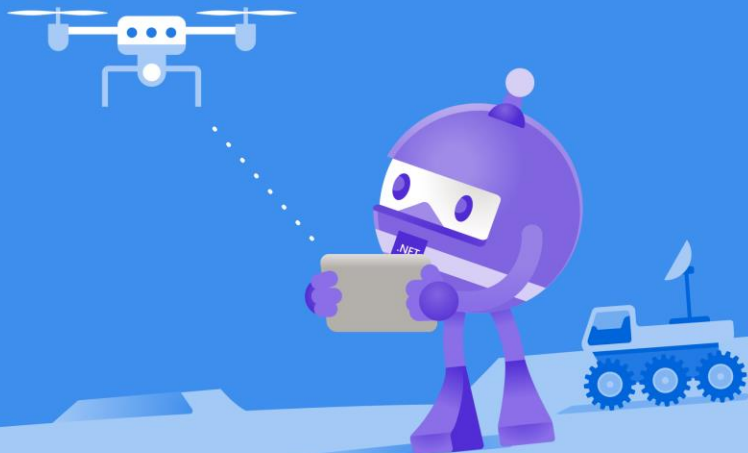


.NET Conf


探索 .NET 新世界



打造自己的繁體中文LUIS服務

尹相志 AllanYiin



A close-up shot of Sophia the Robot, a humanoid robot with a realistic female face, wearing a dark suit. The background is blurred, showing what appears to be a stage or event space with blue lighting.

很抱歉，這樣的技術並不存在於地球，只存在於幻想

**We talked to Sophia —
the first-ever robot citizen**

LUIS

Music

→ Publish

Intents ⊕

None

Entities ⊕

No entities added

Pre-built Entities ⊕

No pre-built entities added

Regex Features ⊕

No patterns added

Phrase List Features ⊕

No phrase list features added

New utterances

Please, enter an utterance. →

Search

Suggest

Review labels

Performance analysis

Label some utterances and click "train", and then I can predict likely errors for your intents and entities.

↺ Train

Microsoft

LUIS (Language Understanding)
意圖識別
實體識別

直接用API不就好了嗎？

當然，調用成本、內容可控以及產品差異性壁壘是更重要的考量。

支援的語言

LUIS 可理解下列語言的語句：

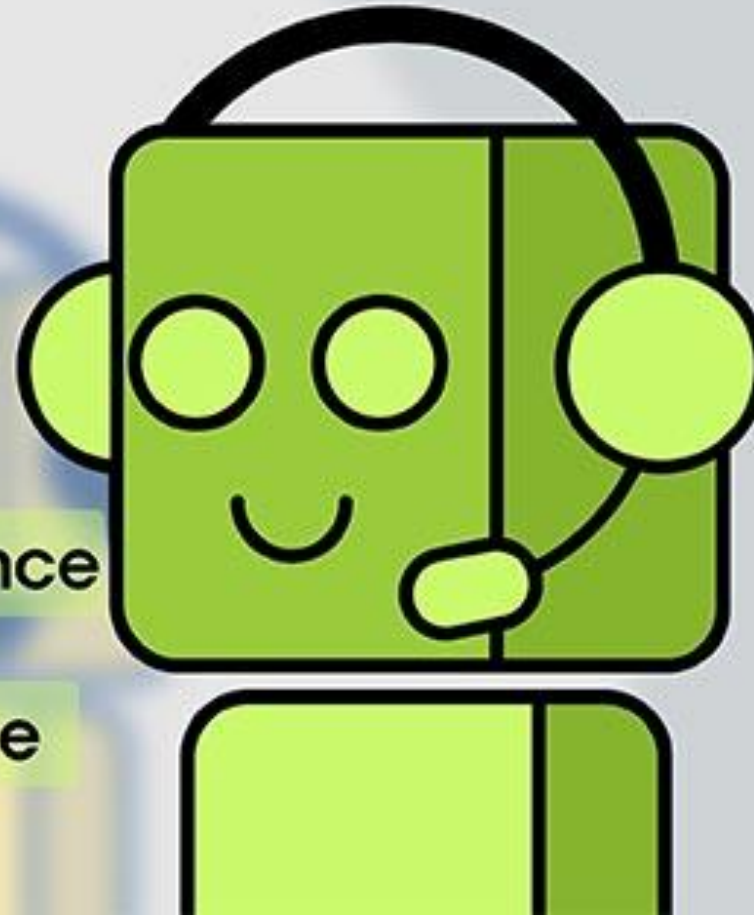
Language	地區設定	預建網域	預建實體	片語清單建議	**文字分析 (情感和 關鍵字)
英文 (美國)	en-US	✓	✓	✓	✓
阿拉伯文 (preview-新式 標準阿拉伯文)	ar-AR	-	-	-	-
*中文	zh-CN	✓	✓	✓	-
荷蘭文	nl-NL	✓	-	-	✓
法文 (法國)	fr-FR	✓	✓	✓	✓

