Word Embeddings & Recurrent Neural Networks

助教:朱璟軒

Word Embeddings

- Getting data
- Data processing (word segmentation…)
- Word2Vec

Word Embeddings

- ~~Getting data~~
- Data processing (~~word segmentation,~~…)
- Word2Vec

gensim

topic modelling for humans

https://radimrehurek.com/gensim/

- Import

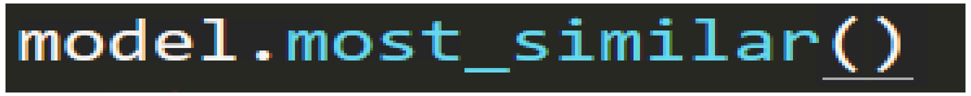
```
from gensim.models import word2vec
```

- Declare model

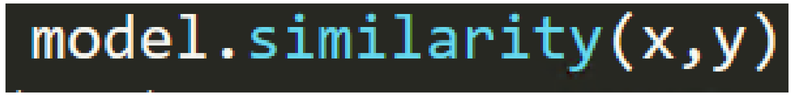
```
model = word2vec.Word2Vec(sentences, size=250)
```

- class gensim.models.word2vec.Word2Vec()
  - **sentences:**The *sentences* iterable can be simply a list of lists of tokens
  - **size:**Dimensionality of the feature vectors
  - **alpha:**The initial learning rate
  - **sg:**Defines the training algorithm. If 1, skip-gram is employed; otherwise, CBOW is used
  - **window:**The maximum distance between the current and predicted word within a sentence
  - **workers:**Use these many worker threads to train the model
  - **min_count:**Ignores all words with total frequency lower than this

