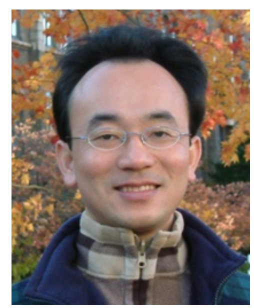
文字探勘
(Text Mining)

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

1082TM03
MBA, BDABI, TKU (E3611) (8480) (Spring 2020)
Mon, 7, 8, 9 (14:10-17:00) (B206)

Chichang Jou
周清江
Associate Professor
副教授
cjou@mail.tku.edu.tw



Min-Yuh Day
戴敏育
Associate Professor
副教授
myday@mail.tku.edu.tw

Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系

Tamkang Universit
淡江大學

2020-03-16

課程大綱 (Syllabus)

週次 (Week) 日期 (Date) 內容 (Subject/Topics)

1 2020/03/02 文字探勘課程介紹
(Course Orientation on Text Mining)

2 2020/03/09 文字探勘基礎：自然語言處理
(Foundations of Text Mining:
Natural Language Processing; NLP)

3 2020/03/16 Python自然語言處理
(Python for Natural Language Processing)

4 2020/03/23 處理和理解文本 (Processing and Understanding Text)

5 2020/03/30 文本表達特徵工程
(Feature Engineering for Text Representation)

6 2020/04/06 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 2020/04/13 文本分類
(Text Classification)

8 2020/04/20 文本摘要和主題模型
(Text Summarization and Topic Models)

9 2020/04/27 期中報告 (Midterm Project Report)

10 2020/05/04 文本相似度和分群
(Text Similarity and Clustering)

11 2020/05/11 語意分析和命名實體識別
(Semantic Analysis and Named Entity Recognition; NER)

12 2020/05/18 情感分析
(Sentiment Analysis)

課程大綱 (Syllabus)

週次 (Week)　日期 (Date)　內容 (Subject/Topics)

13 2020/05/25 人工智慧文本分析個案研究 II
(Case Study on Artificial Intelligence for Text Analytics II)

14 2020/06/01 深度學習和通用句子嵌入模型
(Deep Learning and Universal Sentence-Embedding Models)

15 2020/06/08 問答系統與對話系統
(Question Answering and Dialogue Systems)

16 2020/06/15 期末報告 I (Final Project Presentation I)

17 2020/06/22 期末報告 II (Final Project Presentation II)

