"}, [{"早, 至少比我早...", "2"}], [{"我想我大概什麼都做不了 只能追劇和睡覺了 qaq ~, "2"}, [{"你不是減肥了麼", "3"}], [{"會有人選擇性失明。也會有人選擇性失聰失語咯", "0"}, [{"你在哪裏", "0"}], [{"怎麼就是不喜歡周筆暢呢?!", "3"}, [{"我也不喜歡。。。", "1"}], [{"這麼老套的劇情卻把人哭成什麼一樣", "4"}, [{"我說的沒錯吧 狗血的韓劇劇情", "4"}], [{"每天九點醒, 這是考研該有的狀態麼?", "3"}, [{"這是遠離中傳的節奏", "0"}], [{"見到賴仔咿。。。好耐唔見到過賴仔了。。。激動中 (心)", "5"}, [{"阿~~~有無有!", "4"}], [{"啊 好想盡細方面 [高聲] [好困] [互相拍]
```

圖 3 將 2019 與 2017 年 CECG 資料集整合後的檔案內容

接下來，需要將資料集轉換成 Simple Transformers 訓練 multi-class classification 時所要求的格式。以下分享簡單的程式碼，有經驗的開發者可自由發揮和修改，剛入門的朋友可以考慮直接使用：

程式碼 cecg2data.py

```
import sys, json, time, argparse

parser = argparse.ArgumentParser(description='建立訓練資料')
parser.add_argument('--data', help='CECG 資料集 (JSON 格式)')
parser.add_argument('--save', help='將訓練資料儲存到哪裡 (JSON 格式)')
args = parser.parse_args()

# 讀取 CECG 資料集，整合成訓練格式
def getData():
    # 放置每一組 CECG 對話的變數
    listCECG = []

    # 讀取 JSON 文字結構
    with open(args.data, 'r', encoding="utf8") as file:
        ...

    從 listJSON 取得每一組單論對話 listDoc，
```

其中 `listDoc` 元素有兩個 `list`，分別是：

=> ["現在 刷 朋友 圈 最大 的 快樂 就是 看 代 購 們 各種 直播 。 。 。 。 。", "5"]

=> ["臥 槽 我 也 是", "4"]

...

將 JSON 轉成 list

`listJson = json.loads(file.read())`

走訪每一組對話

`for listDoc in listJson:`

 # 整合所有對話

`listCECG += listDoc`

回傳符合訓練格式的對話資料

`return listCECG`

將 CECG 語料，儲存成 [text, label] 的 JSON 格式

`def saveData(listData):`

`with open(args.save, 'w', encoding='utf-8') as file:`

`file.write(json.dumps(listData, ensure_ascii=False))`

主程式

`try:`

`tStart = time.time() #計時開始`

`saveData(getData())`

`tEnd = time.time() #計時結束`

`print(f"執行花費 {tEnd - tStart} 秒。")`

`except:`

`print("Unexpected error: ", sys.exc_info())`

指令

```
python cecg2train.py --data=stc3_cecg_2017_and_2019_170w.json --save=train.json
```



```
train.json
1  [
2    [
3      "現在 刷 朋友 圈 最大 的 快樂 就是 看 代購 們 各種 直播 。 。 。 。",
4      "5"
5    ],
6    [
7      "臥 槽 我 也 是",
8      "4"
9    ],
10   [
11     "什麼 時候 可以 一 覺 到 天亮 、",
12     "0"
13   ],
14   [
15     "[ 可憐 ] 加油 ! ~ ~",
16     "1"
17   ],
18   [
19     "1200  差點 永遠 睡 在 跑道 上 阿  [ 淚 ]",
20     "2"
21   ],
22   [
23     "爲什麼 跑 那麼 多 ? 會 死 人 的 ! !",
24     "4"
25   ],
26   [
27     "各種 不 爽 。 。 。",
28     "3"
29   ],
30   [
31     "咋 了 ? ? ",
32     "3"
33   ],
34   [
35     "不 活 成 自 己 想 活 的 樣 子 , 就 不 算 活 着 吧 。",
36     "3"
37   ],
38 ]
```

圖 4 將 CECG 資料集轉換成 train.json，其排版後的預覽結果

訓練情緒分類語言模型

接下來，即將開始訓練模型。在模型參數中，預訓練模型選擇 `bert`，`bert` 在這裡是 `simple transformers` 的代碼；`bert-base-chinese` 是指定支援中文的預訓練模型，其它常見的預訓練模型，可以參考 [Hugging Face](#) 網站，或是 `Simple Transformers` 的 [Classification Specifics](#) 頁面；`train_batch_size` 是指每次放入神經網路訓練的樣本數，其值愈高，佔用 GPU 記憶體愈多，模型收斂更快、學得更好（請參考「梯度下降」），但有可能會發生過度擬合（`overfitting`）的問題，變成一定要是剛好或是非常接近原始資料，才會被正確預測出來，這會造成模型泛化能力不足，無法對未知的資料進行準確的預測；有些人會使用 2 的冪來設定 `train_batch_size` 的值，例如 1（很少）、2、4、8、16、32、64、128…等，也

有人會嘗試出一個特定的值（無論奇數或偶數），把 GPU 記憶體用到剛好又不會拋出錯誤，一切視需求調整；`num_train_epochs` 是訓練回合數，每一回合都會將原先的資料重新隨機排序（shuffle）後，再切分成數個 batches（1 個 batch 的樣本數量，就是 batch size），重新放到神經網路訓練。

隨著訓練資料集大小、文句長度等不同，設定上都會有所不同，學習率（learning rate）也會有影響，但這個不在我們討論範圍，讀者可上網查閱，並視情況調整。可以參考下列的訓練程式碼，依需求修改或增刪：

程式碼 train.py

