

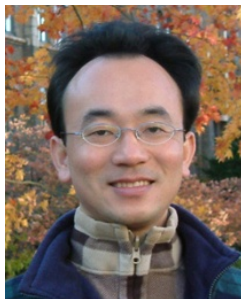
人工智慧文本分析 (AI for Text Analytics)

Python 自然語言處理 (Python for Natural Language Processing)

1091AITA03

MBA, IMTKU (M2455) (8418) (Fall 2020)

Thu 3, 4 (10:10-12:00) (B206)



Min-Yuh Day

戴敏育

Associate Professor

副教授

Institute of Information Management, National Taipei University

國立臺北大學 資訊管理研究所

<https://web.ntpu.edu.tw/~myday>

2020-10-08



課程大綱 (Syllabus)

- | 週次 (Week) | 日期 (Date) | 內容 (Subject/Topics) |
|-----------|------------|---|
| 1 | 2020/09/17 | 人工智慧文本分析課程介紹
(Course Orientation on Artificial Intelligence for Text Analytics) |
| 2 | 2020/09/24 | 文本分析的基礎：自然語言處理
(Foundations of Text Analytics: Natural Language Processing; NLP) |
| 3 | 2020/10/01 | 中秋節 (Mid-Autumn Festival) 放假一天 (Day off) |
| 4 | 2020/10/08 | Python自然語言處理
(Python for Natural Language Processing) |
| 5 | 2020/10/15 | 處理和理解文本
(Processing and Understanding Text) |
| 6 | 2020/10/22 | 文本表達特徵工程
(Feature Engineering for Text Representation) |

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
7	2020/10/29	人工智慧文本分析個案研究 I (Case Study on Artificial Intelligence for Text Analytics I)
8	2020/11/05	文本分類 (Text Classification)
9	2020/11/12	文本摘要和主題模型 (Text Summarization and Topic Models)
10	2020/11/19	期中報告 (Midterm Project Report)
11	2020/11/26	文本相似度和分群 (Text Similarity and Clustering)
12	2020/12/03	語意分析和命名實體識別 (Semantic Analysis and Named Entity Recognition; NER)

課程大綱 (Syllabus)

- | 週次 (Week) | 日期 (Date) | 內容 (Subject/Topics) |
|-----------|------------|--|
| 13 | 2020/12/10 | 情感分析
(Sentiment Analysis) |
| 14 | 2020/12/17 | 人工智慧文本分析個案研究 II
(Case Study on Artificial Intelligence for Text Analytics II) |
| 15 | 2020/12/24 | 深度學習和通用句子嵌入模型
(Deep Learning and Universal Sentence-Embedding Models) |
| 16 | 2020/12/31 | 問答系統與對話系統
(Question Answering and Dialogue Systems) |
| 17 | 2021/01/07 | 期末報告 I (Final Project Presentation I) |
| 18 | 2021/01/14 | 期末報告 II (Final Project Presentation II) |

Outline

- Python for
Natural Language Processing

Python for Natural Language Processing



Connect Google Colab in Google Drive

The image shows a browser window with the Google Drive interface. The address bar displays 'https://drive.google.com/drive/u/2/my-drive'. The main navigation bar includes the Drive logo, a search bar, and utility icons. The left sidebar contains navigation options: 'New', 'My Drive', 'Computers', 'Shared with me', 'Recent', 'Starred', 'Trash', 'Backups', and 'Storage'. The 'New' button is highlighted with a red dashed border. A dropdown menu is open from 'New', listing options: 'New folder...', 'Upload files...', 'Upload folder...', 'Google Docs', 'Google Sheets', 'Google Slides', and 'More'. The 'More' option is also highlighted with a red dashed border. A second dropdown menu is open from 'More', listing: 'Google Forms', 'Google Drawings', 'Google My Maps', 'Google Sites', and 'Connect more apps'. The 'Connect more apps' option is highlighted with a red dashed border. The main content area shows a 'Files' section with a table of file management options: 'Store safely', 'Sync seamlessly', 'Access anywhere', and 'Share easily'. A 'Name' column header is visible on the right side of the file list.

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 box	 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
Share easily

