

人工智慧文本分析



Tamkang
Universit
淡江大學

(Artificial Intelligence for Text Analytics)

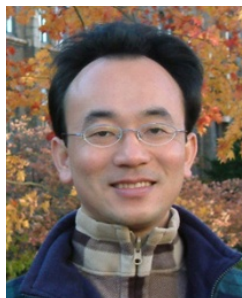
Python 自然語言處理

(Python for Natural Language Processing)

1082AITA03

MBA, IMTKU (M2455) (8410) (Spring 2020)

Wed 8, 9 (15:10-17:00) (B605)



Min-Yuh Day

戴敏育

Associate Professor

副教授

Dept. of Information Management, Tamkang University

淡江大學 資訊管理學系

<http://mail.tku.edu.tw/myday/>

2020-03-18



課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2020/03/04	人工智慧文本分析課程介紹 (Course Orientation on Artificial Intelligence for Text Analytics)
2	2020/03/11	文本分析的基礎：自然語言處理 (Foundations of Text Analytics: Natural Language Processing; NLP)
3	2020/03/18	Python自然語言處理 (Python for Natural Language Processing)
4	2020/03/25	處理和理解文本 (Processing and Understanding Text)
5	2020/04/01	文本表達特徵工程 (Feature Engineering for Text Representation)
6	2020/04/08	人工智慧文本分析個案研究 I (Case Study on Artificial Intelligence for Text Analytics I)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
7	2020/04/15	文本分類 (Text Classification)
8	2020/04/22	文本摘要和主題模型 (Text Summarization and Topic Models)
9	2020/04/29	期中報告 (Midterm Project Report)
10	2020/05/06	文本相似度和分群 (Text Similarity and Clustering)
11	2020/05/13	語意分析和命名實體識別 (Semantic Analysis and Named Entity Recognition; NER)
12	2020/05/20	情感分析 (Sentiment Analysis)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
13	2020/05/27	人工智慧文本分析個案研究 II (Case Study on Artificial Intelligence for Text Analytics II)
14	2020/06/03	深度學習和通用句子嵌入模型 (Deep Learning and Universal Sentence-Embedding Models)
15	2020/06/10	問答系統與對話系統 (Question Answering and Dialogue Systems)
16	2020/06/17	期末報告 I (Final Project Presentation I)
17	2020/06/24	期末報告 II (Final Project Presentation II)
18	2020/07/01	教師彈性補充教學

Outline

- Python for
Natural Language Processing

Python for Natural Language Processing



Connect Google Colab in Google Drive

The screenshot shows the Google Drive web interface. The browser tab is 'My Drive - Google Drive' and the URL is 'https://drive.google.com/drive/u/2/my-drive'. The Drive logo and a search bar are at the top. On the left, the 'New' button is highlighted with a red dashed box. A dropdown menu is open, showing options like 'New folder...', 'Upload files...', 'Upload folder...', 'Google Docs', 'Google Sheets', 'Google Slides', and 'More'. The 'More' option is also highlighted with a red dashed box. A second dropdown menu is open from 'More', showing 'Google Forms', 'Google Drawings', 'Google My Maps', 'Google Sites', and 'Connect more apps'. The 'Connect more apps' option is highlighted with a red dashed box. The main area shows 'My Drive' and 'Quick Access' sections. The bottom of the page has a storage status bar and a 'Get Backup and Sync for Mac' notification.

My Drive - Google Drive

https://drive.google.com/drive/u/2/my-drive

Drive

Search Drive

My Drive

Quick Access

New

My Drive

Computers

Shared with me

Recent

Starred

Trash

Backups

Storage

0 bytes of 15 GB used

UPGRADE STORAGE

Get Backup and Sync for Mac

New folder...

Upload files...

Upload folder...

Google Docs

Google Sheets

Google Slides

More

Google Forms

Google Drawings

Google My Maps

Google Sites

Connect more apps

Name ↑

Google Colab

My Drive - Google Drive x +

https://drive.google.com/drive/u/2/my-drive

Drive

Search Drive

Connect apps to Drive

All ▾ colab x

 ZIP Extractor Extract ZIP files to Google Drive Extraction complete. View extracted files Share Extract another  Test.zip ZIP Extractor 307,585 users	 LUMIN PDF The fast and simple PDF Viewer    Lumin PDF - Beautiful PDF Editor 289,310 users	 cloudconvert CloudConvert 373,161 users
 Sejda Merge PDF - Split PDF - Sejda.com ★★★★★ (1106)	 DocHub Edit, Send & Sign PDFs DocHub - Edit and Sign PDF Docu... 2,131,600 users	 Google Forms Google Forms 4,803,614 users

Get Backup and Sync for Mac

Access anywhere
Every file is online whenever.

Share easily
Give others access to any file or folder.

Name ↑

Google Colab

My Drive - Google Drive x +

https://drive.google.com/drive/u/2/my-drive

Drive

Search Drive

New

My Drive

Computers

Shared with me

Recent

Starred

Trash

Backups

Storage

0 bytes of 15 GB used

[UPGRADE STORAGE](#)

Get Backup and Sync for Mac

Access anywhere

Share easily

Connect apps to Drive

All colab



Colaboratory
offered by <https://colab.research.google.com>
A data analysis tool that combines code, output, and descriptive text into one collaborative document.

[+ CONNECT](#)

Productivity
★★★★★ (195)

Name ↑

Connect Colaboratory to Google Drive

The screenshot shows the Google Drive web interface. A dialog box titled "Connect apps to Drive" is open, displaying a search for "colab". A confirmation message from Colaboratory is shown: "Colaboratory was connected to Google Drive." Below this message is a checked checkbox labeled "Make Colaboratory the default app for files it can open". An "OK" button is visible at the bottom right of the confirmation message. The background shows the Google Drive sidebar with options like "New", "My Drive", "Computers", "Shared with me", "Recent", "Starred", "Trash", "Backups", and "Storage". The top navigation bar includes the Google Drive logo, a search bar, and various utility icons.

My Drive - Google Drive

https://drive.google.com/drive/u/2/my-drive

Drive

Search Drive

New

My Drive

Computers

Shared with me

Recent

Starred

Trash

Backups

Storage

0 bytes of 15 GB used

UPGRADE STORAGE

Get Backup and Sync for Mac

Access anywhere

Share easily

Connect apps to Drive

colab

Colaboratory was connected to Google Drive.

☒ Make Colaboratory the default app for files it can open

OK

Google Colab

The image shows a web browser window with the Google Drive interface. The address bar displays the URL `https://drive.google.com/drive/u/2/my-drive`. The Drive logo and a search bar are at the top. On the left sidebar, the 'New' button is highlighted with a red dashed box. A dropdown menu is open, showing options like 'New folder...', 'Upload files...', 'Upload folder...', 'Google Docs', 'Google Sheets', 'Google Slides', and 'More'. The 'More' option is also highlighted with a red dashed box. A second dropdown menu is open from 'More', listing 'Google Forms', 'Google Drawings', 'Google My Maps', 'Google Sites', and 'Colaboratory'. The 'Colaboratory' option is highlighted with a red dashed box. At the bottom left, there is a notification for 'Get Backup and Sync for Mac'.

My Drive - Google Drive

Search Drive

My Drive

Quick Access

New

My Drive

Computers

Shared with me

Recent

Starred

Trash

Backups

Storage

0 bytes of 15 GB used

UPGRADE STORAGE

Get Backup and Sync for Mac

New folder...

Upload files...

Upload folder...