模型限制

如果您的應用程式超過 LUIS 模型限制，請考慮使用 [LUIS 分派](#) 應用程式或使用 [LUIS 容器](#)。

區域	限制
應用程式名稱	*預設字元上限
應用程式	每個 Azure 撰寫資源500個應用程式
批次測試	10 個資料集，每個資料集 1000 個語句
明確清單	每個應用程式 50 個
外部實體	無限制
對應方式	每個應用程式500：499個自訂意圖，以及所需的 <i>None</i> 意圖。 分派 應用程式有對應的500分派來源。
列出實體	父系：50 個項目，子系：20,000 個項目。正式名稱為*預設字元上限。同義值沒有長度

Artificial Intelligence for Customer Service

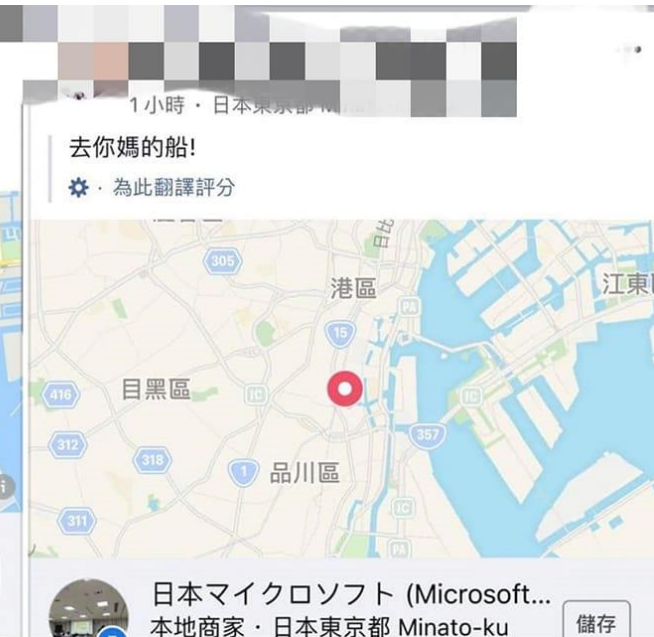
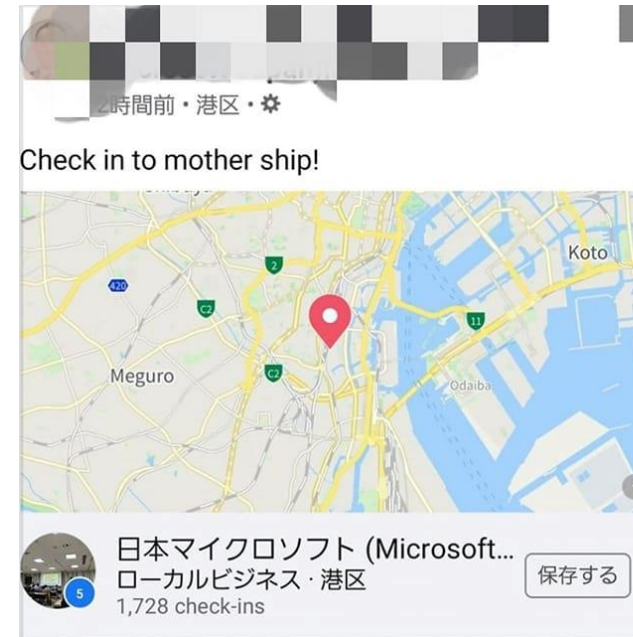