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 ↑

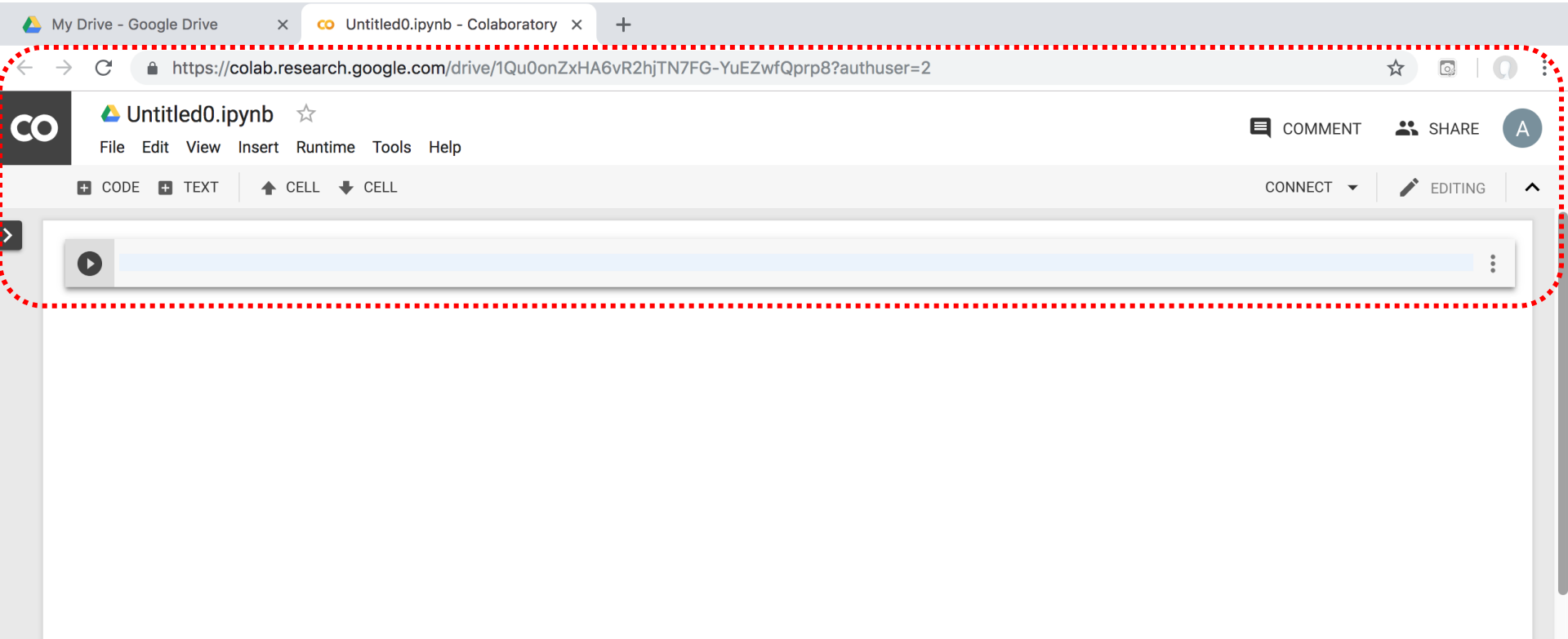
Connect Colaboratory to Google Drive

The screenshot shows a web browser window with the Google Drive interface. The address bar displays `https://drive.google.com/drive/u/2/my-drive`. The main content area is a 'Connect apps to Drive' dialog box. At the top of the dialog, there is a search bar containing the text 'colab'. Below the search bar, a notification card for Colaboratory is displayed. The notification features the Colaboratory logo (two yellow circles) and the text: 'Colaboratory was connected to Google Drive.' Below this, there is a checked checkbox and the text: 'Make Colaboratory the default app for files it can open'. At the bottom right of the notification card is a blue button with the text 'OK'. To the right of the notification card, there is a 'RATE IT' section with a green star icon, the text 'RATE IT', and a rating of 'Productivity ★★★★★ (195)'. The background of the dialog box shows a list of applications, with the Colaboratory app card partially visible on the left. The Google Drive interface is visible in the background, including the 'New' button, 'My Drive', 'Computers', 'Shared with me', 'Recent', 'Starred', 'Trash', 'Backups', and 'Storage' sections. The storage usage is shown as '0 bytes of 15 GB used' with a link to 'UPGRADE STORAGE'. At the bottom of the screen, there is a banner for 'Get Backup and Sync for Mac'.

Google Colab

The image shows a browser window with the Google Drive interface. The address bar shows the URL `https://drive.google.com/drive/u/2/my-drive`. The Drive logo is in the top left, and a search bar is in the top center. On the left side, there is a 'New' button with a plus sign. A red dashed box highlights the 'New' button and the 'My Drive' folder in the left sidebar. A dropdown menu is open from the 'New' button, listing options: 'New folder...', 'Upload files...', 'Upload folder...', 'Google Docs', 'Google Sheets', 'Google Slides', and 'More'. A red dashed box highlights the 'More' option. A second dropdown menu is open from the 'More' option, listing: 'Google Forms', 'Google Drawings', 'Google My Maps', 'Google Sites', 'Colaboratory', and 'Connect more apps'. A red dashed box highlights the 'Colaboratory' option. The background shows the 'Quick Access' section with a preview of a document and the 'Storage' section showing '0 bytes of 15 GB used' and an 'UPGRADE STORAGE' button. A 'Files' section is partially visible at the bottom.

Google Colab



Google Colab

The image shows a browser window with two tabs: "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 interface displays "Untitled0.ipynb" with a star icon. The menu bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". The "Runtime" menu is open, showing the following 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
- Manage sessions

The "Runtime" menu title and the "Change runtime type" option are highlighted with red dashed boxes. The interface also shows "CONNECT" and "EDITING" buttons on the right side.

Run Jupyter Notebook Python3 GPU 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 interface includes a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu bar, there are buttons for '+ CODE', '+ TEXT', '↑ CELL', and '↓ CELL'. On the right side, there are buttons for 'COMMENT', 'SHARE', 'CONNECT', and 'EDITING'. A 'Notebook settings' dialog box is open in the center, featuring a title bar and the following options:

- Runtime type**: A dropdown menu currently set to 'Python 3'.
- Hardware accelerator**: A dropdown menu currently set to 'GPU', with a blue question mark icon to its right.
- Omit code cell output when saving this notebook

At the bottom right of the dialog box, there are two buttons: 'CANCEL' and 'SAVE'.

Google Colab Python Hello World

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

The screenshot shows the Google Colaboratory 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 page title is "Untitled0.ipynb". The menu bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". The toolbar shows "CODE", "TEXT", "CELL", and "CELL" options. The status bar indicates "CONNECTED" and "EDITING". The code cell contains the Python code `print('Hello World')`, and the output cell displays "Hello World".

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

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.