Google Docs

Google Sheets

Google Slides

More

Google Forms

Google Drawings

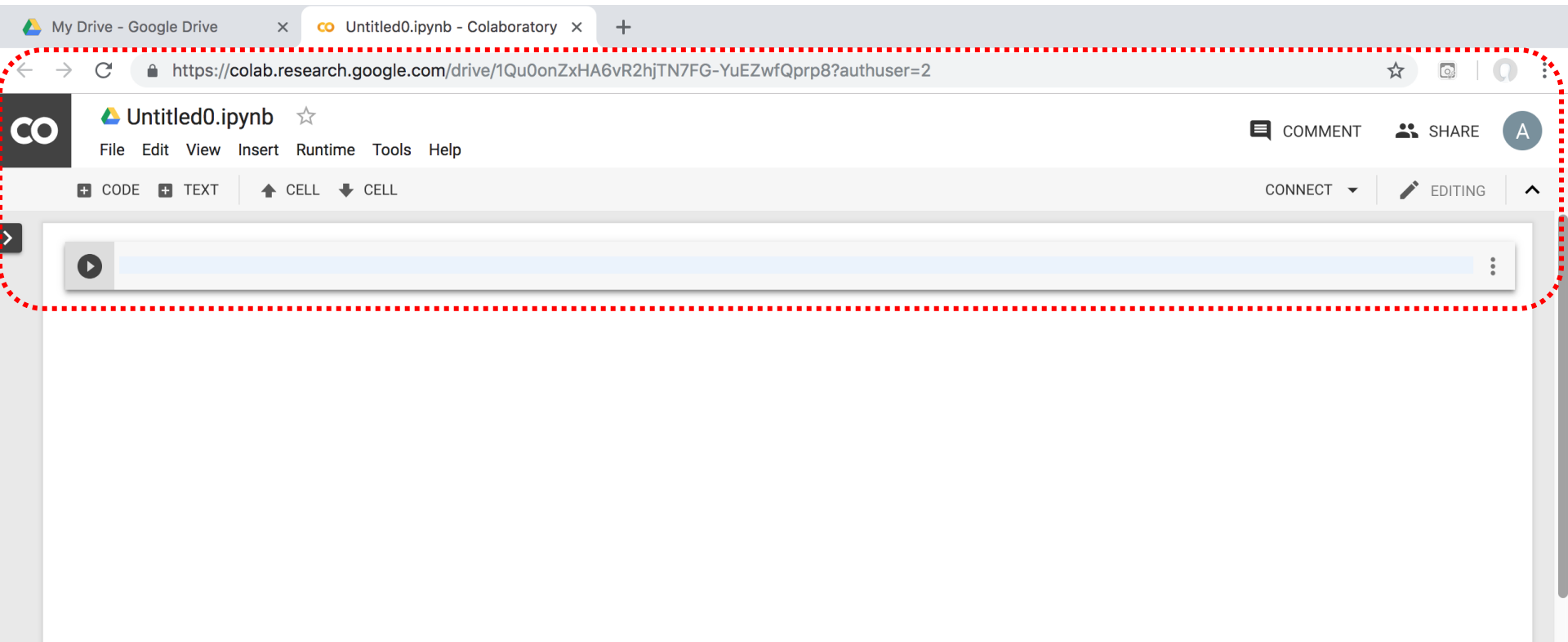
Google My Maps

Google Sites

Colaboratory

Connect more apps

Google Colab



Google Colab

The screenshot shows the Google Colab web interface. At the top, there are browser tabs for 'My Drive - Google Drive' and 'Untitled0.ipynb - Colaboratory'. The address bar shows the URL: <https://colab.research.google.com/drive/1Qu0onZxHA6vR2hjTN7FG-YuEZwfQprp8?authuser=2>. The main header area includes the Google Colab logo, the file name 'Untitled0.ipynb', and a star icon. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. The 'Runtime' menu is open, showing a list of options: 'Run all' (⌘/Ctrl+F9), 'Run before' (⌘/Ctrl+F8), 'Run the focused cell' (⌘/Ctrl+Enter), 'Run selection' (⌘/Ctrl+Shift+Enter), 'Run after' (⌘/Ctrl+F10), 'Interrupt execution' (⌘/Ctrl+M I), 'Restart runtime...' (⌘/Ctrl+M .), 'Restart and run all...', 'Reset all runtimes...', 'Change runtime type', and 'Manage sessions'. The 'Runtime' menu item in the top bar and the 'Change runtime type' option in the dropdown are highlighted with red dashed boxes. On the right side of the interface, there are buttons for 'COMMENT', 'SHARE', and a user profile icon. Below these are 'CONNECT' and 'EDITING' buttons. The main workspace area is visible in the background, showing a code editor with a blue line of code.

Run Jupyter Notebook Python3 GPU Google Colab

The screenshot shows the Google Colab web interface. The browser tab is titled "Untitled0.ipynb - Colaboratory". The address bar shows the URL: <https://colab.research.google.com/drive/1Qu0onZxHA6vR2hjTN7FG-YuEZwfQprp8?authuser=2>. The Colab logo is in the top left. The top navigation bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". On the right, there are buttons for "COMMENT", "SHARE", and a user profile icon. Below the navigation bar, there are tabs for "+ CODE", "+ TEXT", and cell navigation arrows. The main area is a Jupyter notebook editor. A "Notebook settings" dialog box is open in the center. It has a title "Notebook settings". Inside, there are two dropdown menus: "Runtime type" set to "Python 3" and "Hardware accelerator" set to "GPU". Both dropdowns are highlighted with red dashed boxes. Below these, there is a checkbox labeled "Omit code cell output when saving this notebook" which is currently unchecked. At the bottom right of the dialog are "CANCEL" and "SAVE" buttons.

My Drive - Google Drive x Untitled0.ipynb - Colaboratory x +

← → ↻ <https://colab.research.google.com/drive/1Qu0onZxHA6vR2hjTN7FG-YuEZwfQprp8?authuser=2> ☆ 📷 🔊

co Untitled0.ipynb ☆

File Edit View Insert Runtime Tools Help

+ CODE + TEXT ⬆ CELL ⬇ CELL

CONNECT ▾ EDITING ⬆

>

Notebook settings

Runtime type
Python 3 ▾

Hardware accelerator
GPU ▾ ?

☐ Omit code cell output when saving this notebook

CANCEL SAVE

Google Colab Python Hello World

```
print('Hello World')
```

The screenshot displays the Google Colaboratory web interface. The browser tab is titled 'Untitled0.ipynb - Colaboratory'. The address bar shows the URL: <https://colab.research.google.com/drive/1Qu0onZxHA6vR2hjTN7FG-YuEZwfQprp8?authuser=2#scrollTo=6s-m3sER8G1u>. The Colab logo is in the top left. The main header area includes the file name 'Untitled0.ipynb' with a star icon, and navigation links: 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the right, there are 'COMMENT' and 'SHARE' buttons, and a user profile icon labeled 'A'. Below the header, a toolbar shows '+ CODE', '+ TEXT', '↑ CELL', and '↓ CELL'. The status bar on the right indicates 'CONNECTED' with a green checkmark and 'EDITING' with a pencil icon. The main workspace contains a single code cell with a play button icon on the left. The code inside the cell is `print('Hello World')`. Below the code, the output is displayed as 'Hello World' with a copy icon to its left.

Natural Language Processing (NLP) and Text Mining

Raw text

Sentence Segmentation

Tokenization

Part-of-Speech (POS)

Stop word removal

Stemming / Lemmatization

Dependency Parser

String Metrics & Matching

word's stem

am → am

having → hav

word's lemma

am → be

having → have

spaCy:

Natural Language Processing

spaCy

USAGE

MODELS

API

UNIVERSE



Search docs

Industrial-Strength Natural Language Processing

IN PYTHON

Get things done

spaCy is designed to help you do real work — to build real products, or gather real insights. The library respects your time, and tries to avoid wasting it. It's easy to install, and its API is simple and productive. We like to think of spaCy as the Ruby on Rails of Natural Language Processing.

Blazing fast

spaCy excels at large-scale information extraction tasks. It's written from the ground up in carefully memory-managed Cython. Independent research in 2015 found spaCy to be the fastest in the world. If your application needs to process entire web dumps, spaCy is the library you want to be using.

Deep learning

spaCy is the best way to prepare text for deep learning. It interoperates seamlessly with TensorFlow, PyTorch, scikit-learn, Gensim and the rest of Python's awesome AI ecosystem. With spaCy, you can easily construct linguistically sophisticated statistical models for a variety of NLP problems.