18 2020/06/29 教師彈性補充教學

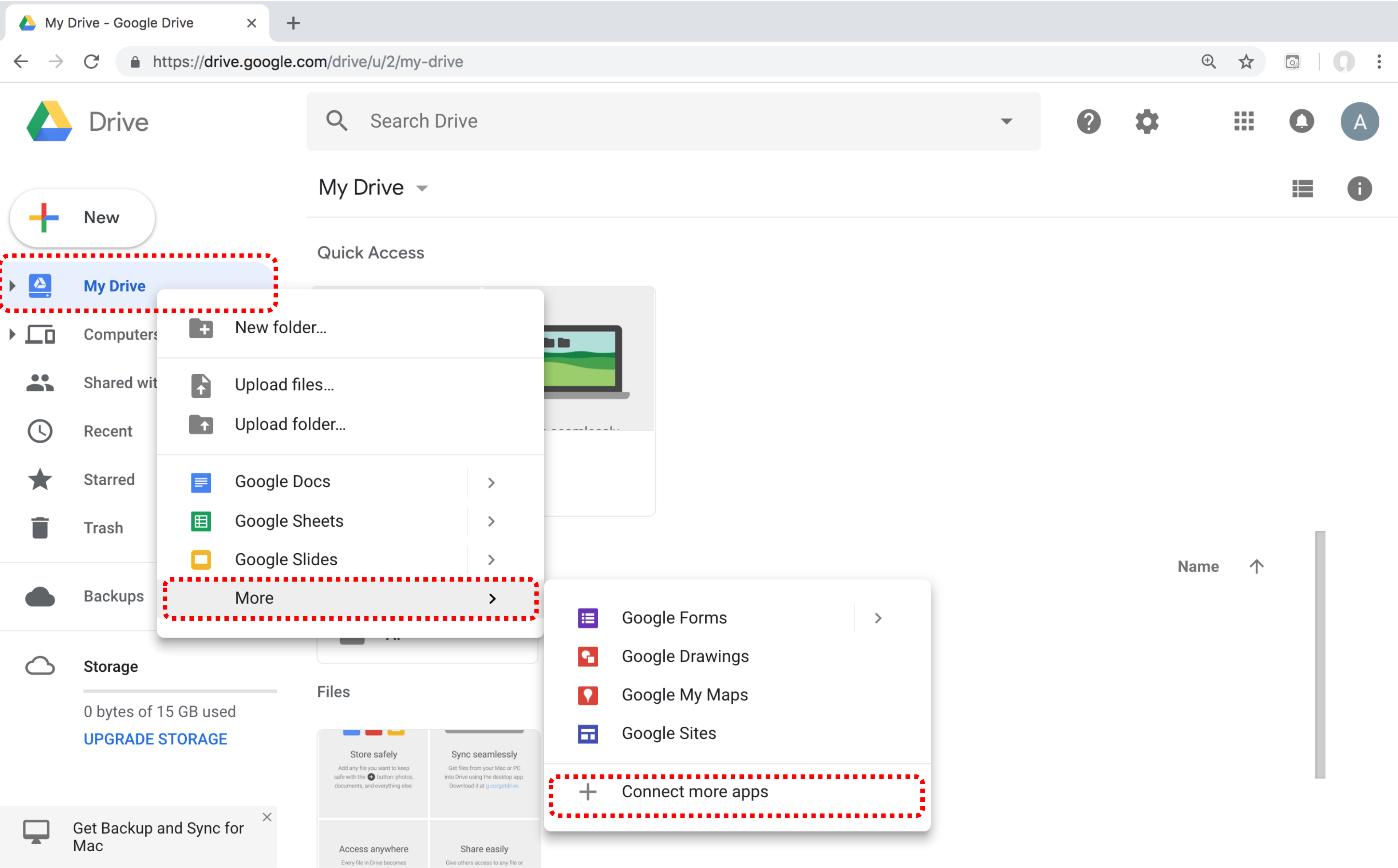
Outline

- Python for
  Natural Language Processing