智能客服的核心意義不在於替代客服人員，而是希望透過分擔掉重複性高的簡單問題，來釋放出客服人員的時間與腦力，來解決更重要更需要人的溫暖才能處理的問題。

也因此考核的標準不是整體正確率，而是限制範圍內的意圖正確率，而且當客戶問了不在意圖清單內的問題，能夠正確的轉人工客服。

深度學習的核心在最佳化 雖然努力的在為機器找尋更好的評估指標 但語言的複雜度永遠是難以掌握的

找到约 56,600,000 条结果 (用时 0.30 秒)



中文的恐怖之處

窮舉法對於中文來說 是無效的



trident

`pip install tridentx`
<https://github.com/AllanYiin/trident>

`Pytorch>=1.4`
`Tensorflow>=2.2`

不用可惱於該學
tensorflow還是pytorch

意圖識別就是將輸入對話轉換為n選一的選擇題

意圖識別

你好，我明天出國，希望能夠
提高信用卡額度

了解，我來為您查詢..

Artificial Intelligence
for
Customer Service

把問答題變成選擇題


打開app應用
詢問公車路線
數學計算
閒聊
詢問電影場次
查詢聯絡人
查詢食譜
詢問日期
電子郵件指令
詢問電視節目
詢問航班
詢問健康資訊
詢問樂透
詢問地圖
詢問足球賽事










簡訊指令
音樂指令
詢問新聞
詢問小說
詢問詩詞
詢問廣播節目
詢問謎語
鬧鐘時程設定
詢問股票
電話指令
詢問火車班次
詢問翻譯
切換電視頻道
影片播放指令
詢問天氣
開啟網站

本次範例有31
個意圖類別

whats your intent_pytorch

File Edit View Insert Cell Kernel Widgets Help








 Run
 


 Code

Logout

Trusted

Python 3

```
In [1]: %matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
from IPython import display
```

洞悉你的意圖(pytorch版)

意圖識別是機器與人透過語言交互的重要關鍵，因為發出命令者不一定會用固定的格式發言，因此機器必須要能夠根據句子內容來判斷發言者的實際意圖。本次數據集包含31種意圖(一個句子僅符合單一意圖)，接下來我們將介紹如何利用卷積神經網路進行自動意圖識別。

```
In [2]: import codecs
import numpy as np
import random
import pickle
import os
import math
os.environ['TRIDENT_BACKEND'] = 'pytorch'
#!pip install tridentx --upgrade
import trident as T
from trident import *
```

trident 0.7.0

Using Pytorch backend.
Image Data Format: channels_first.
Image Channel Order: rgb.
Pytorch version:1.7.0+cu110.
Automatic Mixed Precision Support:True.
Using pillow image backend.

Pillow version:7.1.2

建模實作:使用Text
CNN建立意圖識別分類
模型

https://github.com/AllanYiin/DeepBelief_Course5_Examples/tree/master/epoch802_洞悉你的意圖

文字模型也需要數據增強

```

random.shuffle(train_data) #把訓練數據集順序打亂
random.shuffle(test_data) #把訓練數據集順序打亂
idx0=0
idx1=0
def data_augmentation(txt):
    txt=list(txt)
    #隨機對調字的順序
    if random.randint(0,10)<=3 and len(txt)>6:
        pos= random.randint(1,len(txt)-2)
        w1=txt[pos]
        w2=txt[pos+1]
        txt[pos]=w2
        txt[pos+1]=w1
    #隨機插入虛字
    if random.randint(0,100)%7<=3 and len(txt)>=6:
        wordlist=['的','了','若','可','然','也','或',' ',' ','。',' ',' ',' ']
        w1=random.choice(wordlist)
        txt.insert(random.randint(1,len(txt)-2), w1)
    #隨機插入標點
    if random.randint(0,100)%5<2:
        wordlist=[' ',' ','。',' ','? ','! ',' ',' ',' ']
        w1=random.choice(wordlist)
        txt.append(w1)
    return ''.join(txt)