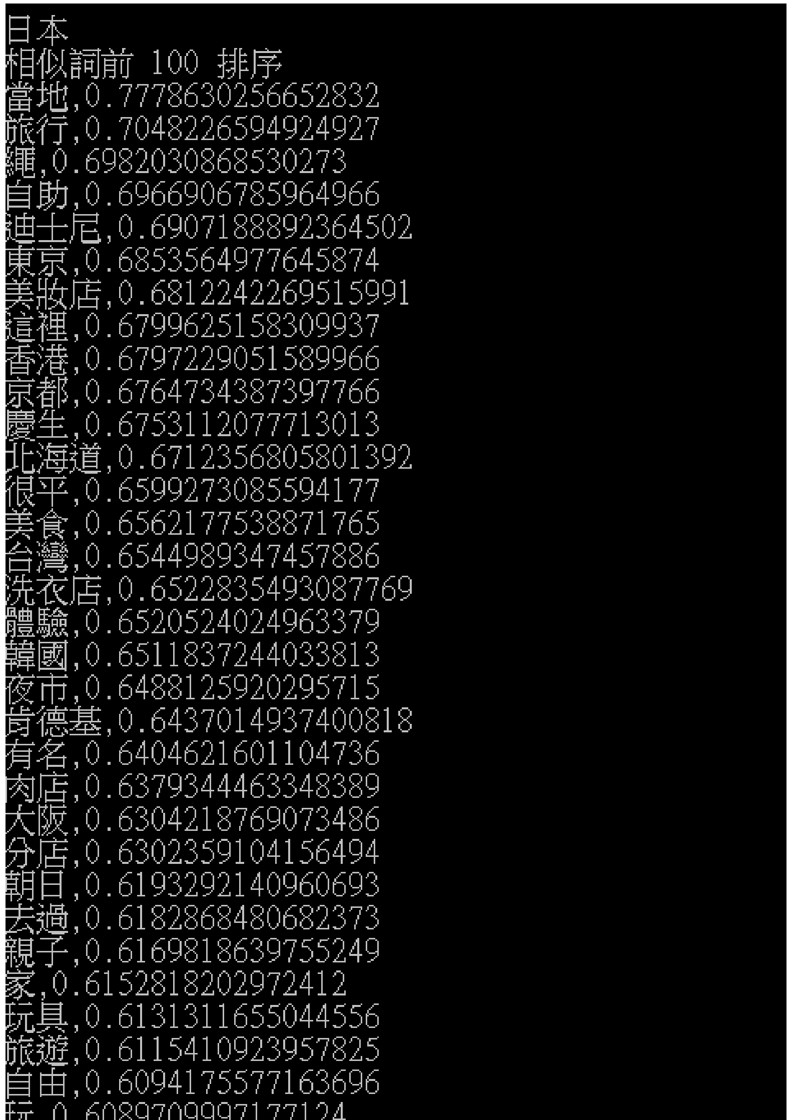
```
model.save("word2vec.model")
```



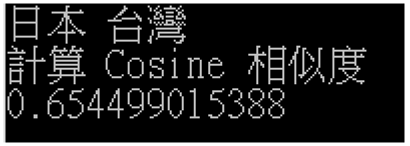
```
model.most_similar()
```



```
model.similarity(x,y)
```



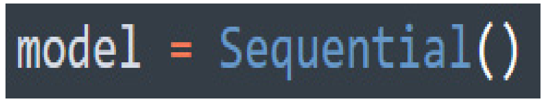
```
日本
相似詞前 100 排序
當地,0.7778630256652832
旅行,0.7048226594924927
繩,0.6982030868530273
自助,0.6966906785964966
迪士尼,0.6907188892364502
東京,0.6853564977645874
美妝店,0.6812242269515991
這裡,0.6799625158309937
香港,0.6797229051589966
京都,0.6764734387397766
慶生,0.6753112077713013
北海道,0.6712356805801392
很平,0.6599273085594177
美食,0.6562177538871765
台灣,0.6544989347457886
洗衣店,0.6522835493087769
體驗,0.6520524024963379
韓國,0.6511837244033813
夜市,0.6488125920295715
肯德基,0.6437014937400818
有名,0.6404621601104736
肉店,0.6379344463348389
大阪,0.6304218769073486
分店,0.6302359104156494
朝日,0.6193292140960693
去過,0.6182868480682373
親子,0.6169818639755249
家,0.6152818202972412
玩具,0.6131311655044556
旅遊,0.6115410923957825
自由,0.6094175577163696
玩,0.6089709997177124
```



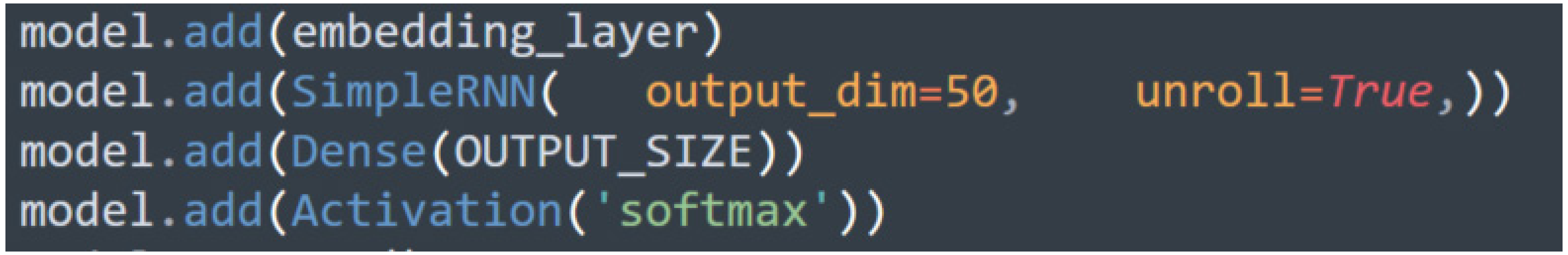
```
日本 台灣
計算 Cosine 相似度
0.654499015388
```