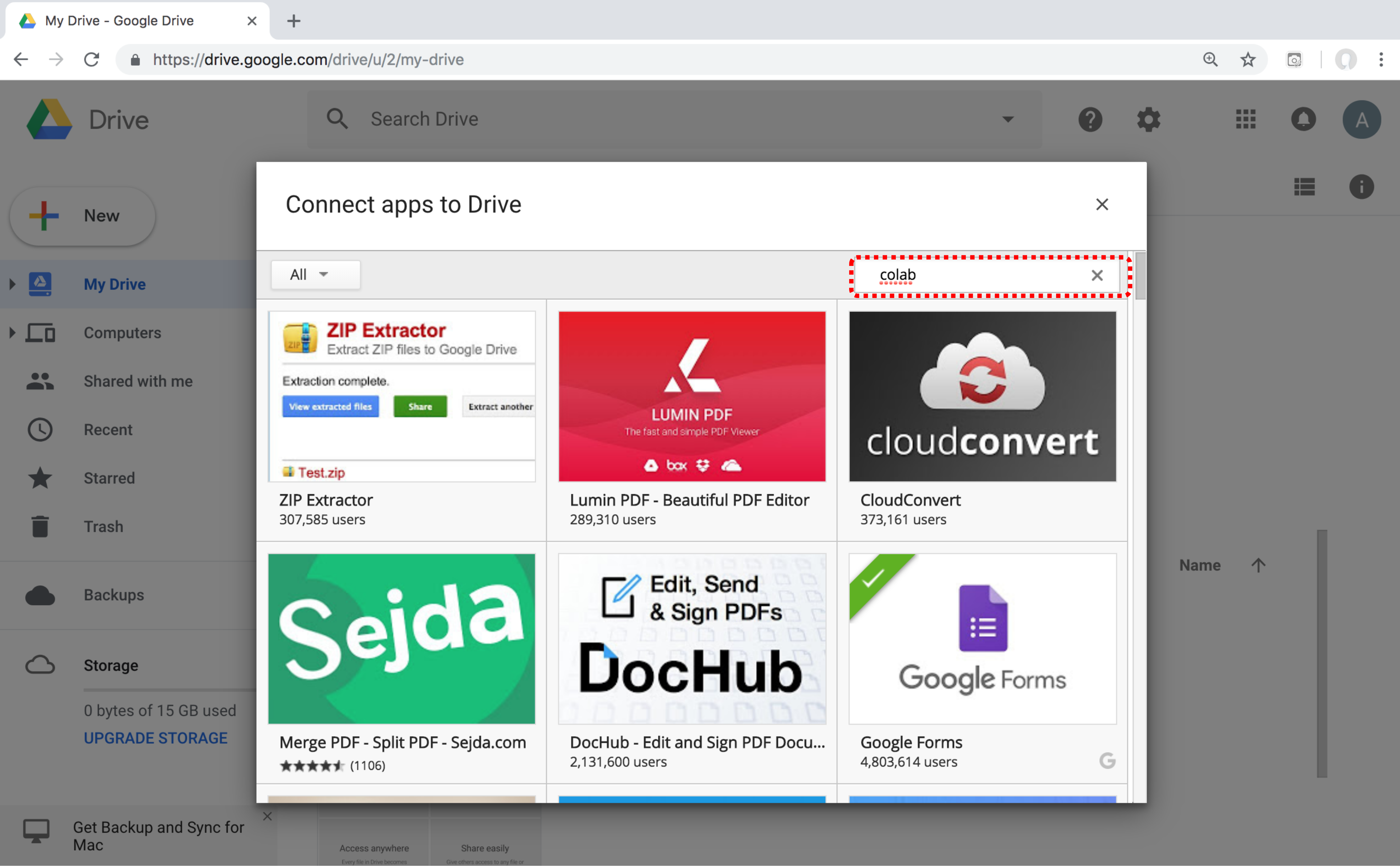
Python for Natural Language Processing

Source: https://www.python.org/community/logos/

