Chat Bot 專題

Ming-Hsiang Su

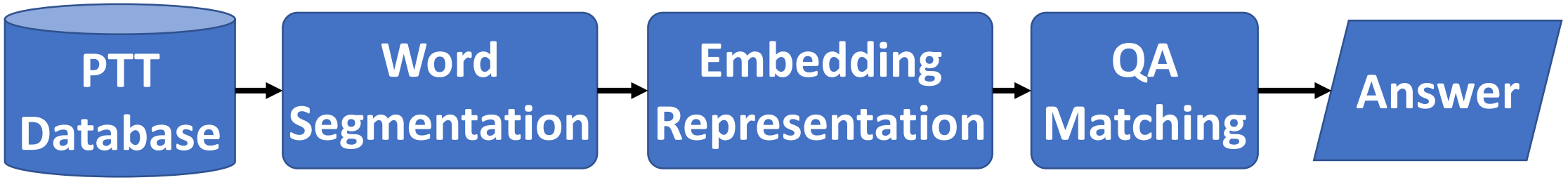
Project

- Write a program using the programming language (Python, C++,…) you are familiar with and/or the open source tools to construct a chatbot.
- Basic issue:
  - Feature Extraction: Word Embedding.
  - QA matching: Cosine measure.

Environment

- About deep learning:
  - https://ppt.cc/fe4Nfx

Processing


PTT Database
Word Segmentation
Embedding Representation
QA Matching
Answer

PTT Database

- PTT Chinese Database:
  - Total data: 418202
  - Download URL: https://goo.gl/TxJLPL

Word Segmentation (1/5)

- Jieba: Chinese text segmentation.
  - https://github.com/fxsjy/jieba
  - Install command: **pip install jieba**

```
(tensorflow) D:\使用者\huntfox\Desktop\chatwithme_VSM>python
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()

(tensorflow) D:\使用者\huntfox\Desktop\chatwithme_VSM>pip install jieba
Collecting jieba
Installing collected packages: jieba
Successfully installed jieba-0.39

(tensorflow) D:\使用者\huntfox\Desktop\chatwithme_VSM>
```

Word Segmentation (2/5)

- 3 models:
  - Full mode
  - Precise mode
  - Search engine mode

Word Segmentation (3/5)

- 3 parameters:
  - Input sentence
  - Model
  - HMM model
- Example:
  - **jieba.cut("今天晚餐的牛肉真的太好吃了", cut_all=False, HMM=True)**

Word Segmentation (4/5)

- Full mode:

```
(tensorflow) D:\使用者\huntfox\Desktop\chatwithme_VSM>python
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import jieba
>>> seg_list = jieba.cut("今天晚餐的牛肉真的太好吃了", cut_all=True)
>>> print("Full Mode: " + "/ ".join(seg_list))
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\huntfox\AppData\Local\Temp\jieba.cache
Loading model cost 0.935 seconds.
Prefix dict has been built succesfully.
Full Mode: 今天/ 晚餐/ 的/ 牛肉/ 真的/ 太/ 好吃/ 了
>>>
```

Word Segmentation (5/5)

- Load user dictionary:

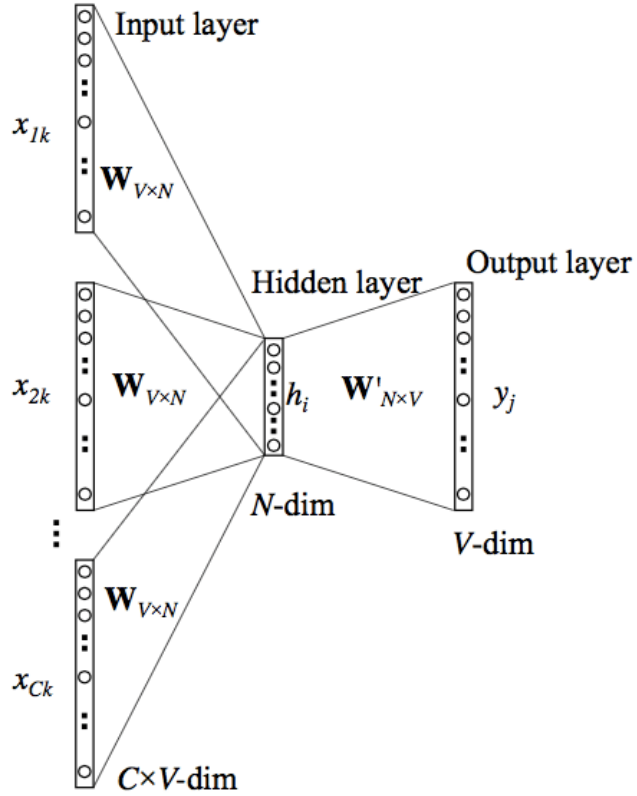
```
(tensorflow) D:\使用者\huntfox\Desktop\chatwithme_VSM>python
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import jieba
>>> dic_file = './corpus/GigaWord_Dic_extend.txt'
>>> jieba.load_userdict(dic_file)
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\huntfox\AppData\Local\Temp\jieba.cache
Loading model cost 0.908 seconds.
Prefix dict has been built succesfully.
>>> seg_list = jieba.cut("今天晚餐的牛肉真的太好吃了")
>>> print("Precise Mode: " + "/ ".join(seg_list))
Precise Mode: 今天/ 晚餐/ 的/ 牛肉/ 真的/ 太/ 好吃/ 了
>>>
```

Embedding Representation (1/10)

- Word2Vec:
  - Word2vec is a group of related models that are used to produce word embeddings.
  - Word2vec can utilize either of two model architectures to produce a distributed representation of words:
    - Continuous bag-of-words (CBOW).
    - Continuous skip-gram.

Embedding Representation (2/10)

- CBOW model:
  - Have a great day.
  - we are trying to predict a target word (**day**) using a single context input word **great**.



Embedding Representation (3/10)

- Download zhwiki database from Wikimedia
  - https://dumps.wikimedia.org/zhwiki/20190520/
- Install gensim package
  - pip install gensim

```
(tensorflow) D:\WIKI中文語料>pip install gensim
Collecting gensim
  Downloading https://files.pythonhosted.org/packages/81/80/858ef502e80baa6384b75fd5c89f01074b791a13b830487f9e25bdce50ec
/gensim-3.7.3.tar.gz (23.4MB)
    100% |████████████████████████████████| 23.4MB 559kB/s
Requirement already satisfied: numpy>=1.11.3 in d:\anaconda3\envs\tensorflow\lib\site-packages (from gensim) (1.16.3)
Requirement already satisfied: scipy>=0.18.1 in d:\anaconda3\envs\tensorflow\lib\site-packages (from gensim) (1.2.1)
Requirement already satisfied: six>=1.5.0 in d:\anaconda3\envs\tensorflow\lib\site-packages (from gensim) (1.12.0)
```

Embedding Representation (4/10)

```
from gensim.corpora import WikiCorpus
wiki_corpus = WikiCorpus('zhwiki-latest-pages-articles-multistream.xml.bz2', dictionary={})
texts_num = 0
with open("wiki_texts.txt",'w',encoding='utf-8') as output:
        for text in wiki_corpus.get_texts():
            output.write(' '.join(text) + '\n')
            texts_num += 1
            if texts_num % 10000 == 0:
                logging.info("已處理 %d 篇文章" % texts_num)
```

Embedding Representation (5/10)

```
(tensorflow) D:\WIKI中文語料>python wiki_to_txt.py zhwiki-latest-pages-articles-multistream.xml.bz2
D:\Anaconda3\envs\tensorflow\lib\site-packages\gensim\utils.py:1254: UserWarning: detected Windows; aliasing chunkize to
 chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
2019-06-02 07:04:24,415 : INFO : 已處理 10000 篇文章
2019-06-02 07:05:04,644 : INFO : 已處理 20000 篇文章
2019-06-02 07:05:40,961 : INFO : 已處理 30000 篇文章
2019-06-02 07:06:17,459 : INFO : 已處理 40000 篇文章
2019-06-02 07:06:55,245 : INFO : 已處理 50000 篇文章
2019-06-02 07:07:33,734 : INFO : 已處理 60000 篇文章
2019-06-02 07:08:08,978 : INFO : 已處理 70000 篇文章
2019-06-02 07:08:45,070 : INFO : 已處理 80000 篇文章
2019-06-02 07:09:20,652 : INFO : 已處理 90000 篇文章
2019-06-02 07:09:57,891 : INFO : 已處理 100000 篇文章
2019-06-02 07:10:45,063 : INFO : 已處理 110000 篇文章
2019-06-02 07:11:24,813 : INFO : 已處理 120000 篇文章
2019-06-02 07:12:07,505 : INFO : 已處理 130000 篇文章
```