classification-use RNN

- Import
- Declare variables(X_train,X_test,Y_train, Y_test)
- Declare Model



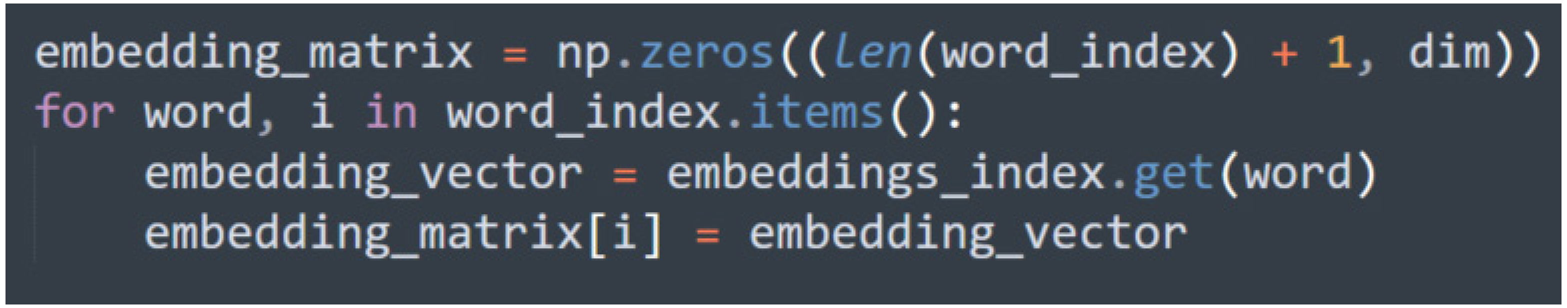
```
model = Sequential()
```



```
model.add(embedding_layer)
model.add(SimpleRNN(   output_dim=50,    unroll=True,))
model.add(Dense(OUTPUT_SIZE))
model.add(Activation('softmax'))
```

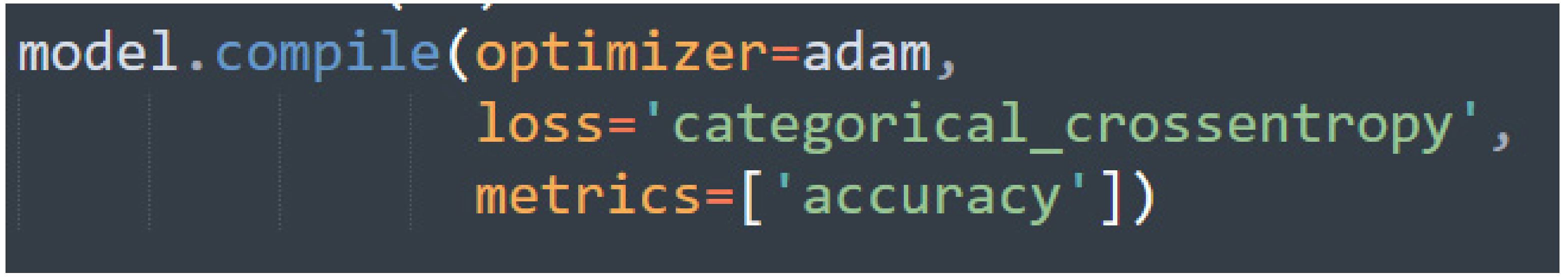
Embedding_layer

- Load pretrain word_bedding for layer weight



```python
embedding_matrix = np.zeros((len(word_index) + 1, dim))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    embedding_matrix[i] = embedding_vector
```