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab notebook titled "python101.ipynb". The left sidebar contains a "Table of contents" with various NLP-related topics. The main area displays two code cells. The first cell uses spaCy to process a sentence about Steve Jobs and Wozniak, highlighting entities like "PERSON", "ORG", "DATE", and "GPE". The second cell uses spaCy and pandas to create a DataFrame of tokenized text with columns for text, lemma, pos, tag, pos_explain, and stopword. The output is a table with 8 rows of data.

```
1 text = "Steve Jobs and Steve Wozniak incorporated Apple Computer on January 3, 1977, in Cupertino, California."
2 doc = nlp(text)
3 displacy.render(doc, style="ent", jupyter=True)
```

Steve Jobs PERSON and Steve Wozniak PERSON incorporated Apple Computer ORG on January 3, 1977 DATE , in Cupertino GPE , California GPE .

```
[ ] 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", "tag", "pos_explain", "stopword")
6 rows = []
7 for t in doc:
8     row = [t.text, t.lemma_, t.pos_, t.tag_, spacy.explain(t.pos_), t.is_stop]
9     rows.append(row)
10 df = pd.DataFrame(rows, columns=cols)
11 df
```

	text	lemma	pos	tag	pos_explain	stopword
0	Stanford	Stanford	PROPN	NNP	proper noun	False
1	University	University	PROPN	NNP	proper noun	False
2	is	be	VERB	VBZ	verb	True
3	located	locate	VERB	VBN	verb	False
4	in	in	ADP	IN	adposition	True
5	California	California	PROPN	NNP	proper noun	False
6	.	.	PUNCT	.	punctuation	False
7	It	-PRON-	PRON	PRP	pronoun	True

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab notebook titled "python101.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". A "Table of contents" sidebar on the left lists various topics under "Text Analytics and Natural Language Processing (NLP)", including "Python for Natural Language Processing", "spaCy Chinese Model", "Open Chinese Convert", "Jieba", "NLTK", "Stanza", and "Text Processing and Understanding". The main notebook area displays a code cell with the following content:

```
+ 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] 1 !python -m spacy download en_core_web_sm

[3] 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 AUX 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
```

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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



python101.ipynb ☆

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

Comment Share Settings Profile

+ 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

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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



python101.ipynb ☆

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

Comment Share Settings

+ 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

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

<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.

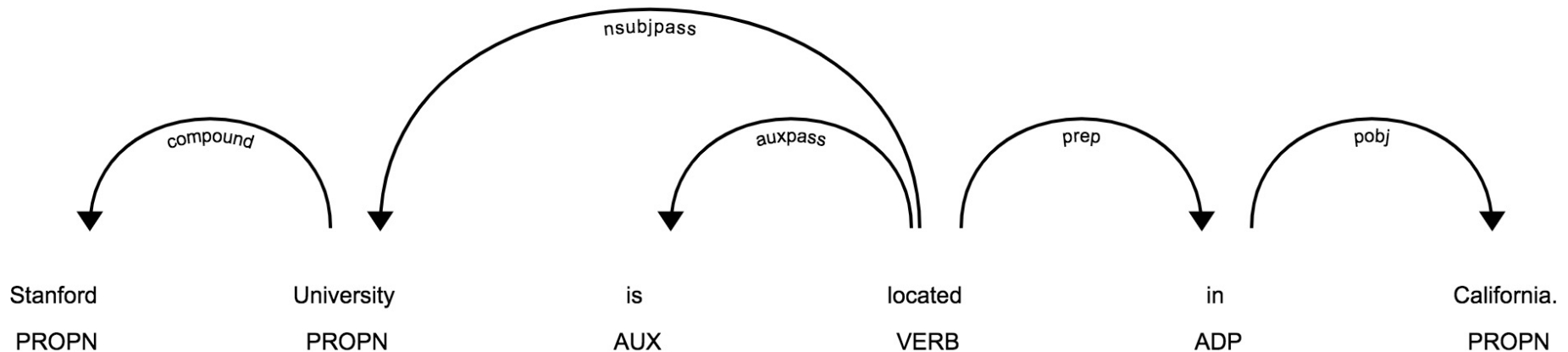
<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

```
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)
5 displacy.render(doc, style="dep", jupyter=True)
```

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



<https://tinyurl.com/aintpupython101>

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.

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab notebook titled "python101.ipynb". The left sidebar contains a "Table of contents" with various NLP topics. The main area displays two code cells. The first cell uses the `displacy.render` function to visualize the Named Entity Recognition (NER) output for a sentence about Steve Jobs and Steve Wozniak. The second cell uses the `spacy` library to load a model and process a sentence about Stanford University, with the output shown as a DataFrame table.

```
1 text = "Steve Jobs and Steve Wozniak incorporated Apple Computer on January 3, 1977, in Cupertino, California."
2 doc = nlp(text)
3 displacy.render(doc, style="ent", jupyter=True)
```

Steve Jobs PERSON and Steve Wozniak PERSON incorporated Apple Computer ORG on January 3, 1977 DATE , in Cupertino GPE , California GPE .

```
[ ] 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", "tag", "pos_explain", "stopword")
6 rows = []
7 for t in doc:
8     row = [t.text, t.lemma_, t.pos_, t.tag_, spacy.explain(t.pos_), t.is_stop]
9     rows.append(row)
10 df = pd.DataFrame(rows, columns=cols)
11 df
```

	text	lemma	pos	tag	pos_explain	stopword
0	Stanford	Stanford	PROPN	NNP	proper noun	False
1	University	University	PROPN	NNP	proper noun	False
2	is	be	VERB	VBZ	verb	True
3	located	locate	VERB	VRB	verb	False
4	in	in	ADP	IN	adposition	True
5	California	California	PROPN	NNP	proper noun	False
6	.	.	PUNCT	.	punctuation	False
7	It	-PRON-	PRON	PRP	pronoun	True

<https://tinyurl.com/aintpupython101>

MONPA 罔拍：

正體中文斷詞、詞性標註以及命名實體辨識的多任務模型

```
1 # MONPA 罔拍：正體中文斷詞、詞性標註以及命名實體辨識的多任務模型
2 # Source: https://github.com/monpa-team/monpa
3 !pip install monpa
```

```
1 import monpa
2 sentence = "銀行產業正在改變，金融機構欲挖角科技人才"
3 words = monpa.cut(sentence)
4 print(sentence)
5 print(" ".join(words))
6 result_pseg = monpa.pseg(sentence)
7 for item in result_pseg:
8     print(item)
```