<https://spacy.io/>

Python in Google Colab

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

 python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share ⚙️ A

+ Code + Text

RAM  Disk  Editing ^

▼ Text Analytics and Natural Language Processing (NLP)

▼ Python for Natural Language Processing

spaCy

- spaCy: Industrial-Strength Natural Language Processing in Python
- Source: <https://spacy.io/usage/spacy-101>

[] 1 `#!python -m spacy download en_core_web_sm`



```
1 import spacy
2 nlp = spacy.load("en_core_web_sm")
3 doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
4 for token in doc:
5     print(token.text, token.pos_, token.dep_)
```

 Apple PROPN nsubj
is VERB aux
looking VERB ROOT
at ADP prep
buying VERB pcomp
U.K. PROPN compound
startup NOUN dobj
for ADP prep
\$ SYM quantmod
1 NUM compound
billion NUM pobj

Python in Google Colab

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

 python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share Settings A

+ Code + Text

✓ RAM Disk Editing ^

```
[ ] 1 import spacy
    2 nlp = spacy.load("en_core_web_sm")
    3 doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
    4 import pandas as pd
    5 cols = ("text", "lemma", "POS", "explain", "stopword")
    6 rows = []
    7 for t in doc:
    8     row = [t.text, t.lemma_, t.pos_, spacy.explain(t.pos_), t.is_stop]
    9     rows.append(row)
   10 df = pd.DataFrame(rows, columns=cols)
   11 df
```



	text	lemma	POS	explain	stopword
0	Apple	Apple	PROPN	proper noun	False
1	is	be	VERB	verb	True
2	looking	look	VERB	verb	False
3	at	at	ADP	adposition	True
4	buying	buy	VERB	verb	False
5	U.K.	U.K.	PROPN	proper noun	False
6	startup	startup	NOUN	noun	False
7	for	for	ADP	adposition	True
8	\$	\$	SYM	symbol	False
9	1	1	NUM	numeral	False
10	billion	billion	NUM	numeral	False

Python in Google Colab

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

 python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

✓ RAM
Disk

Editing

```
[ ] 1 import spacy
    2 nlp = spacy.load("en_core_web_sm")
    3 doc = nlp("Stanford University is located in California. It is a great university.")
    4 import pandas as pd
    5 cols = ("text", "lemma", "POS", "explain", "stopword")
    6 rows = []
    7 for t in doc:
    8     row = [t.text, t.lemma_, t.pos_, spacy.explain(t.pos_), t.is_stop]
    9     rows.append(row)
   10 df = pd.DataFrame(rows, columns=cols)
   11 df
```



	text	lemma	POS	explain	stopword
0	Stanford	Stanford	PROPN	proper noun	False
1	University	University	PROPN	proper noun	False
2	is	be	VERB	verb	True
3	located	locate	VERB	verb	False
4	in	in	ADP	adposition	True
5	California	California	PROPN	proper noun	False
6	.	.	PUNCT	punctuation	False
7	It	-PRON-	PRON	pronoun	True
8	is	be	VERB	verb	True
9	a	a	DET	determiner	True
10	great	great	ADJ	adjective	False
11	university	university	NOUN	noun	False
12	.	.	PUNCT	punctuation	False

Python in Google Colab

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

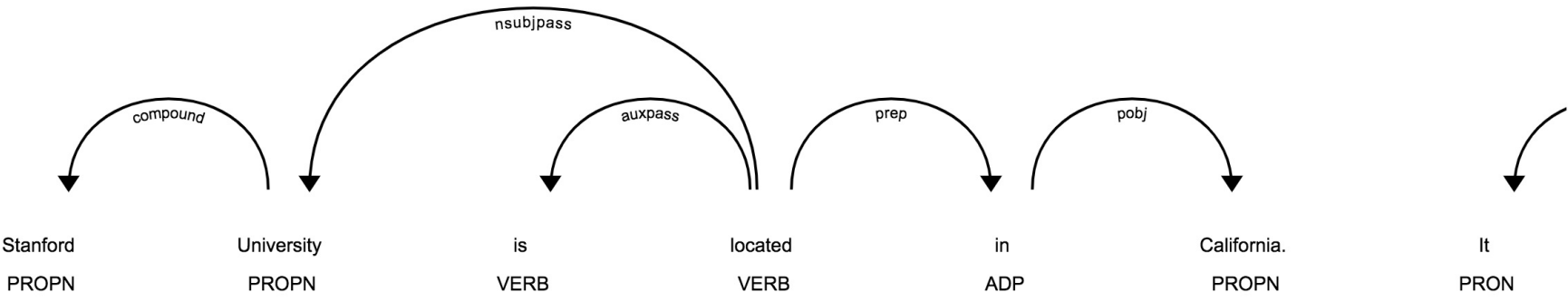
python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

✓ RAM  Disk  Editing

```
[ ] 1 from spacy import displacy
    2 displacy.render(doc, style="dep", jupyter=True)
```



The diagram shows a dependency parse tree for the sentence "Stanford University is located in California. It". The nodes are labeled with their part of speech: Stanford (PROPN), University (PROPN), is (VERB), located (VERB), in (ADP), California. (PROPN), and It (PRON). The dependencies are represented by curved arrows with labels: "compound" from Stanford to University, "nsubjpass" from University to is, "auxpass" from is to located, "prep" from located to in, "pobj" from in to California., and a separate arrow from It.

Python in Google Colab

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>



python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] 1 import spacy
    2 nlp = spacy.load("en_core_web_sm")
    3 text = "Stanford University is located in California. It is a great university."
    4 doc = nlp(text)
    5 for ent in doc.ents:
    6     print(ent.text, ent.label_)
```

☞ Stanford University ORG
California GPE

```
[ ] 1 from spacy import displacy
    2 text = "Stanford University is located in California. It is a great university."
    3 doc = nlp(text)
    4 displacy.render(doc, style="ent", jupyter=True)
```

☞ Stanford University ORG is located in California GPE . It is a great university.

Python in Google Colab

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

 python101.ipynb ☆

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL

CONNECT EDITING

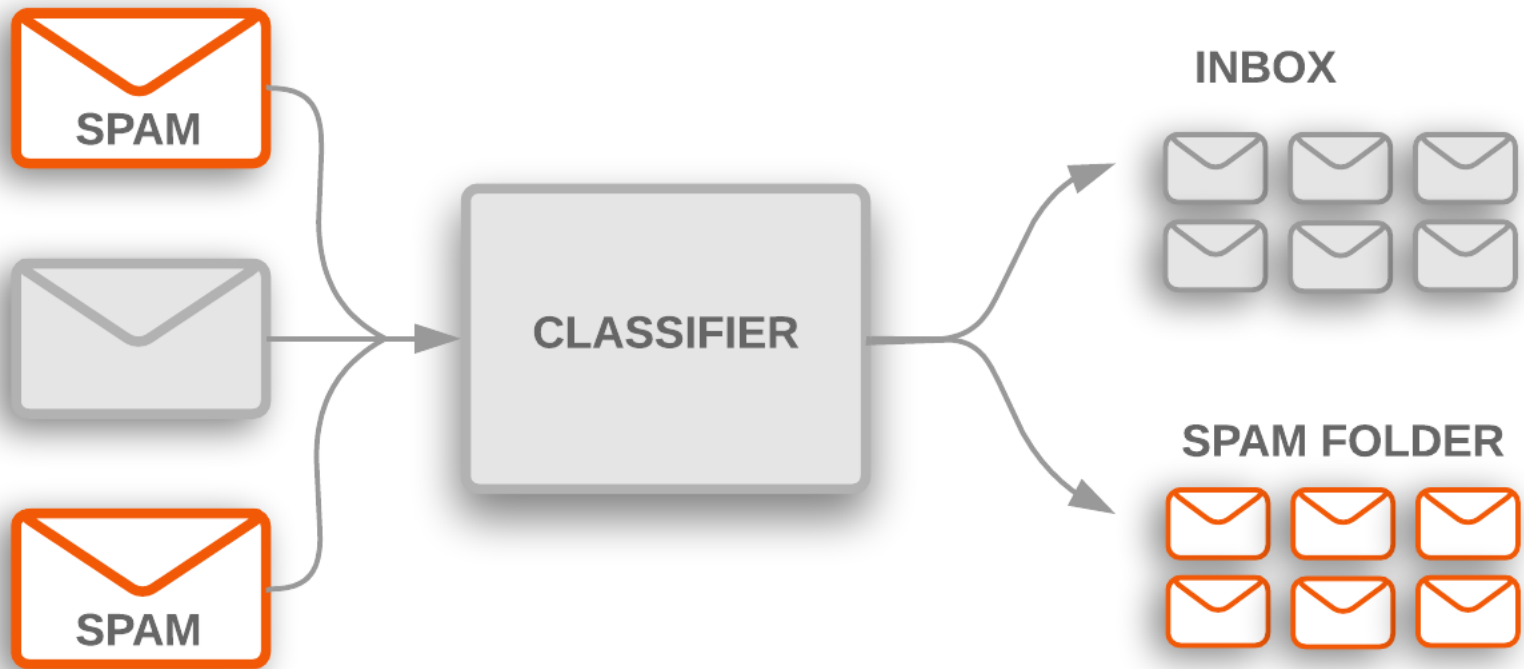
Keras preprocessing text

```
1 # keras.preprocessing.text Tokenizer
2 from keras.preprocessing.text import Tokenizer
3 # define 5 documents
4 docs = ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
5 # create the tokenizer
6 t = Tokenizer()
7 # fit the tokenizer on the documents
8 t.fit_on_texts(docs)
9 print('docs:', docs)
10 print('word_counts:', t.word_counts)
11 print('document_count:', t.document_count)
12 print('word_index:', t.word_index)
13 print('word_docs:', t.word_docs)
14 # integer encode documents
15 texts_to_matrix = t.texts_to_matrix(docs, mode='count')
16 print('texts_to_matrix:')
17 print(texts_to_matrix)
```

Using TensorFlow backend.