Embedding Representation (6/10)

```
import jieba
jieba.set_dictionary('extra_dict/dict.txt.big')
stopword_set = set()
with open('extra_dict/stop_words.txt','r', encoding='utf-8') as stopwords:
        for stopword in stopwords:
            stopword_set.add(stopword.strip('\n'))
output = open('wiki_seg.txt', 'w', encoding='utf-8')
```

Embedding Representation (7/10)

```
with open('wiki_texts.txt', 'r', encoding='utf-8') as content :
    for texts_num, line in enumerate(content):
        line = line.strip('\n’)
        line = Converter('zh-hant').convert(line)
        line = line
        words = jieba.cut(line, cut_all=False)
        for word in words:
            if word not in stopword_set:
                output.write(word + ' ')
        output.write('\n')
        if (texts_num + 1) % 10000 == 0:
            logging.info("已完成前 %d 行的斷詞" % (texts_num + 1))
```

Embedding Representation (8/10)

```
(tensorflow) D:\WIKI中文語料>python segment.py
Building prefix dict from D:\WIKI中文語料\extra_dict\dict.txt.big ...
2019-06-02 07:30:45,183 : DEBUG : Building prefix dict from D:\WIKI中文語料\extra_dict\dict.txt.big ...
Dumping model to file cache C:\Users\huntfox\AppData\Local\Temp\jieba.u5e46a19d9221c67c1029f8d0730ba526.cache
2019-06-02 07:30:47,428 : DEBUG : Dumping model to file cache C:\Users\huntfox\AppData\Local\Temp\jieba.u5e46a19d9221c67c1029f8d0730ba526.cache
Loading model cost 2.432 seconds.
2019-06-02 07:30:47,616 : DEBUG : Loading model cost 2.432 seconds.
Prefix dict has been built succesfully.
2019-06-02 07:30:47,616 : DEBUG : Prefix dict has been built succesfully.
2019-06-02 07:34:24,847 : INFO : 已完成前 10000 行的斷詞
2019-06-02 07:37:05,261 : INFO : 已完成前 20000 行的斷詞
```

Embedding Representation (9/10)

- from gensim.models import word2vec
- from gensim.test.utils import common_texts
- sentences = word2vec.LineSentence('wiki_seg.txt')
- model = word2vec.Word2Vec(sentences, size = 50, window = 5, workers = 9, sg = 0, min_count=5)
- model.save('wiki.word2vec_50.bin')
- # model = word2vec.Word2Vec.load('wiki.word2vec_50.bin’)
- # vec_obj = model.wv["冰沙"]