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', 'ORG')
('產業', 'Na')
('正在', 'D')
('改變', 'VC')
('，', 'COMMACATEGORY')
('金融', 'Na')
('機構', 'Nc')
('欲', 'VK')
('挖角', 'VA')
('科技', 'Na')
('人才', 'Na')

jieba

words = jieba.cut(sentence)

```
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)

<https://tinyurl.com/aintpupython101>

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/ldkrssi/jieba-zh_TW
 - 结巴中文斷詞台灣繁體版本

Python in Google Colab

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

python101.ipynb ☆

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CONNECT EDITING

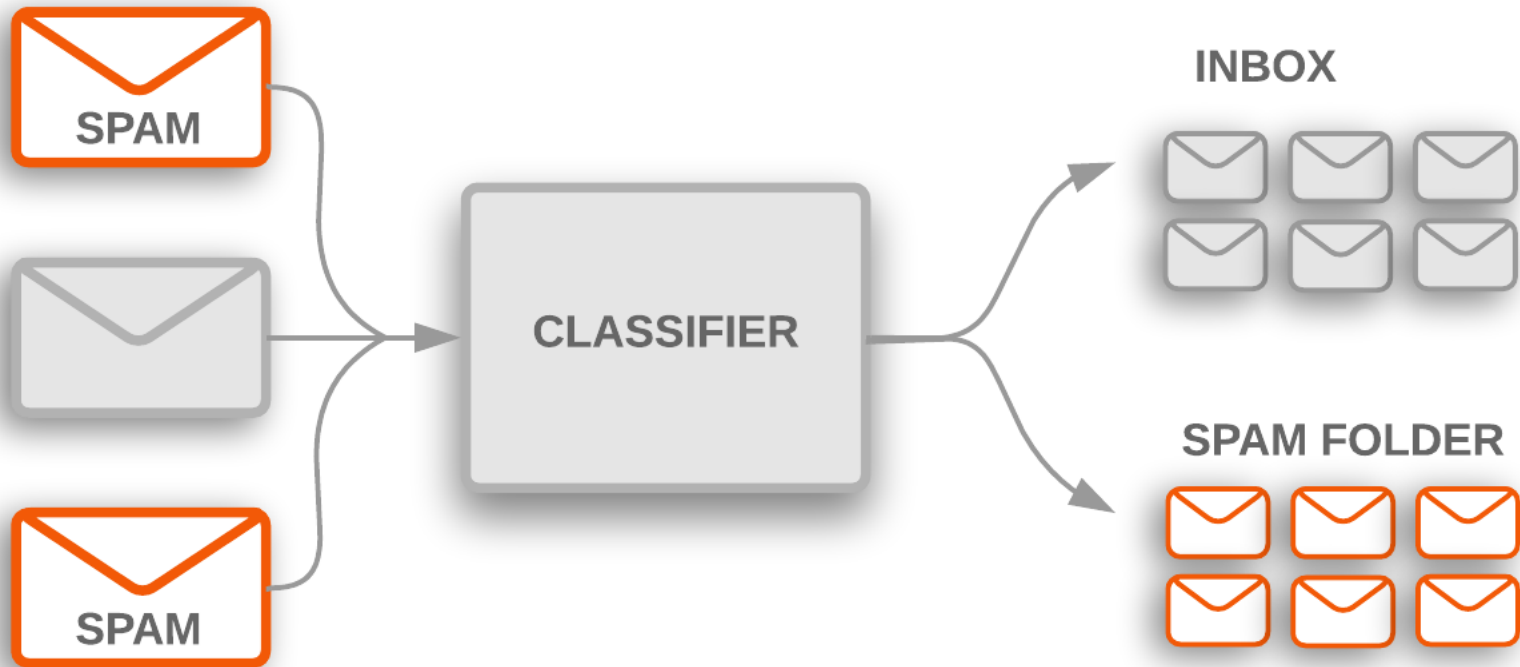
▼ Keras preprocessing text

```
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.]
 [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. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