```

我想聽蔡依林的新歌怪美的。

我想蔡聽依林的新歌怪美的。

我想聽蔡依林的新怪歌美的。

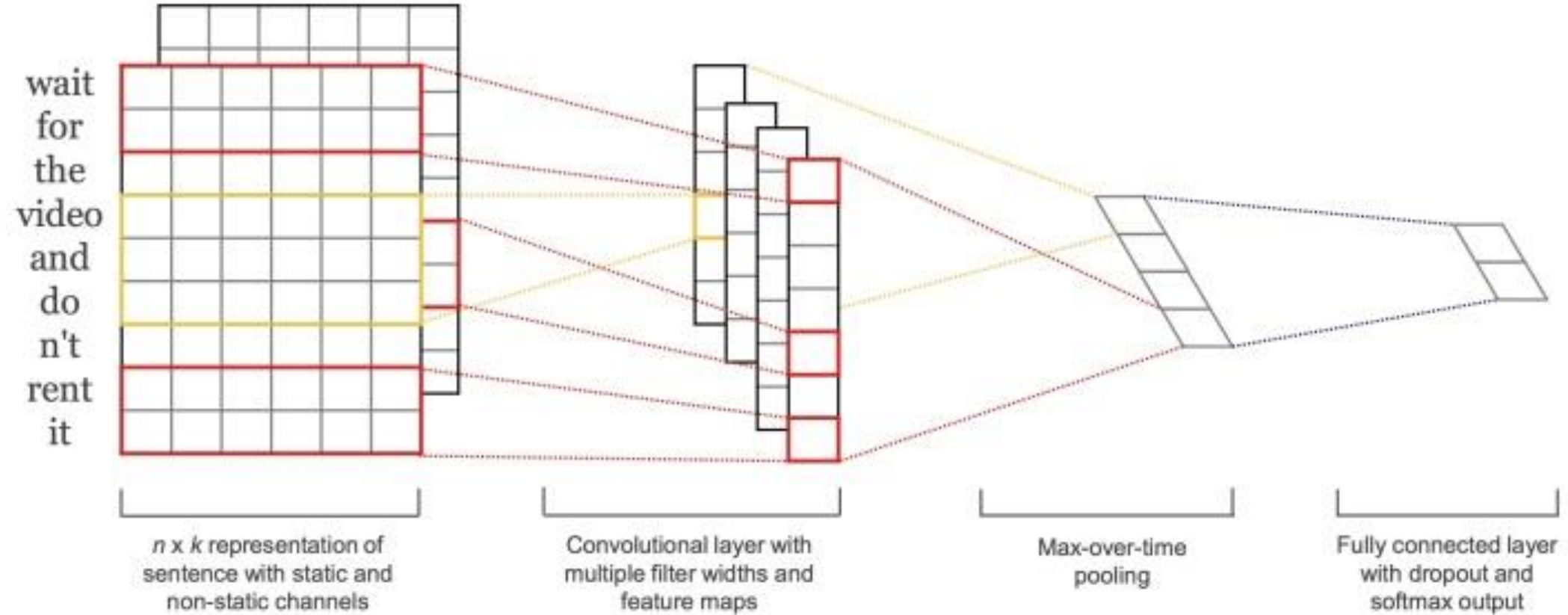
我想聽了蔡依林的新歌怪美的。

我想聽蔡依林的新的歌怪美的。

我想聽蔡依林，的新歌怪美的。

我想聽蔡依林的新歌怪美的。

如果這些句子不會影響你判斷語意，那麼
機器也應該不受影響



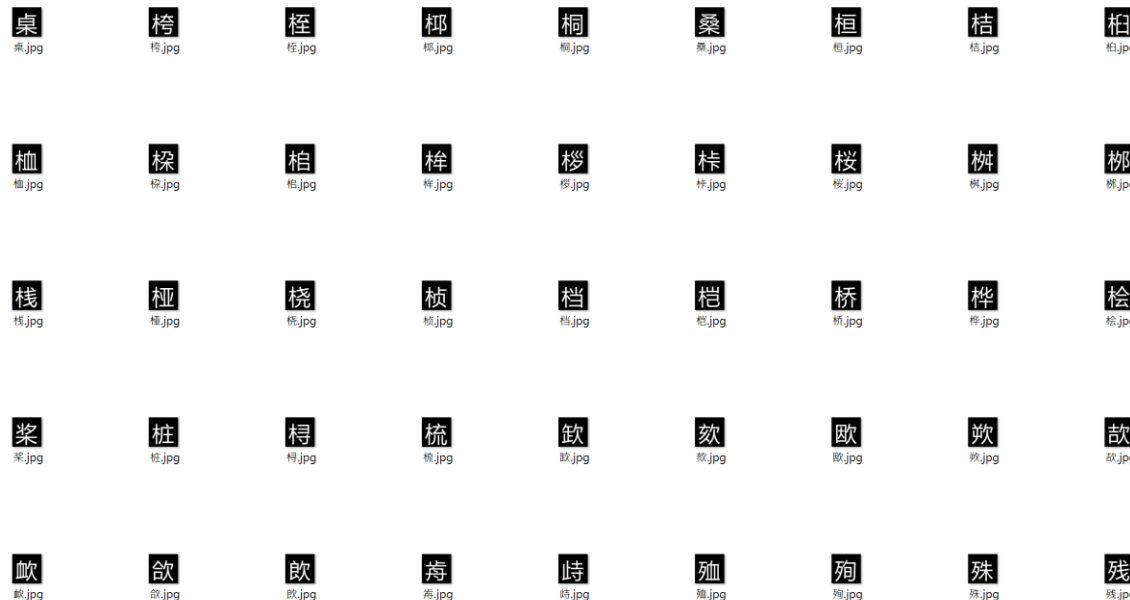
如何產生中文表徵?

~~Onehot~~ 太稀疏太大

~~詞向量~~ 得先分詞錯誤率疊加

~~Bert~~ 推論時間太長

字向量 快、沒有詞典外詞的問題



字向量是怎麼得到的??

字向量的原始輸入為64x64的中文字圖形
透過以下多任務訓練

1. 基於圖形猜部首(形)
2. 基於圖形猜讀音(音)
3. 基於圖形猜onehot
4. 字為基礎的n-gram(根據前後文字猜中間字)
5. 詞為基礎的n-gram(根據前後詞猜中間詞, 詞向量是詞中各字的字向量加總後除以字數)
6. 繁簡體以及異體字之間的字向量應該差異極小

最終可以將每個中文字編碼為256的特徵向量



輸入張量 (8,256, 128)

其中8是批次軸, 256是通道, 128是空間軸(字數, 不足128用零補滿)

我想聽蔡依林的新歌怪美的。

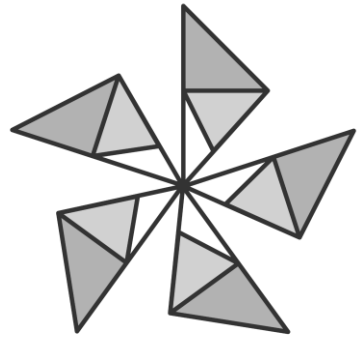
一維卷積處理n個字之間的語意特徵
(8,num_filters,128)

疊合Concatenate
(8,num_filters*n,128)

下採樣
downsample

GlobalAvgPool
+ SoftMax
(8,31)





ONNX RUNTIME

ONNX Runtime 又是什麼?