```
from simpletransformers.classification import ClassificationModel,
ClassificationArgs
import pandas as pd
import json, time, argparse

# 帶入自訂的執行參數
parser = argparse.ArgumentParser(description='訓練情緒分類語言模型')
parser.add_argument('--data', help='訓練資料')
args = parser.parse_args()

# 讀取訓練資料
def getDataFrame():
    # 讀取 JSON 文字結構
    with open(args.data, 'r', encoding="utf8") as file:
        listJson = json.loads(file.read()) # 將 JSON 轉成陣列

    # 將訓練資料轉成 panda DataFrame，並提供 headers
    df = pd.DataFrame(listJson)
    df.columns = ["text", "labels"]

    # 將 labels 欄位的資料，轉成數值，才能完整符合訓練格式
    df["labels"] = pd.to_numeric(df["labels"])

    # 回傳 DataFrame
    return df

# 訓練模型
def train(df):
    # 輸出語言模型的目錄名稱
```

```

dir_name = 'bert-base-chinese-bs-64-epo-3'

# 自訂參數
model_args = ClassificationArgs()
model_args.train_batch_size = 64
model_args.num_train_epochs = 3
model_args.output_dir = f"outputs/{dir_name}"

# 建立 ClassificationModel (會自動下載預訓練模型)
'''
如果僅有 CPU，可以將以下參數調整成 use_cuda=False
'''
model = ClassificationModel(
    'bert', # 選擇 bert (simple transformers 模型代碼)
    'bert-base-chinese', # 支援中文的 bert 預訓練模型
    use_cuda=True, # 啟用 GPU
    num_labels=6, # multi-class 有 6 類，所以寫 6
    args=model_args # 帶入自訂參數
)

# 訓練 model，會將
model.train_model(df)

# 主程式
if __name__ == "__main__":
    tStart = time.time() # 計時開始
    train( getDataFrame() )
    tEnd = time.time() # 計時結束

    # 輸出程式執行的時間
    print(f"執行花費 {tEnd - tStart} 秒。")

```

指令

```
python train.py --data=train.json
```

Epochs 0/3. Running Loss: 0.8043: 2% | 1033/53727 [09:55<12:04:41, 1.21it/s]

圖 5 訓練進度，在這裡會訓練 3 回合

提醒

拋出錯誤

RuntimeError: CUDA out of memory.

通常是 `train_batch_size` 設定太高了，造成 GPU 記憶體不足；試著降低數值，如果是以 2 的冪來調整，那就試著向下調整，一次不夠就再降，降到可以正常執行；若是模型預測成效不佳，就提升 `num_train_epochs`，讓神經網路多學幾回，再回頭評估成效。

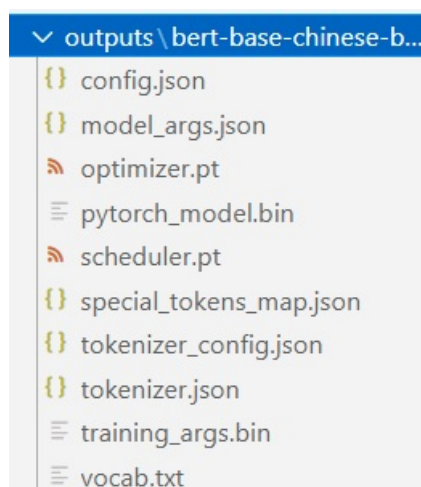


圖 6 輸出模型會儲存在 outputs 當中的 bert-base-chinese-bs-64-epo-3 資料夾

預測文字的情緒

完成了語言模型的訓練後，準備開始進行預測，預測的回傳結果會是一個 list，代表可以一次預測一個以上的句子，對多個句子進行預測時，回傳的順序也是各個句子的情緒分類結果。有一個需要注意的地方，就是預測時的模型參數設定，通常與訓練時的模型參數設定一致：

程式碼 predict.py