Text Classification

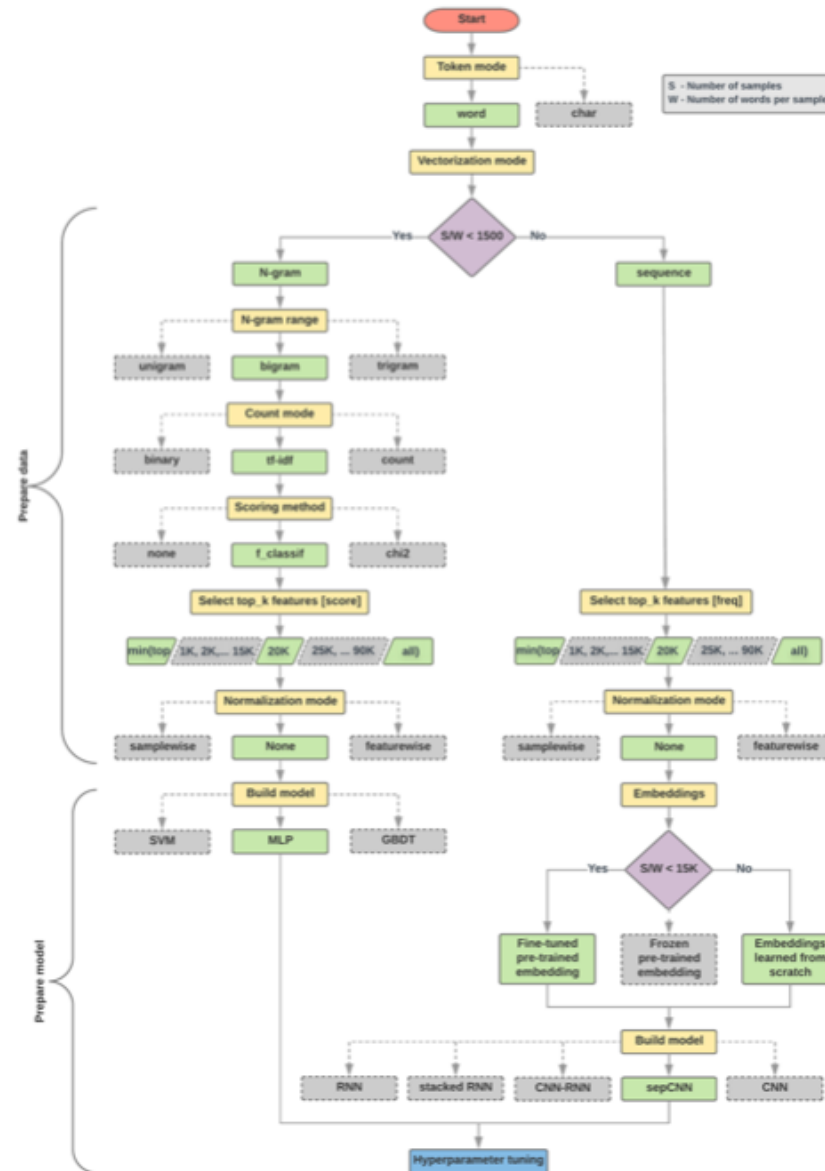


Text Classification Workflow

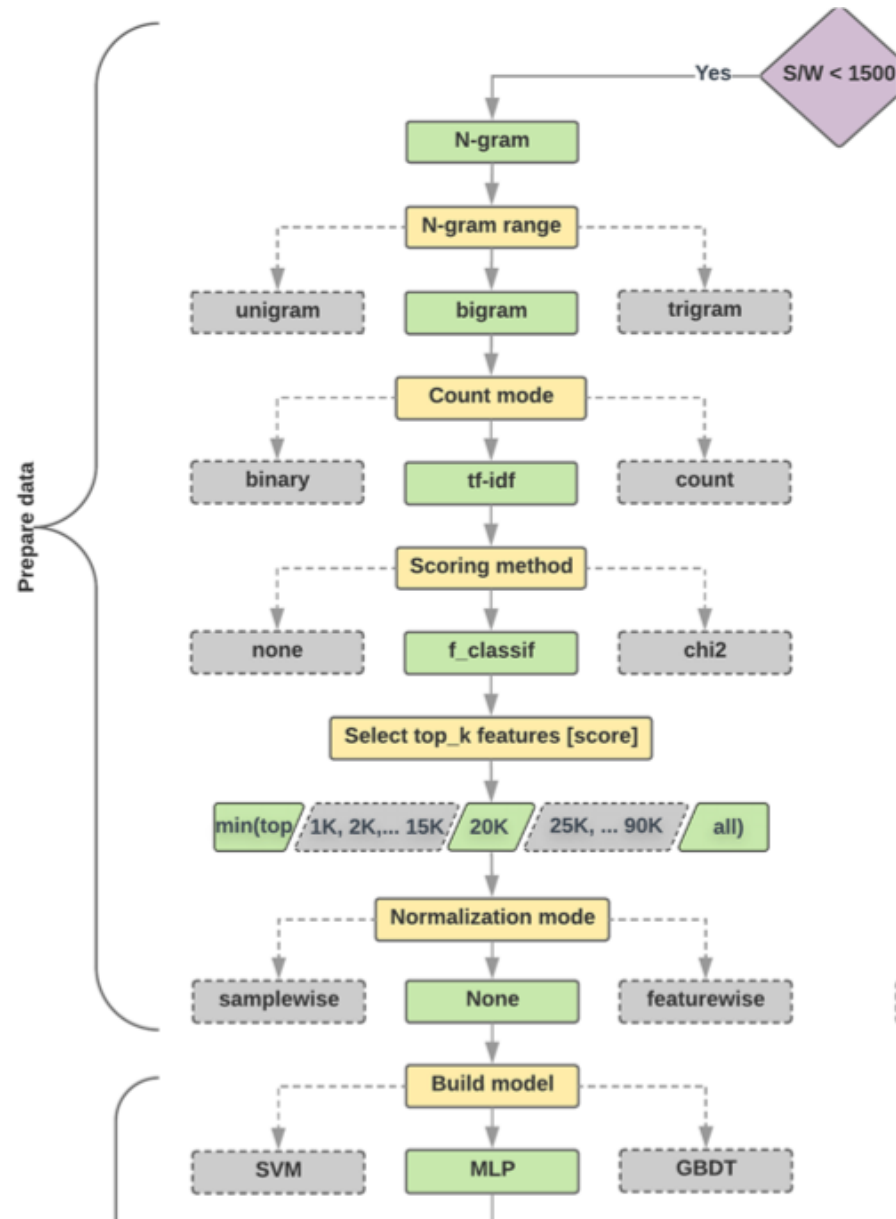
- Step 1: Gather Data
- Step 2: Explore Your Data
- Step 2.5: Choose a Model*
- Step 3: Prepare Your Data
- Step 4: Build, Train, and Evaluate Your Model
- Step 5: Tune Hyperparameters
- Step 6: Deploy Your Model