<https://github.com/microsoft/onnxruntime>

可以支持的語言

Python (3.5~3.7)

C

C# (.Net Standard>1.1)

C++

Ruby

可以支援以下加速機制

MLAS (Microsoft Linear Algebra Subprograms)

NVIDIA CUDA (**CUDA 10.0** and **cuDNN 7.6**)

Intel MKL-ML

Intel MKL-DNN - subgraph optimization

Intel nGraph

NVIDIA TensorRT

Intel OpenVINO





Nuphar Model Compiler

DirectML

ACL (in preview, for ARM Compute Library)

瀏覽 已安裝 更新

onnxruntime ☒ 包括搶鮮版

	Microsoft.ML.OnnxRuntime <input checked="" type="checkbox"/> 依 Microsoft, 135K 項下載 This package contains ONNX Runtime for .Net platforms	v1.0.0
 搶鮮版	Aiinfra.OnnxRuntime.Gpu 依 Microsoft, 702 項下載 This package contains ONNX Runtime for .Net platforms	v0.1.5-dev-ec8f5bb6
	Microsoft.ML.OnnxRuntime.Gpu <input checked="" type="checkbox"/> 依 Microsoft, 13.6K 項下載 This package contains ONNX Runtime for .Net platforms	v1.0.0
	Microsoft.ML.OnnxRuntime.MKLML <input checked="" type="checkbox"/> 依 Microsoft, 1.8K 項下載 This package contains ONNX Runtime for .Net platforms	v1.0.0

```
model 0 Step: 4s319ms | Loss: 0.08819 | accuracy: 98.171% | learning rate: 1.000e-03 | epoch: 95 ( 100/144 )
model 0 Step: 2s418ms | Loss: 0.08813 | accuracy: 98.173% | learning rate: 1.000e-03 | epoch: 95 ( 110/144 )
model 0 Step: 3s277ms | Loss: 0.08806 | accuracy: 98.174% | learning rate: 1.000e-03 | epoch: 95 ( 120/144 )
model 0 first layer gradients: 1.973e-02 | last layer gradients: 1.037e-02
model 0 Step: 2s486ms | Loss: 0.08800 | accuracy: 98.175% | learning rate: 1.000e-03 | epoch: 95 ( 130/144 )
```

In [56]:

```
model.eval()
model.save_onnx('Models/text_cnn.onnx')
```

我們已經將模型匯出成為onnx格式，要使用它來做推論的步驟其實非常簡單，首先要匯入onnxruntime的包，然後將模型載入。

In [57]:

```
import onnx
import onnxruntime as ort

#載入onnx模型進行檢查，不需要每次都做，只要轉檔後做一次即可
predictor = onnx.load('Models/text_cnn.onnx')
print(onnx.checker.check_model(predictor))
```