```
docs: ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
word_counts: OrderedDict([('well', 1), ('done', 1), ('good', 1), ('work', 2), ('great', 1), ('effort', 1), ('nice', 1), ('excellent', 1)])
document_count: 5
word_index: {'work': 1, 'well': 2, 'done': 3, 'good': 4, 'great': 5, 'effort': 6, 'nice': 7, 'excellent': 8}
word_docs: {'done': 1, 'well': 1, 'work': 2, 'good': 1, 'great': 1, 'effort': 1, 'nice': 1, 'excellent': 1}
texts_to_matrix:
[[0. 0. 1. 1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 0.]
 [0. 1. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1.]]
```

Text Classification

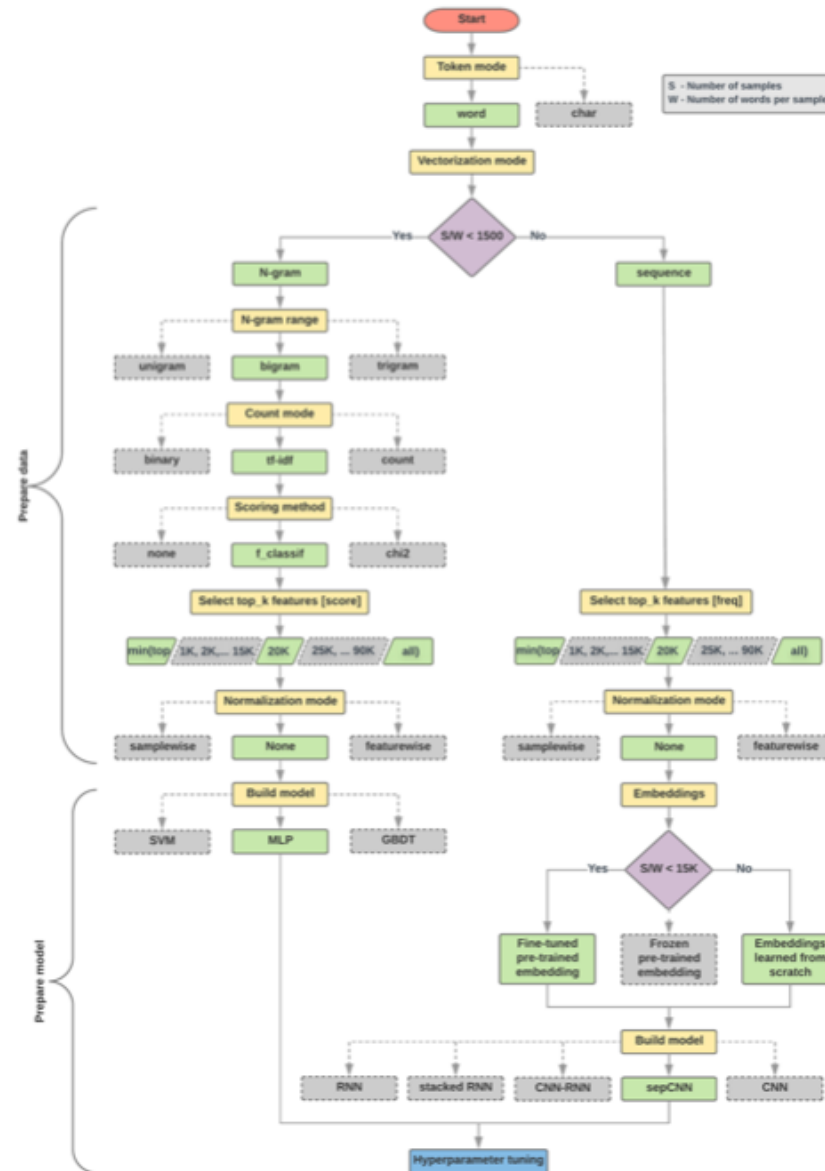


Text Classification Workflow

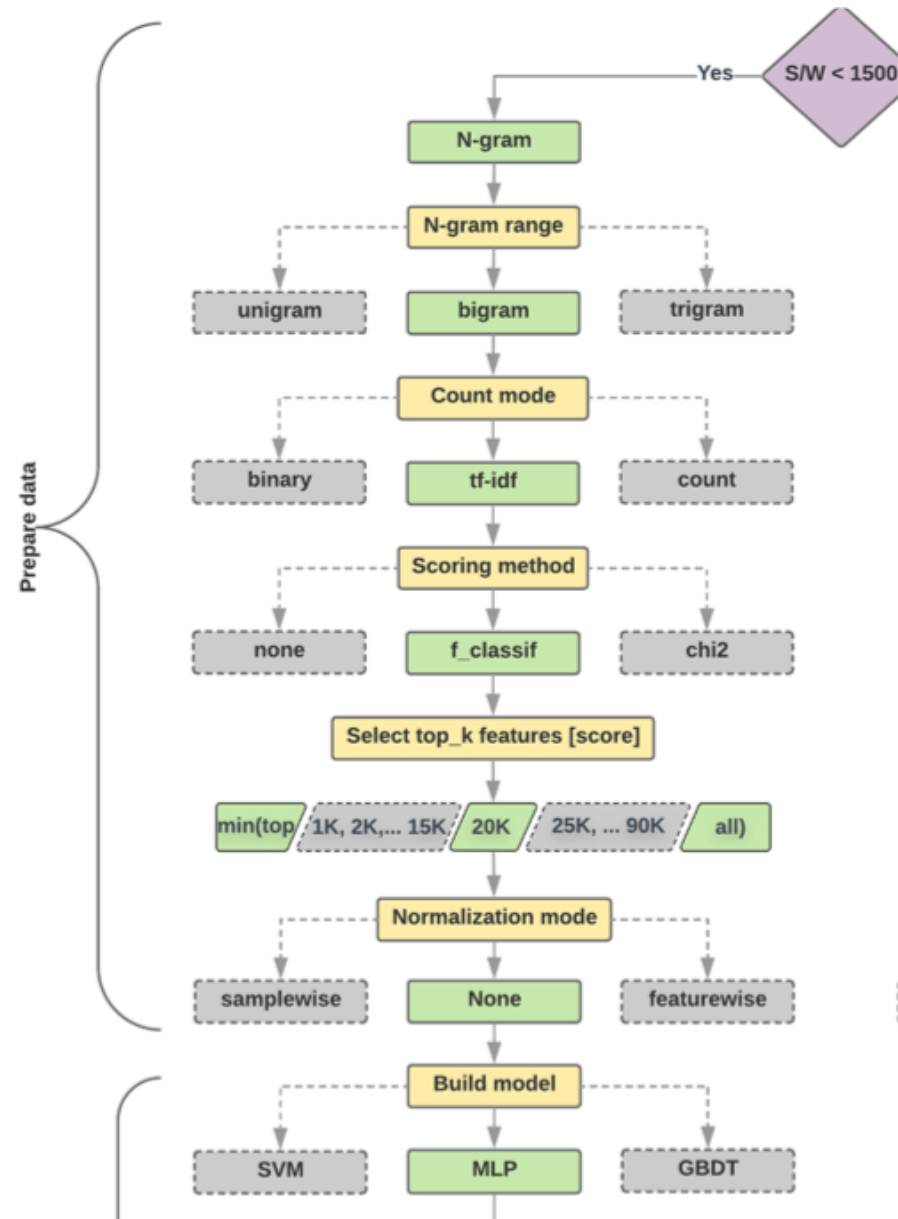
- Step 1: Gather Data