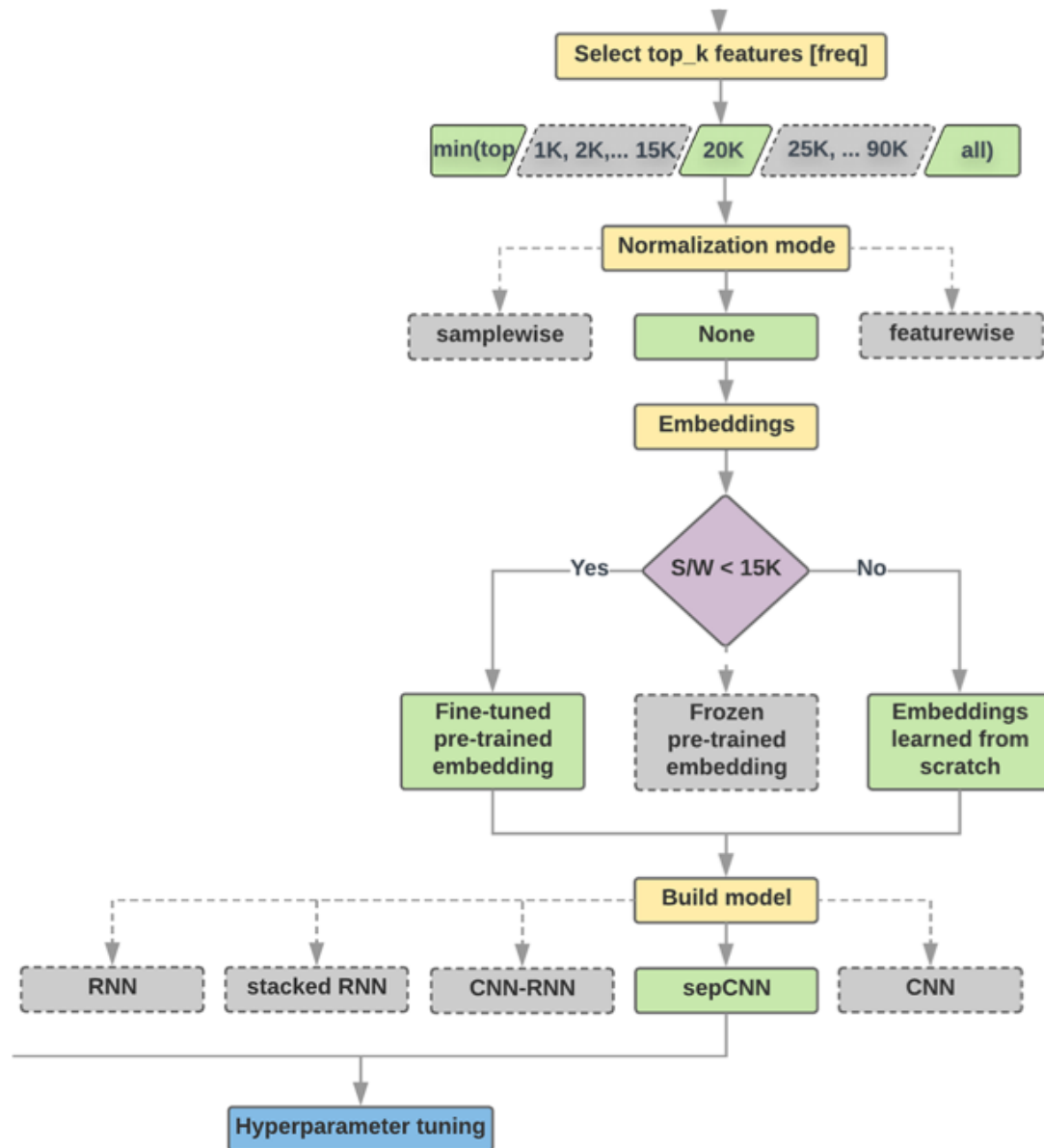
Text Classification Flowchart



Text Classification S/W<1500: N-gram



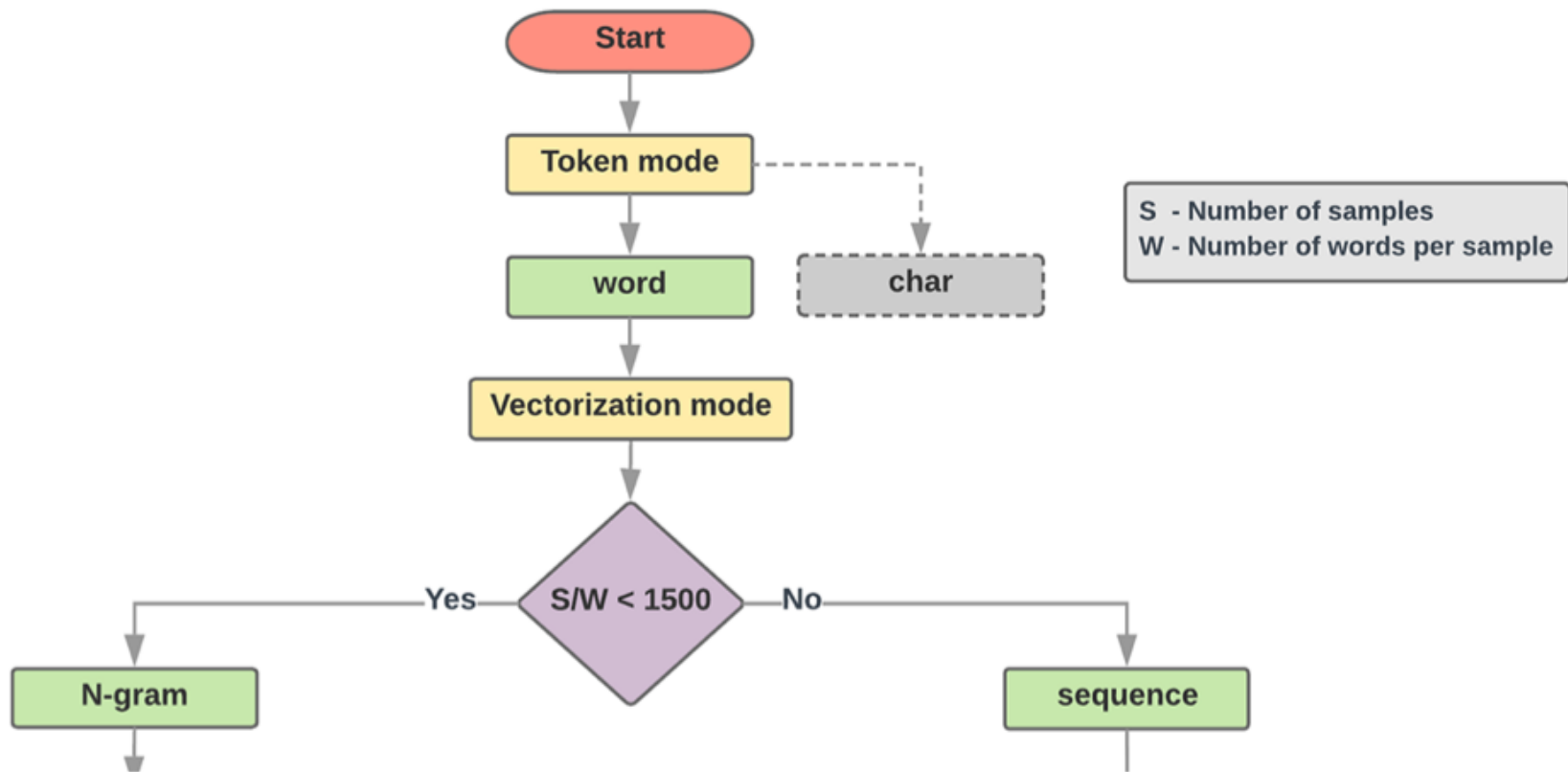
Text Classification $S/W \geq 1500$: Sequence