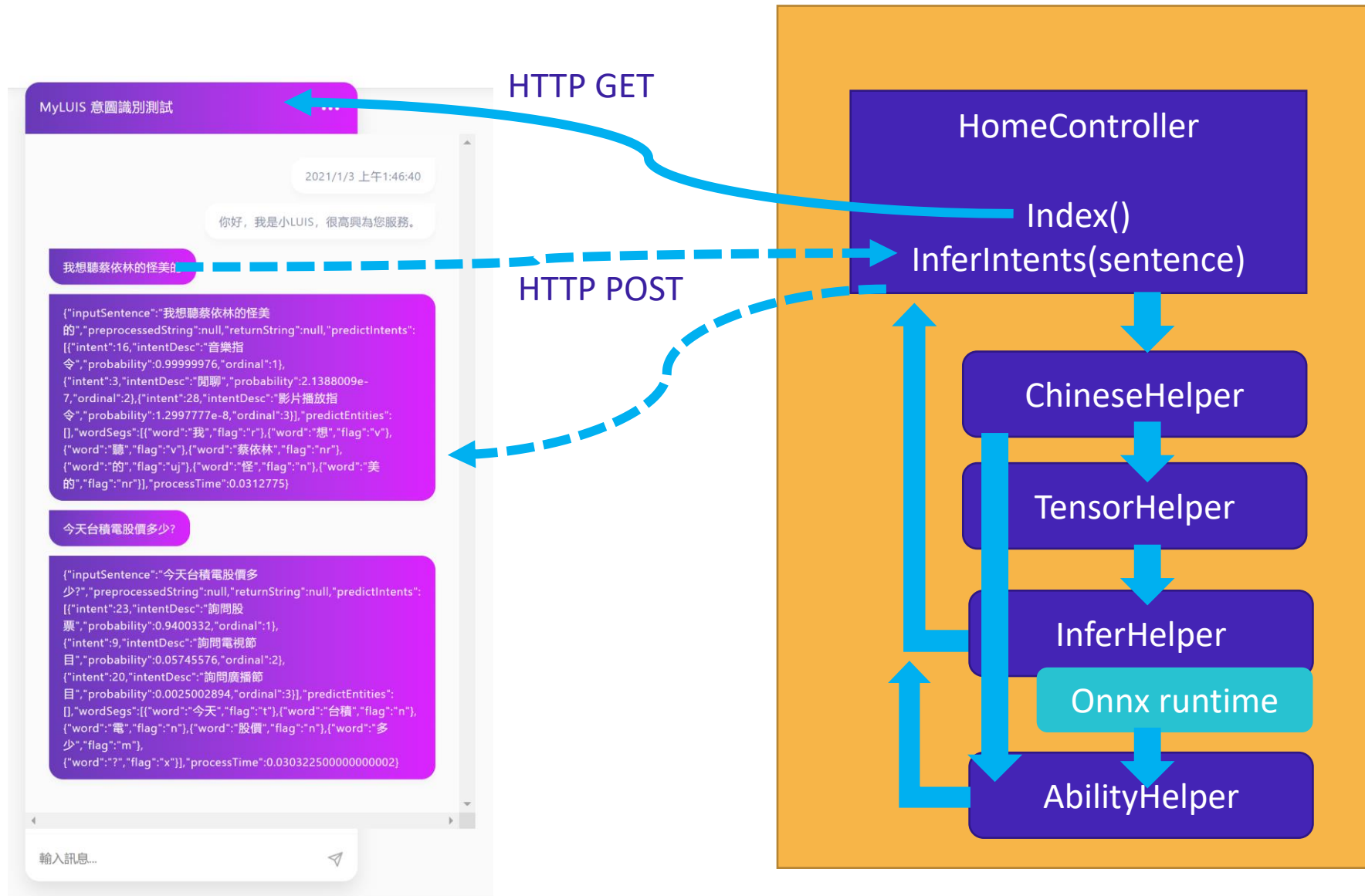
None

In [58]:

```
#載入onnx模型至onnxruntime
luis_session = ort.InferenceSession('Models/text_cnn.onnx')
#取出輸入張量形狀
input_name = luis_session.get_inputs()[0].name
input_name
```

'input'

在trident中如何處理模型轉onnx



繁簡互轉, 全半形互轉

將文字轉成字向量序列再
轉成onnx dense tensor

執行cpu推論, 選取前3
高機率意圖回傳

根據推論的意圖, 識別的實體(人名地名), 透過對應能力api將產生結果回傳



```

public static List<NamedOnnxValue> StringToEmbeddedTensor(string sentence)
{
    if (char_embedding == null)
    {
        char_embedding = np.Load<float[,]>(Properties.Resources.embed);
    }
    var container = new List<NamedOnnxValue>();
    NDArray features = np.zeros((256,128));
    for (int i = 0; i < sentence.Length; i++)
    {
        int idx = Char2Index(sentence[i]);
        features[string.Format("{0}", i)] = idx != -1 ? char_embedding[string.Format("{0}", idx)] : np.random.standard_normal(256);
    }
    features = features.astype(np.float32);
    features = features.flatten();
    var tensor = new DenseTensor<float>(features.ToArray<float>(), new int[] { 1,256,128 });
    var onnxvalue = NamedOnnxValue.CreateFromTensor<float>("input", tensor);
    container.Add(onnxvalue);
    return container;
}

// 2 個參考
public static List<PredictIntent> Sentence2Intent(string sentence)
{
    //將句子轉成onnx張量容器(最大長度128個字，每個字對應成長度為256的特徵向量)
    var container = TensorHelper.StringToEmbeddedTensor(sentence);

    //產生推論結果
    var results = np.array(luis_session.Run(container).ToList()[0].AsEnumerable<float>().ToArray());
    //將推論結果由大至小排序(argsort是由小至大，然後":-1"是反轉)取前三名
    var top_results = np.argsort<float>(results);
    //根據索引串回intent(透過index2intent)以及對應機率
    List<PredictIntent> final_result = new List<PredictIntent>();

    for (int i = -1; i > -4; i--)
    {
        int idx = top_results.GetData(new int[] { i });
        float probs = results.GetData(new int[] { idx });
        final_result.Add(new PredictIntent() { Intent = (intent)idx, Ordinal = -i, Probability = probs });
    }
    return final_result;
}