- Model compile

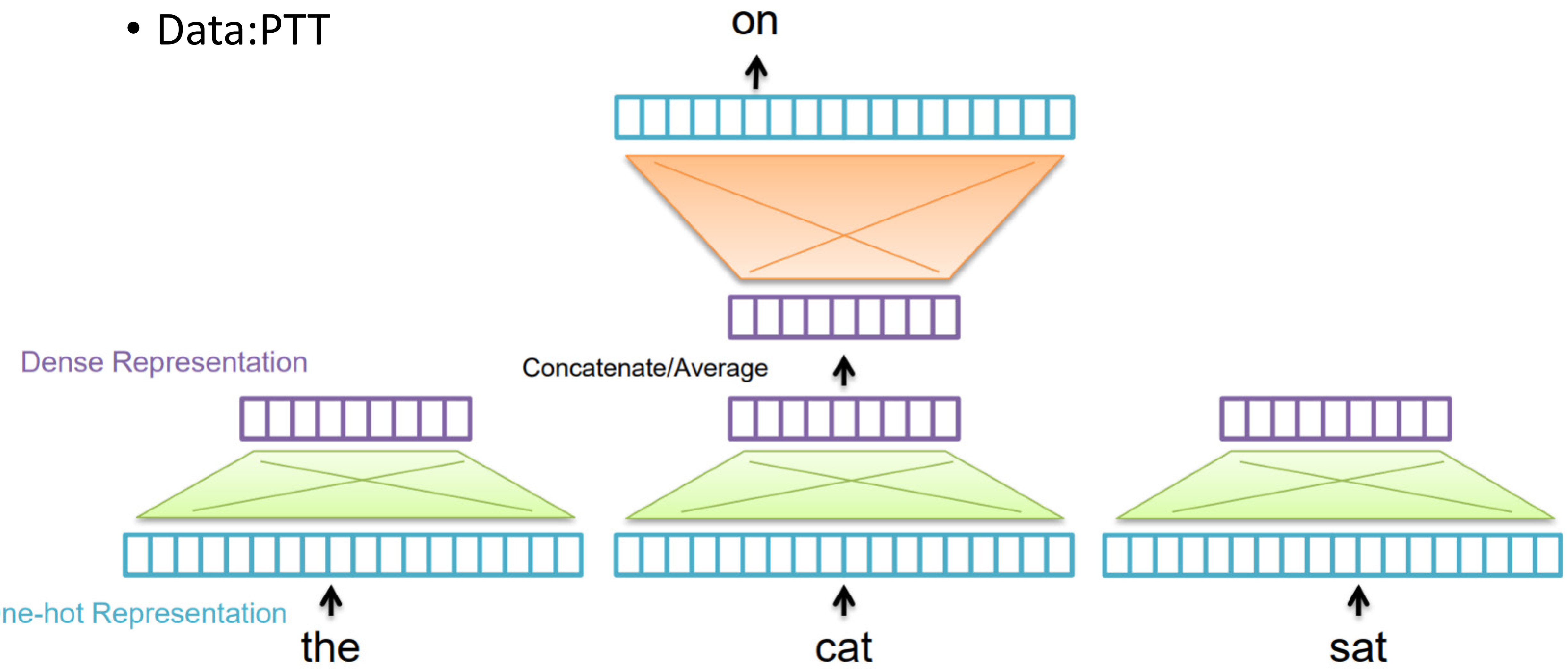


```
model.compile(optimizer=adam,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

- Training
- Testing

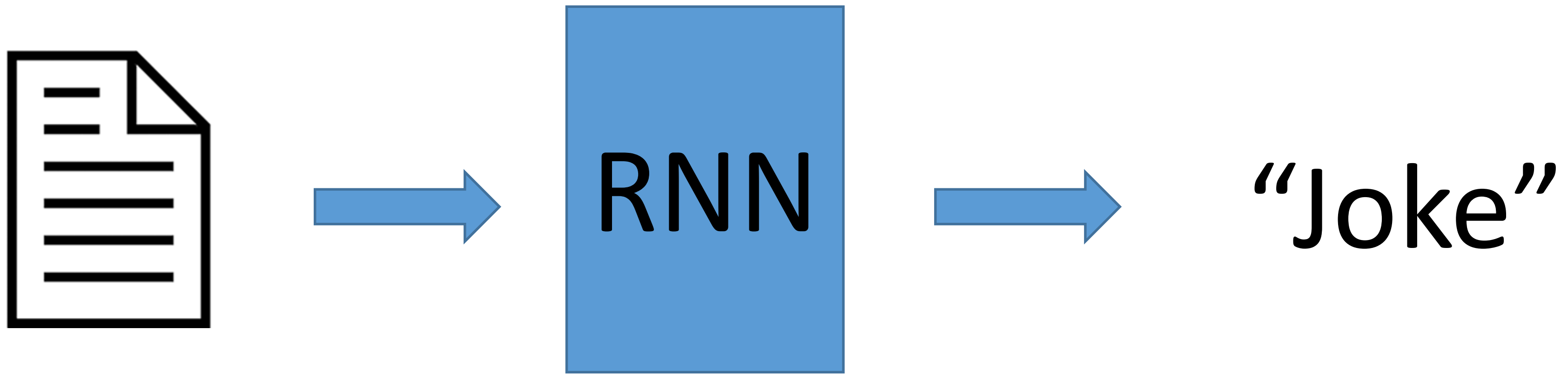
HW3-1: Word Embedding

- Data:PTT



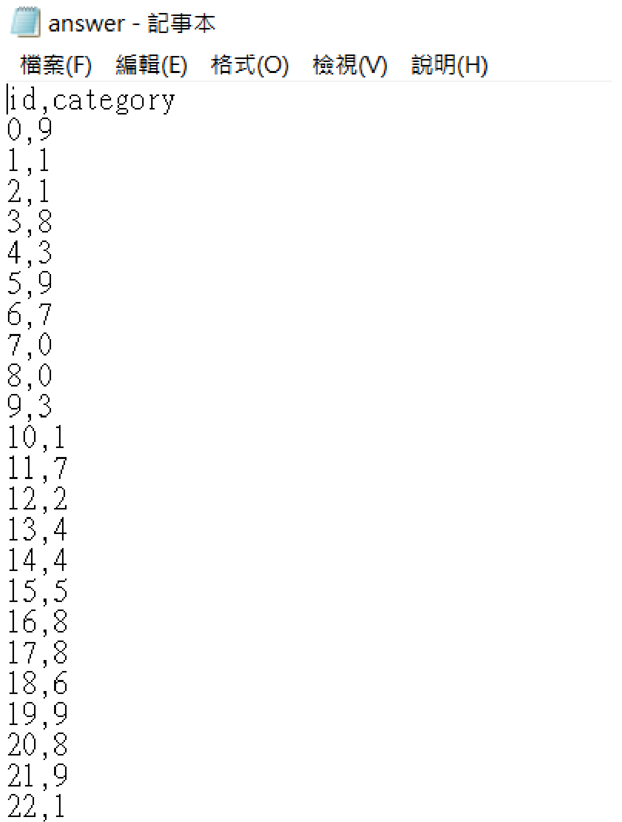
HW3-2: Document Classification

- Data:PTT(training:900*10,testing:1000)
- Kaggle: https://bit.ly/2Hkm7MU



Submission Format

- {'Japan_Travel': 0, 'KR_ENTERTAIN': 1, 'Makeup': 2, 'Tech_Job': 3, 'WomenTalk': 4, 'babymother': 5, 'e-shopping': 6, 'graduate': 7, 'joke': 8, 'movie': 9}



```
id,category
0,9
1,1
2,1
3,8
4,3
5,9
6,7
7,0
8,0
9,3
10,1
11,7
12,2
13,4
14,4
15,5
16,8
17,8
18,6
19,9
20,8
21,9
22,1
```

Baseline

- NN structure
    - Embedding_layer(output_dim=400)
    - SimpleRNN(output_dim=50, unroll=True)
    - Dense(output_dim=10)
    - Activation('softmax')
- optimizer=adam
- loss=categorical_crossentropy
- metrics=accuracy
- Epoch:10

Scoring(15%)

- Code: must
- Report: 7%
- Kaggle: 8%
  - Must over baseline
  - $YourScore = 3 + 5 \times \frac{YourAcc - Baseline}{MaxAcc - Baseline}$
- Dead line: two weeks

What report should cover?

- HW3-1(2%)
- HW3-2
  - Model description (1%)
  - How do you improve your performance (3%)
  - Experiment settings and results (1%)
    - Ex. Epochs, training time, hyperparameters, etc.
- No more than 2 pages
- Please written in Chinese