Step 2.5: Choose a Model

Samples/Words < 1500

$$150,000/100 = 1500$$

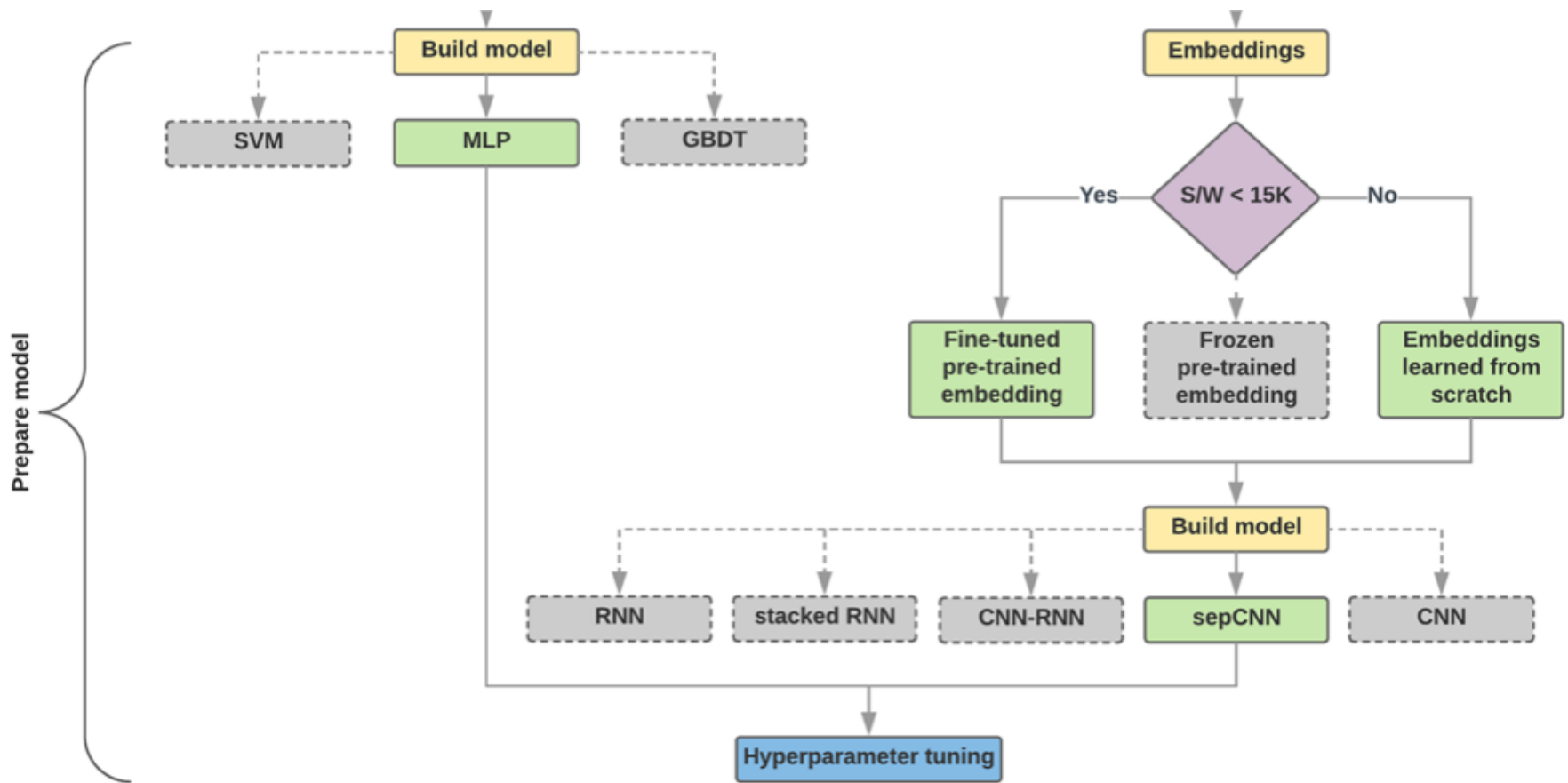


IMDb review dataset,
the samples/words-per-sample ratio is ~ 144

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]

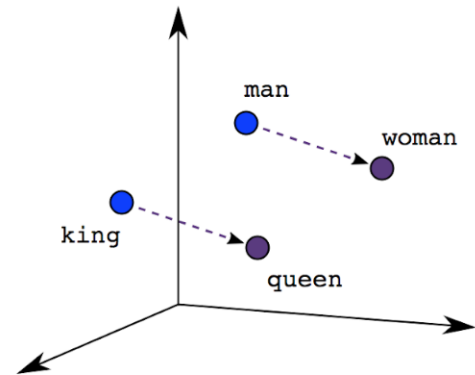
T2: 'The mouse ran down' =
[1, 2, 3, 6]