Embedding Representation (10/10)

```
(tensorflow) D:\WIKI中文語料>python 3_1_Word2Vec_VectorTrain.py wiki_seg.txt wiki.word2vec_300.bin
2019-06-02 17:04:41,934 : WARNING : consider setting layer size to a multiple of 4 for greater performance
D:\Anaconda3\envs\tensorflow\lib\site-packages\gensim\models\base_any2vec.py:743: UserWarning: C extension not loaded, training will be slow.
  "C extension not loaded, training will be slow. "
2019-06-02 17:04:41,996 : INFO : collecting all words and their counts
2019-06-02 17:04:41,996 : WARNING : this function is deprecated, use smart_open.open instead
2019-06-02 17:04:42,059 : INFO : PROGRESS: at sentence #0, processed 0 words, keeping 0 word types
2019-06-02 17:04:47,715 : INFO : PROGRESS: at sentence #10000, processed 11262237 words, keeping 689834 word types
2019-06-02 17:04:51,471 : INFO : PROGRESS: at sentence #20000, processed 19507940 words, keeping 986286 word types
2019-06-02 17:04:54,666 : INFO : PROGRESS: at sentence #30000, processed 26910139 words, keeping 1194132 word types
2019-06-02 17:04:57,675 : INFO : PROGRESS: at sentence #40000, processed 33805312 words, keeping 1382329 word types
2019-06-02 17:05:00,560 : INFO : PROGRESS: at sentence #50000, processed 40250282 words, keeping 1539875 word types
2019-06-02 17:05:03,330 : INFO : PROGRESS: at sentence #60000, processed 46467297 words, keeping 1685525 word types
2019-06-02 17:05:05,896 : INFO : PROGRESS: at sentence #70000, processed 52323121 words, keeping 1815031 word types
2019-06-02 17:05:08,359 : INFO : PROGRESS: at sentence #80000, processed 57946595 words, keeping 1943453 word types
2019-06-02 17:05:10,771 : INFO : PROGRESS: at sentence #90000, processed 63441824 words, keeping 2057358 word types
2019-06-02 17:05:13,272 : INFO : PROGRESS: at sentence #100000, processed 69037784 words, keeping 2165897 word types
2019-06-02 17:05:15,624 : INFO : PROGRESS: at sentence #110000, processed 74348815 words, keeping 2263301 word types
2019-06-02 17:05:17,773 : INFO : PROGRESS: at sentence #120000, processed 79179331 words, keeping 2356249 word types
2019-06-02 17:05:20,217 : INFO : PROGRESS: at sentence #130000, processed 84662078 words, keeping 2460118 word types
2019-06-02 17:05:22,483 : INFO : PROGRESS: at sentence #140000, processed 89694603 words, keeping 2549999 word types
2019-06-02 17:05:24,859 : INFO : PROGRESS: at sentence #150000, processed 95053046 words, keeping 2643173 word types
```

QA Matching (1/6)

- Cosine similarity:
  - A measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.
  - $similarity = \cos(\theta) = \frac{A \cdot B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$

    where A_i and B_i are components of vector A and B respectively.

QA Matching (2/6)

```
model = "wiki.word2vec_50.bin"
model_w2v = word2vec.Word2Vec.load(model)
candidates = []
with open(target, encoding='utf-8')as f:
    for line in f:
        candidates.append(line.strip().split())
```

QA Matching (3/6)

```
text = "為什麼PTT這麼多人看棒球？"
words = list(jieba.cut(text.strip()))
word = []
for w in words:
    if w not in model_w2v.wv.vocab:
        print("input word %s not in dict. skip this turn" % w)
    else:
        word.append(w)
```

QA Matching (4/6)

```
flag = False
res = []
index = 0
for candidate in candidates:
    for c in candidate:
        if c not in model_w2v.wv.vocab:
            print("candidate word %s not in dict. skip this turn" % c)
            flag = True
    if flag:
        break
    score = model_w2v.n_similarity(word, candidate)
    resultInfo = {'id': index, "score": score, "text": " ".join(candidate)}
    res.append(resultInfo)
    index += 1
```

QA Matching (5/6)

```
res.sort(key=lambda x: x['score'], reverse=True)
result = []
for i in range(len(res)):
    if res[i]['score'] > 0.80:
        dict_temp = {res[i]['id']: res[i]['text'], 'score': res[i]['score']}
        result.append(dict_temp)
```

QA Matching (6/6)

```
result=[{293: '肥宅 才 看 棒球 系壘 一堆 胖子', 'score': 0.900805},
{13: '有 完 沒完 去 棒球 板 啦', 'score': 0.89326227},
{292: '我 棒球 慢 壘 兩邊 都 打 打擊 沒 問題 守備 就 不 太 行 了', 'score': 0.8441039},
```

Evaluation(1/2)

- Question upload time: **2019/06/20 pm 18:00**
- Answer upload time: **2019/06/20 pm 22:00**
- Total number of questions: 1000
- The type of question: Choice question
- Example:
  - 為什麼PTT這麼多人看棒球？　(1)還好啦，海陸下基地常態，下好下滿 (2)下一個是華哥啦 (3)肥宅才看棒球，系壘一堆胖子。(4)以前在台灣超紅

Evaluation(2/2)

- FTP information:
  - IP: **140.116.82.118**
  - ID: **multimedia**
  - Password: **2019spring**
- Upload file format:
  1. **學號.csv**
  2. **程式**
  3. **程式說明 word 檔**

Q&A