- Step 2: Explore Your Data
- Step 2.5: Choose a Model*
- Step 3: Prepare Your Data
- Step 4: Build, Train, and Evaluate Your Model
- Step 5: Tune Hyperparameters
- Step 6: Deploy Your Model



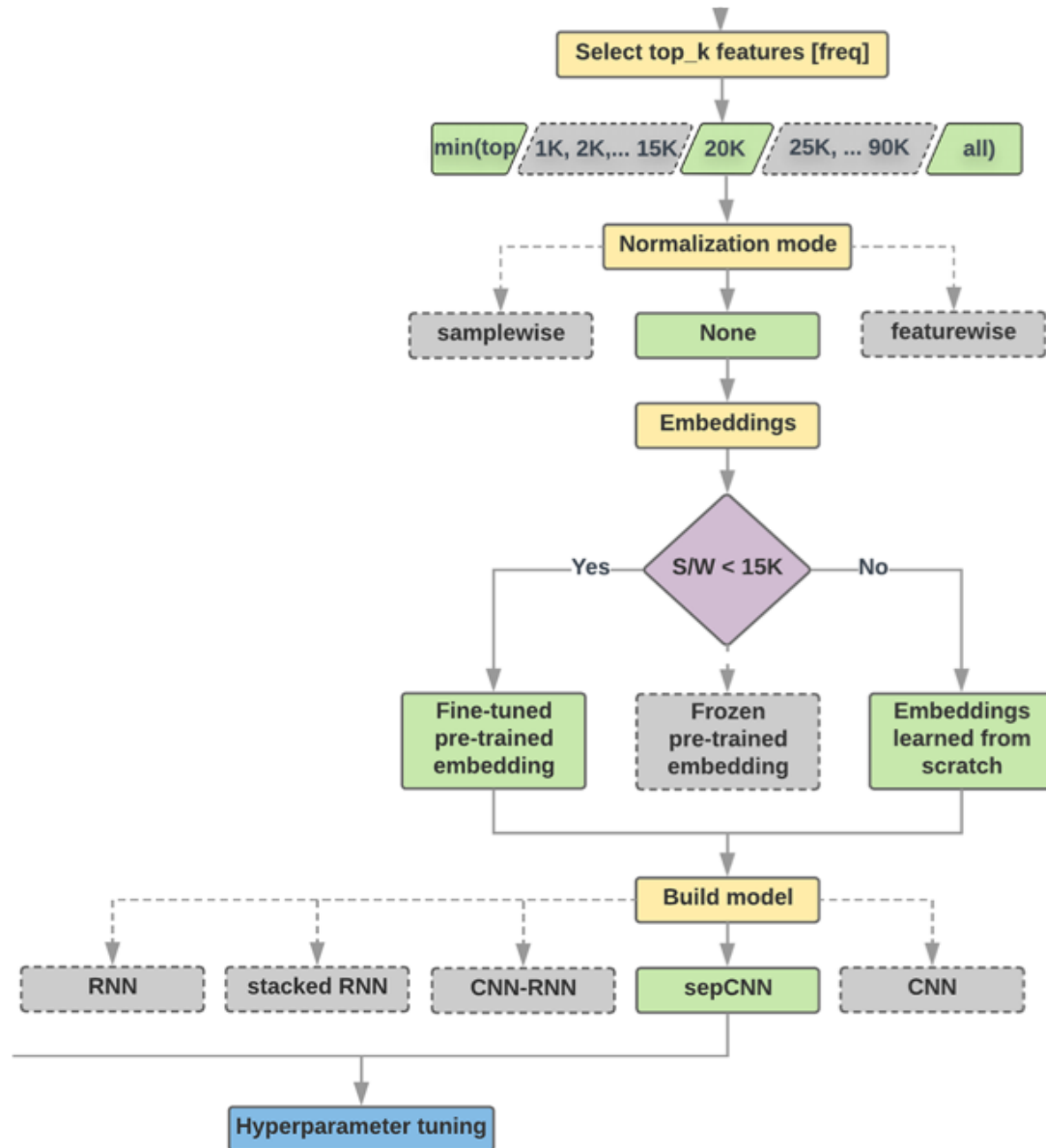
Text Classification Flowchart



Text Classification S/W<1500: N-gram



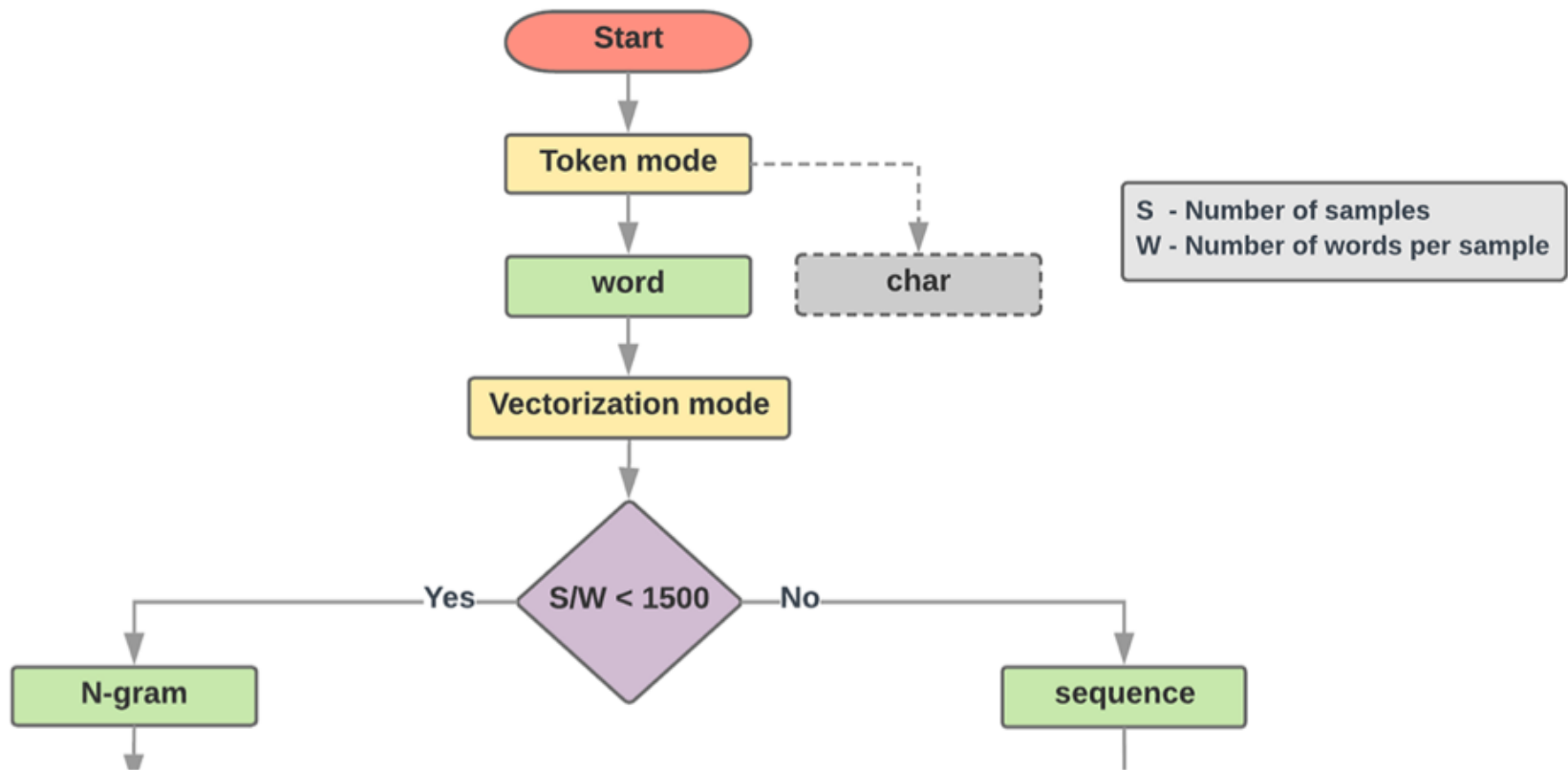
Text Classification $S/W \geq 1500$: Sequence



Step 2.5: Choose a Model

Samples/Words < 1500

$$150,000/100 = 1500$$

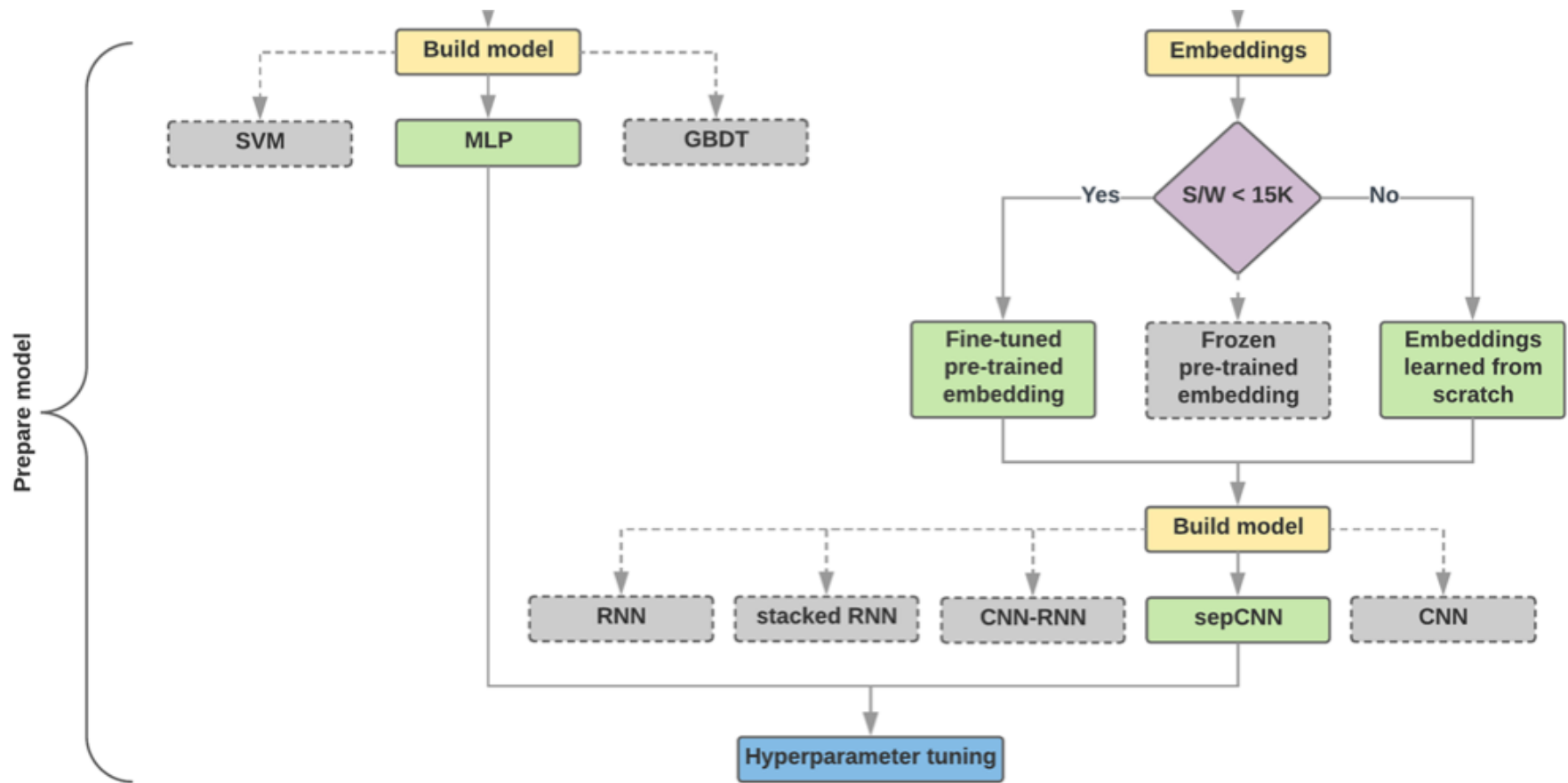


IMDb review dataset,
the samples/words-per-sample ratio is ~ 144

Step 2.5: Choose a Model

Samples/Words < 15,000

1,500,000/100 = 15,000



Step 3: Prepare Your Data

Texts:

T1: 'The mouse ran up the clock'

T2: 'The mouse ran down'

Token Index:

```
{'the': 1, 'mouse': 2, 'ran': 3, 'up': 4, 'clock': 5, 'down': 6,}.
```

NOTE: 'the' occurs most frequently,
so the index value of 1 is assigned to it.
Some libraries reserve index 0 for unknown tokens,
as is the case here.

Sequence of token indexes:

T1: 'The mouse ran up the clock' =
[1, 2, 3, 4, 1, 5]

T2: 'The mouse ran down' =
[1, 2, 3, 6]

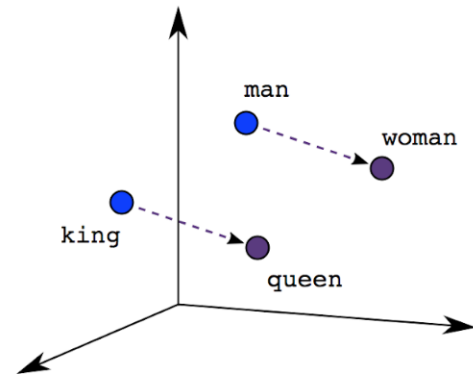
One-hot encoding

'The mouse ran up the clock' =

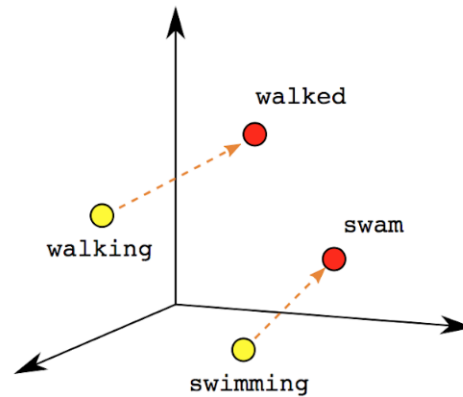
The	1	[	[0, 1, 0, 0, 0, 0, 0],
mouse	2		[0, 0, 1, 0, 0, 0, 0],
ran	3		[0, 0, 0, 1, 0, 0, 0],
up	4		[0, 0, 0, 0, 1, 0, 0],
the	1		[0, 1, 0, 0, 0, 0, 0],
clock	5		[0, 0, 0, 0, 0, 1, 0]]

[0, 1, 2, 3, 4, 5, 6]

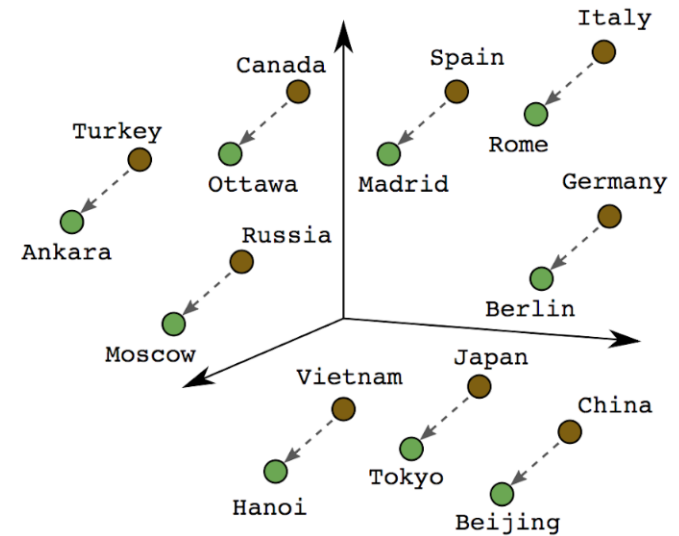
Word embeddings



Male-Female

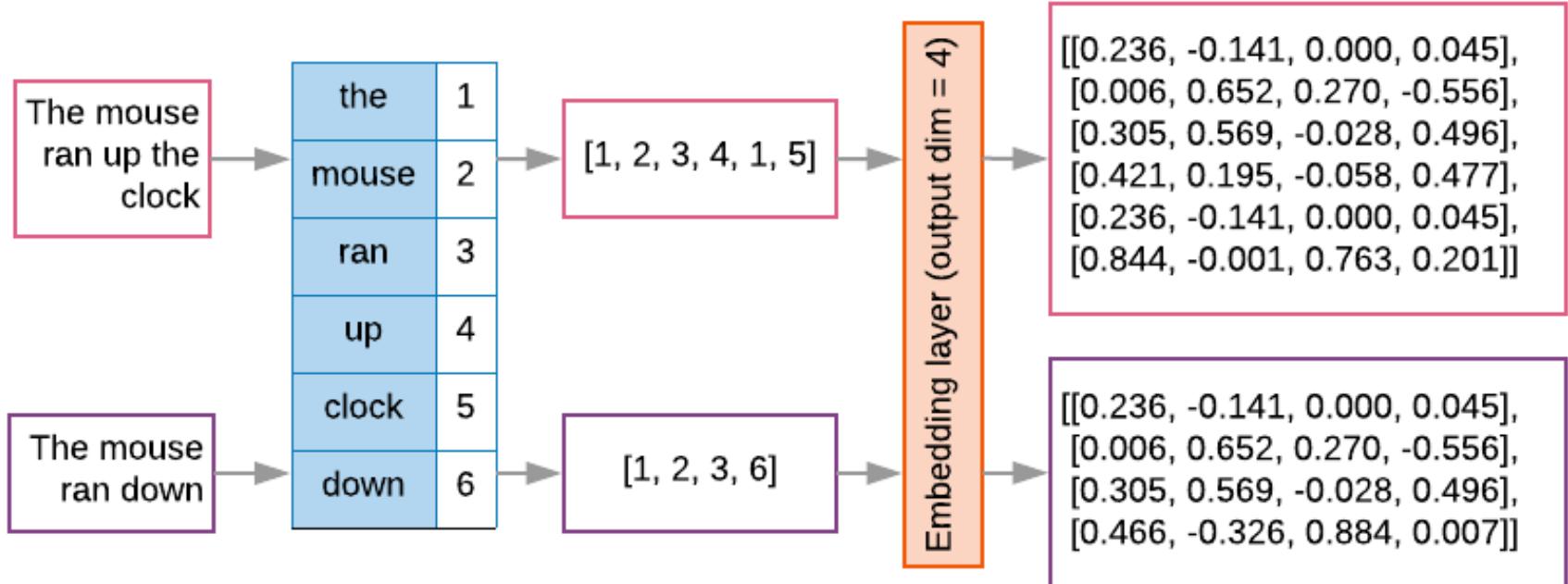


Verb Tense



Country-Capital

Word embeddings