One-hot encoding

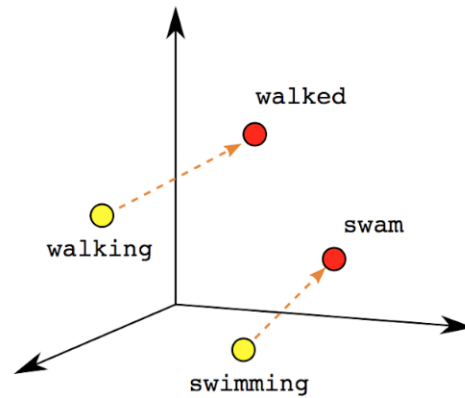
'The mouse ran up the clock' =

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]

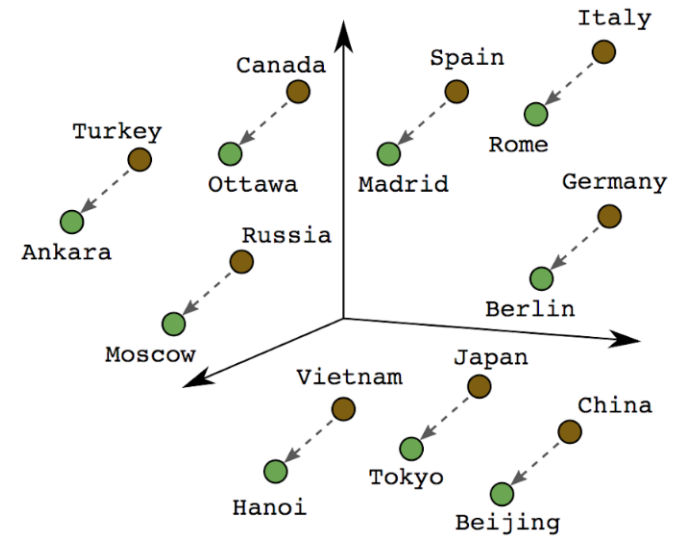
Word embeddings



Male-Female

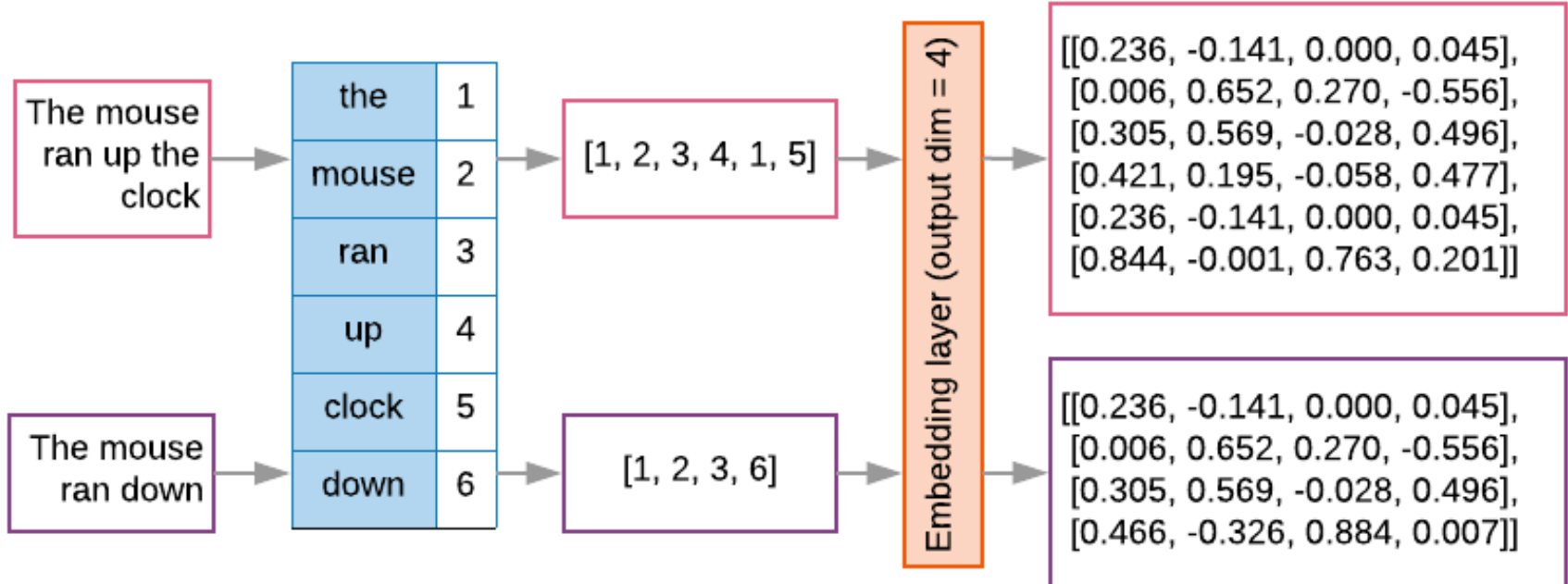


Verb Tense



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.

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/1x16h1GhHsLrLYtPCvCHaoO1W-i_gror

The screenshot shows a Google Colab notebook titled "tf02_basic-text-classification.ipynb". The interface includes a top navigation bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus. On the right, there are "COMMENT" and "SHARE" buttons. Below the navigation bar, there are buttons for "+ CODE", "+ TEXT", "↑ CELL", and "↓ CELL". A "CONNECT" dropdown and "EDITING" button are also visible.

The left sidebar contains a "Table of contents" with the following items:

- 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

The main content area shows the following sections:

- ▶ Copyright 2018 The TensorFlow Authors.
 - ↳ 2 cells hidden
- ▼ Text classification with movie reviews

Below the section header, there are three links: "View on TensorFlow.org", "Run in Google Colab", and "View source on GitHub".

The text below the links reads:

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](#).

The code cell shows the following code:

```
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.