```
from simpletransformers.classification import ClassificationModel,
ClassificationArgs
import time

# 預測情緒
def predict(listTestData):
```

```
# 輸出模型存在的目錄名稱
dir_name = 'bert-base-chinese-bs-64-epo-3'

# 自訂參數
model_args = ClassificationArgs()
model_args.train_batch_size = 64
model_args.num_train_epochs = 3

# 讀取 ClassificationModel
model = ClassificationModel(
    'bert',
    f"outputs/{dir_name}", # 這裡要改成訓練完成的模型資料夾路徑
    use_cuda=True,
    cuda_device=0,
    num_labels=6,
    args=model_args
)

# 預測
predictions, raw_outputs = model.predict(listTestData)

# 回傳預測結果，會是一個 list
return predictions

# 主程式
if __name__ == "__main__":
    # 計時開始
    tStart = time.time()

    # 準備預測情緒類別，資料可以不只 1 句！
    listTestData = [
        "現在 刷 朋友 圈 最大 的 快樂 就是 看 代購 們 各種 直播 。 。 。 。 。",
        "你 幹 了 什麼",
        "不 愁 吃 ， 不 愁 穿 ， 不 愁 住 ， 不 愁 行 ， 還 愁 啥 呢 ?"
    ]

    # 進行預測
    print( predict(listTestData) )
```

```
# 計時結束
tEnd = time.time()

# 輸出程式執行的時間
print(f"執行花費 {tEnd - tStart} 秒。")
```

指令

```
python predict.py
```

將自訂的「現在 刷 朋友 圈 最大 的 快樂 就是 看 代購 們 各種 直播 。 。 。 。」、「你 幹 了 什麼」、「不 愁 吃，不 愁 穿，不 愁 住，不 愁 行，還 愁 啥 呢？」三句話進行預測，會得到 [5, 4, 3] 的輸出結果，皆為情緒標記的值，5 代表 happiness，4 代表 anger，3 代表 disgust。表 1 提供了文句的預測結果，大家覺得如何呢？

索引	文句	預測情緒標記	標記意義
0	現在 刷 朋友 圈 最大 的 快樂 就是 看 代購 們 各種 直播 。 。 。 。	5	happiness
1	你 幹 了 什麼	4	anger
2	不 愁 吃，不 愁 穿，不 愁 住，不 愁 行，還 愁 啥 呢？	3	disgust

表 1 文句的預測結果

後記

本文使用了 Simple Transformers 取得預訓練模型，提供了訓練與預測的方法，期許協助開發者或使用者完成下游任務，進一步檢視未分類的文句，提供合適的情緒標記，然而文章尚有不足之處，例如資料集尚未去除重複或無意義對話，以及尚未進行模型成效評估等，篇幅有限，無法詳細說明，讀者在訓練自訂的情緒分類模型時，務必進行完善的資料預處理，以及評估訓練完畢後的語言模型，相信對下游的分類任務上，會有正面、實質的幫助。

參考資料

[1] 自然語言處理

<https://zh.wikipedia.org/zh-hant/%E8%87%AA%E7%84%B6%E8%AF%AD%E8%A8%80%E5%A4%84%E7%90%86>



[2] Simple Transformers

<https://simpletransformers.ai/>

[3] 範例程式碼

https://github.com/telunyang/python_multiclass_classification

[4] Hugging Face

<https://huggingface.co/>

[5] Multiclass, Multilabel 以及 Multitask 的區別

<https://cynthiachuang.github.io/Difference-between-Multiclass-Multilabel-and-Multitask-Problem/>

[6] Epoch, Batch size, Iteration, Learning Rate

<https://medium.com/%E4%BA%BA%E5%B7%A5%E6%99%BA%E6%85%A7-%E5%80%92%E5%BA%95%E6%9C%89%E5%A4%9A%E6%99%BA%E6%85%A7/epoch-batch-size-iteration-learning-rate-b62bf6334c49>

[7] transformers

<https://github.com/huggingface/transformers>

[8] NTCIR-14 Short Text Conversation Task (STC-3)

<http://sakailab.com/ntcir14stc3/>