```
t1 = 'The mouse ran up the clock'
t2 = 'The mouse ran down'
s1 = t1.lower().split(' ')
s2 = t2.lower().split(' ')
terms = s1 + s2
sortedset = sorted(set(terms))
print('terms =', terms)
print('sortedset =', sortedset)
```

```
1 t1 = 'The mouse ran up the clock'
2 t2 = 'The mouse ran down'
3 s1 = t1.lower().split(' ')
4 s2 = t2.lower().split(' ')
5 terms = s1 + s2
6 sortedset = sorted(set(terms))
7 print('terms =', terms)
8 print('sortedset =', sortedset)
```

```
terms = ['the', 'mouse', 'ran', 'up', 'the', 'clock', 'the', 'mouse', 'ran', 'down']
sortedset = ['clock', 'down', 'mouse', 'ran', 'the', 'up']
```



```
t1 = 'The mouse ran up the clock'
t2 = 'The mouse ran down'
s1 = t1.lower().split(' ')
s2 = t2.lower().split(' ')
terms = s1 + s2
print(terms)

tfdict = {}
for term in terms:
    if term not in tfdict:
        tfdict[term] = 1
    else:
        tfdict[term] += 1

a = []
for k,v in tfdict.items():
    a.append('{} {}'.format(k,v))
print(a)
```

```
['the', 'mouse', 'ran', 'up', 'the', 'clock', 'the', 'mouse', 'ran', 'down']
['the', 3, 'mouse', 2, 'ran', 2, 'up', 1, 'clock', 1, 'down', 1]
```

```
sorted_by_value_reverse = sorted(tfdict.items(),
key=lambda kv: kv[1], reverse=True)
```

```
sorted_by_value_reverse_dict =
dict(sorted_by_value_reverse)
```

```
id2word = {id: word for id, word in
enumerate(sorted_by_value_reverse_dict)}
```

```
word2id = dict([(v, k) for (k, v) in
id2word.items()])
```

```
sorted_by_value: [('up', 1), ('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3)]
sorted_by_value2: ['the', 'mouse', 'ran', 'up', 'clock', 'down']
sorted_by_value_reverse: [('the', 3), ('mouse', 2), ('ran', 2), ('up', 1), ('clock', 1), ('down', 1)]
sorted_by_value_reverse_dict {'the': 3, 'mouse': 2, 'ran': 2, 'up': 1, 'clock': 1, 'down': 1}
id2word {0: 'the', 1: 'mouse', 2: 'ran', 3: 'up', 4: 'clock', 5: 'down'}
word2id {'the': 0, 'mouse': 1, 'ran': 2, 'up': 3, 'clock': 4, 'down': 5}
len_words: 6
sorted_by_key: [('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3), ('up', 1)]
the, 3
mouse, 2
ran, 2
up, 1
clock, 1
down, 1
```

```

sorted_by_value = sorted(tfdict.items(), key=lambda kv: kv[1])
print('sorted_by_value: ', sorted_by_value)
sorted_by_value2 = sorted(tfdict, key=tfdict.get, reverse=True)
print('sorted_by_value2: ', sorted_by_value2)
sorted_by_value_reverse = sorted(tfdict.items(), key=lambda kv: kv[1], reverse=True)
print('sorted_by_value_reverse: ', sorted_by_value_reverse)
sorted_by_value_reverse_dict = dict(sorted_by_value_reverse)
print('sorted_by_value_reverse_dict', sorted_by_value_reverse_dict)
id2word = {id: word for id, word in enumerate(sorted_by_value_reverse_dict)}
print('id2word', id2word)
word2id = dict([(v, k) for (k, v) in id2word.items()])
print('word2id', word2id)
print('len_words:', len(word2id))

sorted_by_key = sorted(tfdict.items(), key=lambda kv: kv[0])
print('sorted_by_key: ', sorted_by_key)

tfstring = '\n'.join(a)
print(tfstring)
tf = tfdict.get('mouse')
print(tf)

```

```

sorted_by_value:  [('up', 1), ('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3)]
sorted_by_value2:  ['the', 'mouse', 'ran', 'up', 'clock', 'down']
sorted_by_value_reverse:  [('the', 3), ('mouse', 2), ('ran', 2), ('up', 1), ('clock', 1), ('down', 1)]
sorted_by_value_reverse_dict {'the': 3, 'mouse': 2, 'ran': 2, 'up': 1, 'clock': 1, 'down': 1}
id2word {0: 'the', 1: 'mouse', 2: 'ran', 3: 'up', 4: 'clock', 5: 'down'}
word2id {'the': 0, 'mouse': 1, 'ran': 2, 'up': 3, 'clock': 4, 'down': 5}
len_words: 6
sorted_by_key:  [('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3), ('up', 1)]
the, 3
mouse, 2
ran, 2
up, 1
clock, 1
down, 1

```

from keras.preprocessing.text import Tokenizer

```
1 from keras.preprocessing.text import Tokenizer
2 # define 5 documents
3 docs = ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
4 # create the tokenizer
5 t = Tokenizer()
6 # fit the tokenizer on the documents
7 t.fit_on_texts(docs)
8 print('docs:', docs)
9 print('word_counts:', t.word_counts)
10 print('document_count:', t.document_count)
11 print('word_index:', t.word_index)
12 print('word_docs:', t.word_docs)
13 # integer encode documents
14 texts_to_matrix = t.texts_to_matrix(docs, mode='count')
15 print('texts_to_matrix:')
16 print(texts_to_matrix)
```

```
docs: ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
word_counts: OrderedDict([('well', 1), ('done', 1), ('good', 1), ('work', 2), ('great', 1), ('effort', 1), ('ni
document_count: 5
word_index: {'work': 1, 'well': 2, 'done': 3, 'good': 4, 'great': 5, 'effort': 6, 'nice': 7, 'excellent': 8}
word_docs: {'done': 1, 'well': 1, 'work': 2, 'good': 1, 'great': 1, 'effort': 1, 'nice': 1, 'excellent': 1}
texts_to_matrix:
[[0. 0. 1. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

from keras.preprocessing.text import Tokenizer

```
from keras.preprocessing.text import Tokenizer
# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice
work', 'Excellent!']
# create the tokenizer
t = Tokenizer()
# fit the tokenizer on the documents
t.fit_on_texts(docs)
print('docs:', docs)
print('word_counts:', t.word_counts)
print('document_count:', t.document_count)
print('word_index:', t.word_index)
print('word_docs:', t.word_docs)
# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='count')
print('texts_to_matrix:')
print(texts_to_matrix)
```

```
texts_to_matrix =  
t.texts_to_matrix(docs, mode='count')
```

```
docs: ['Well done!', 'Good work', 'Great effort',  
       'nice work', 'Excellent!']  
word_counts: OrderedDict([('well', 1), ('done', 1),  
                           ('good', 1), ('work', 2), ('great', 1), ('effort', 1),  
                           ('nice', 1), ('excellent', 1)])  
document_count: 5  
word_index: {'work': 1, 'well': 2, 'done': 3, 'good':  
4, 'great': 5, 'effort': 6, 'nice': 7, 'excellent': 8}  
word_docs: {'done': 1, 'well': 1, 'work': 2, 'good': 1,  
            'great': 1, 'effort': 1, 'nice': 1, 'excellent': 1}  
texts_to_matrix:  
[[0. 0. 1. 1. 0. 0. 0. 0. 0.]  
 [0. 1. 0. 0. 1. 0. 0. 0. 0.]  
 [0. 0. 0. 0. 0. 1. 1. 0. 0.]  
 [0. 1. 0. 0. 0. 0. 0. 1. 0.]  
 [0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

`t.texts_to_matrix(docs, mode='tfidf')`

```
from keras.preprocessing.text import Tokenizer
# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice work',
        'Excellent!']
# create the tokenizer
t = Tokenizer()
# fit the tokenizer on the documents
t.fit_on_texts(docs)
print('docs:', docs)
print('word_counts:', t.word_counts)
print('document_count:', t.document_count)
print('word_index:', t.word_index)
print('word_docs:', t.word_docs)
# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='tfidf')
print('texts_to_matrix:')
print(texts_to_matrix)
```