```

```

txt='今天台股電股價是多少?'
def sentence2tensor(words):
    feature=np.zeros((256,128)) #預設都為零
    for i in range(128):
        if i < len(words):
            feature[:,i]=chars[words[i].lower()] if words[i].lower() in chars else np.random.randn(256)
    return expand_dims(feature,0).astype(np.float32)

print(sentence2tensor(txt).shape)

```

(1, 256, 128)

```

#把剛才流程包成函數
def sentence2intent(words):
    #透過onnx進行推論
    pred_intent= luis_session.run(None, {input_name:sentence2tensor(words)} )[0]

    #透過argmax找出機率最高前三名的意圖索引位置
    idx=argsort(pred_intent,-1)[::-1][0][:3]

    #列印出意圖名稱以及機率
    result=OrderedDict()
    for i in range(3):
        result[i]=(intent_dict[idx[i]],pred_intent[0][idx[i]])
    return result

sentence2intent(txt)

```

```
{ 0: ('詢問股票', 0.99999994), 1: ('詢問電視節目', 5.855678e-07), 2: ('打開app應用', 3.2424852e-10) }
```


2021/1/3 上午6:48:28

你好，我是小LUIS，很高興為您服務。

我想聽蔡依林的怪美的

```
{"inputSentence":"我想聽蔡依林的怪美的","preprocessedString":"我想聽蔡依林的怪美的","returnString":null,"predictIntents":[{"intent":16,"intentDesc":"音樂指令","probability":0.99999976,"ordinal":1}, {"intent":3,"intentDesc":"閒聊","probability":2.1388009e-7,"ordinal":2}, {"intent":28,"intentDesc":"影片播放指令","probability":1.2997777e-8,"ordinal":3}], "predictEntities":[{"entityType":0,"entityDesc":"人名","entityString":"蔡依林","probability":0}, {"entityType":0,"entityDesc":"人名","entityString":"美的","probability":0}], "wordSegs":[{"word":"我","flag":"r"}, {"word":"想","flag":"v"}, {"word":"聽","flag":"v"}, {"word":"蔡依林","flag":"nr"}, {"word":"的","flag":"uj"}, {"word":"怪","flag":"n"}, {"word":"美的","flag":"nr"}], "processTime":0.0217093}
```

輸入訊息...



實作Demo
<https://github.com/AllanYiin/MyLUIS>

其他需要重視的技術與非技術議題



真是好棒棒，
你們家是沒有
活人來接電話
了嗎？

- 需要根據額外的情緒偵測模型發現客戶情緒不佳，或者是根據意圖識別結果信心水準偏低來評估客戶回答的是意圖之外問題，而轉人工客服。
- 商用的聊天機器人盡量避免無意義的閒聊，以免佔用系統資源，需要透過對話技巧婉轉的結束對話。
- 強烈建議透過架構規劃、詳盡的壓力測試來實現CPU推論(本實作範例cpu單次推論約在0.02秒左右)，因為gpu推論運營成本實在高出太多。



Thanks for joining!

Ask questions on Twitter using #dotNETConf



.NET Conf
2020

特別感謝

91APP
Technical Network

KK TIX



HackMD

MVP
Microsoft®
Most Valuable
Professional

Microsoft

Build School

STUDY4
為 學 習 而 生

以及各位參與活動的你們