```
texts_to_matrix:
[[0.  0.  1.25276297  1.25276297  0.  0.  0.  0.  0. ]
 [0.  0.98082925  0.  0.  1.25276297  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  1.25276297  1.25276297  0.  0. ]
 [0.  0.98082925  0.  0.  0.  0.  0.  1.25276297  0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  1.25276297]]
```

NLTK (Natural Language Toolkit)

NLTK 3.0 documentation

[NEXT](#) | [MODULES](#) | [INDEX](#)

Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to [over 50 corpora and lexical resources](#) such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active [discussion forum](#).

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

[Natural Language Processing with Python](#) provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more. The book is being updated for Python 3 and NLTK 3. (The original Python 2 version is still available at http://nltk.org/book_1ed.)

Some simple things you can do with NLTK

Tokenize and tag some text:

```
>>> import nltk
```

TABLE OF CONTENTS

[NLTK News](#)[Installing NLTK](#)[Installing NLTK Data](#)[Contribute to NLTK](#)[FAQ](#)[Wiki](#)[API](#)[HOWTO](#)

SEARCH

Enter search terms or a module, class or function name.

Python Jieba “结巴” 中文分词

GitHub, Inc. [US] <https://github.com/fxsjy/jieba> ☆

Personal Open source Business Explore Pricing Blog Support This repository Search Sign in Sign up

fxsjy / jieba

Watch 761 Star 7,187 Fork 2,252

Code Issues 226 Pull requests 14 Projects 0 Wiki Pulse Graphs

结巴中文分词

485 commits 2 branches 23 releases 31 contributors MIT

Branch: master New pull request Find file Clone or download

fxsjy committed on GitHub Merge pull request #382 from huntzhan/master Latest commit 8ba26cf on Aug 5, 2016

extra_dict	update to v0.33	2 years ago
jieba	Bugfix for HMM=False in parallelism.	6 months ago
test	Bugfix for HMM=False in parallelism.	6 months ago
.gitattributes	first commit	4 years ago
.gitignore	update jieba3k	2 years ago
Changelog	version change 0.38	a year ago
LICENSE	add a license file	4 years ago
MANIFEST.in	include Changelog & README.md in the distribution package	4 years ago
README.md	Update README.md	8 months ago

<https://github.com/fxsjy/jieba>

Python Jieba “结巴” 中文分词

```
import jieba
import jieba.posseg as pseg
sentence = "銀行產業正在改變，金融機構欲挖角科技人才"
words = jieba.cut(sentence)
print(sentence)
print(" ".join(words))
wordspos = pseg.cut(sentence)
result = ''
for word, pos in wordspos:
    print(word + ' (' + pos + ')')
    result = result + ' ' + word + ' (' + pos + ')'
print(result.strip())
```

import jieba

words = jieba.cut(sentence)

```
import jieba
import jieba.posseg as pseg
sentence = "銀行產業正在改變，金融機構欲挖角科技人才"
words = jieba.cut(sentence)
print(sentence)
print(" ".join(words))    #銀行 產業 正在 改變 ， 金融 機構 欲 挖角 科技人才

wordspos = pseg.cut(sentence)
result = ''
for word, pos in wordspos:
    print(word + ' (' + pos + ')')
    result = result + ' ' + word + ' (' + pos + ')'
print(result.strip())    #銀行(n) 產業(n) 正在(t) 改變(v) ，(x) 金融(n) 機構(n) 欲(d) 挖角(n) 科技人才(n)
```

銀行產業正在改變，金融機構欲挖角科技人才

銀行 產業 正在 改變 ， 金融 機構 欲 挖角 科技人才

銀行 (n)

產業 (n)

正在 (t)

改變 (v)

， (x)

金融 (n)

機構 (n)

欲 (d)

挖角 (n)

科技人才 (n)

銀行(n) 產業(n) 正在(t) 改變(v) ，(x) 金融(n) 機構(n) 欲(d) 挖角(n) 科技人才(n)



+ CODE + TEXT ↑ CELL ↓ CELL



```
1 import jieba
2 import jieba.posseg as pseg
3 sentence = "銀行產業正在改變，金融機構欲挖角科技人才"
4 words = jieba.cut(sentence)
5 print(sentence)
6 print(" ".join(words))
7 wordspos = pseg.cut(sentence)
8 result = ''
9 for word, pos in wordspos:
10     print(word + ' (' + pos + ')')
11     result = result + ' ' + word + ' (' + pos + ') '
12 print(result.strip())
```



銀行產業正在改變，金融機構欲挖角科技人才

銀行 產業 正在 改變 ， 金融 機構 欲 挖角 科技人才

銀行 (n)

產業 (n)

正在 (t)

改變 (v)

， (x)

金融 (n)

機構 (n)

欲 (d)

挖角 (n)

科技人才 (n)

銀行(n) 產業(n) 正在(t) 改變(v) ，(x) 金融(n) 機構(n) 欲(d) 挖角(n) 科技人才(n)

Python Jieba “结巴” 中文分词

- <https://github.com/fxsjy/jieba>
- `jieba.set_dictionary('data/dict.txt.big')`
 - `#/anaconda/lib/python3.5/site-packages/jieba`
 - `dict.txt` (5.4MB)(349,046)
 - `dict.txt.big.txt` (8.6MB)(584,429)
 - `dict.txt.small.txt` (1.6MB)(109,750)
 - `dict.tw.txt` (4.2MB)(308,431)
- https://github.com/ldkrshi/jieba-zh_TW
 - 结巴中文斷詞台灣繁體版本

TensorFlow NLP Examples

- Basic Text Classification
(Text Classification) (46 Seconds)
 - https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb
- NMT with Attention
(20-30 minutes)
 - https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/nmt_with_attention/nmt_with_attention.ipynb

Text Classification

IMDB Movie Reviews

https://colab.research.google.com/drive/1x16h1GhHsLlrLYtPCvCHaoO1W-i_gror



tf02_basic-text-classification.ipynb ☆

File Edit View Insert Runtime Tools Help

+ CODE + TEXT ↑ CELL ↓ CELL

CONNECT ▾ EDITING ▾

COMMENT SHARE

A

Table of contents Code snippets Files X

Copyright 2018 The TensorFlow Authors.

Licensed under the Apache License, Version 2.0 (the "License");

MIT License

Text classification with movie reviews

Download the IMDB dataset

Explore the data

Convert the integers back to words

Prepare the data

Build the model

Hidden units

Loss function and optimizer

Create a validation set

Train the model

Evaluate the model

► Copyright 2018 The TensorFlow Authors.

↳ 2 cells hidden

▼ Text classification with movie reviews

 [View on TensorFlow.org](#)  [Run in Google Colab](#)  [View source on GitHub](#)

This notebook classifies movie reviews as *positive* or *negative* using the text of the review. This is an example of *binary*—or two-class—classification, an important and widely applicable kind of machine learning problem.

We'll use the [IMDB dataset](#) that contains the text of 50,000 movie reviews from the [Internet Movie Database](#). These are split into 25,000 reviews for training and 25,000 reviews for testing. The training and testing sets are *balanced*, meaning they contain an equal number of positive and negative reviews.

This notebook uses [tf.keras](#), a high-level API to build and train models in TensorFlow. For a more advanced text classification tutorial using `tf.keras`, see the [MLCC Text Classification Guide](#).



```
1 # memory footprint support libraries/code
2 !ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
3 !pip install gputil
4 !pip install psutil
5 !pip install humanize
6 import psutil
7 import humanize
8 import os
9 import GPUtil as GPU
10 GPUs = GPU.getGPUs()
11 gpu = GPUs[0]
12 def printm():
13     process = psutil.Process(os.getpid())
```

Source: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb

Summary

- Python for
Natural Language Processing

References

- Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson.
- Rajesh Arumugam (2018), Hands-On Natural Language Processing with Python: A practical guide to applying deep learning architectures to your NLP applications, Packt.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805.
- Christopher D. Manning and Hinrich Schütze (1999), Foundations of Statistical Natural Language Processing, The MIT Press.
- Dipanjan Sarkar (2016), Text Analytics with Python: A Practical Real-World Approach to Gaining Actionable Insights from your Data, Apress.
- Jake VanderPlas (2016), Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly Media.
- Steven Bird, Ewan Klein and Edward Loper (2009), Natural Language Processing with Python, O'Reilly Media, <http://www.nltk.org/book/> , http://www.nltk.org/book_1ed/
- Nitin Hardeniya (2015), NLTK Essentials, Packt.
- Bing Liu (2009), Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data, Springer.