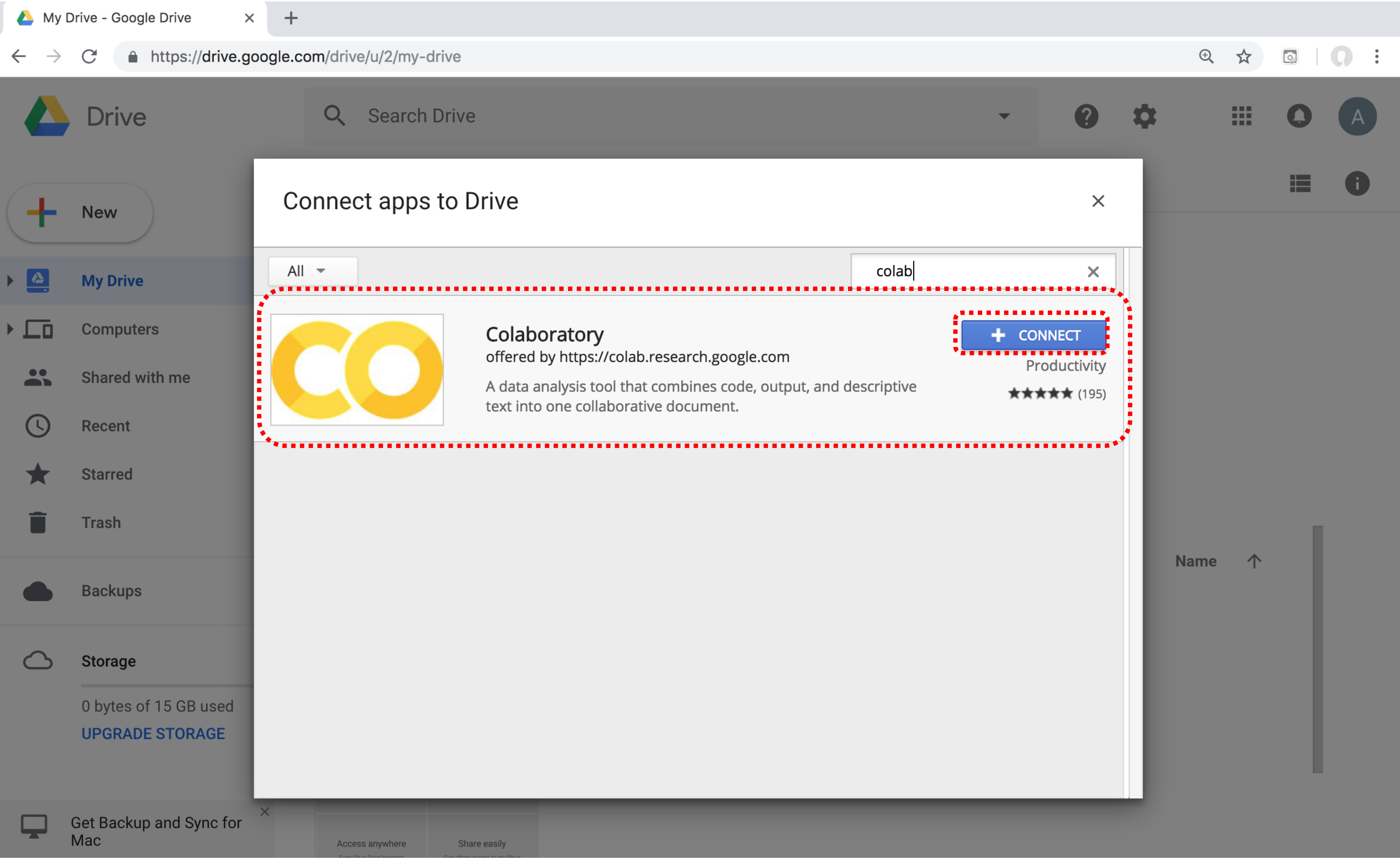
Connect Google Colab in Google Drive



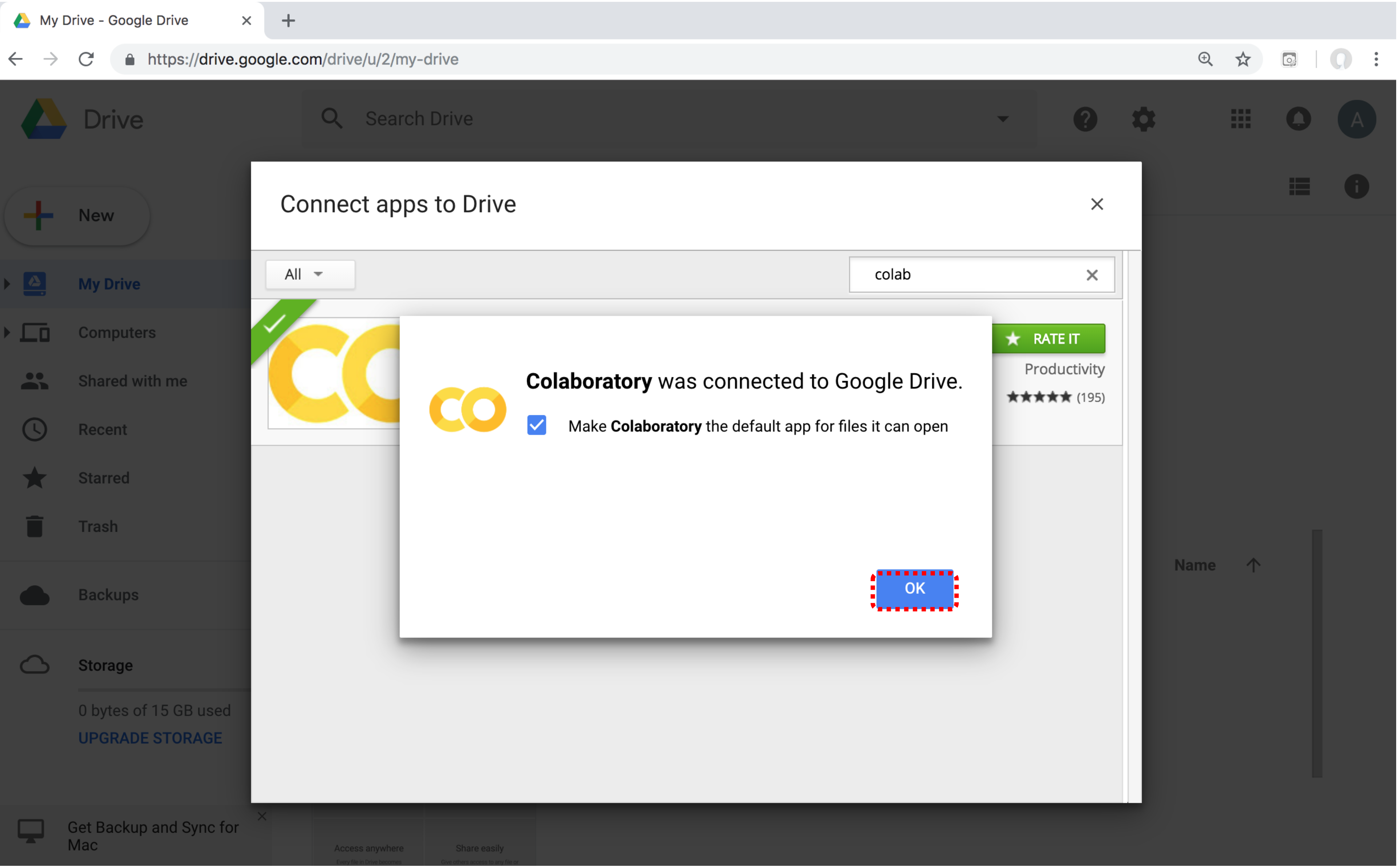
Google Colab



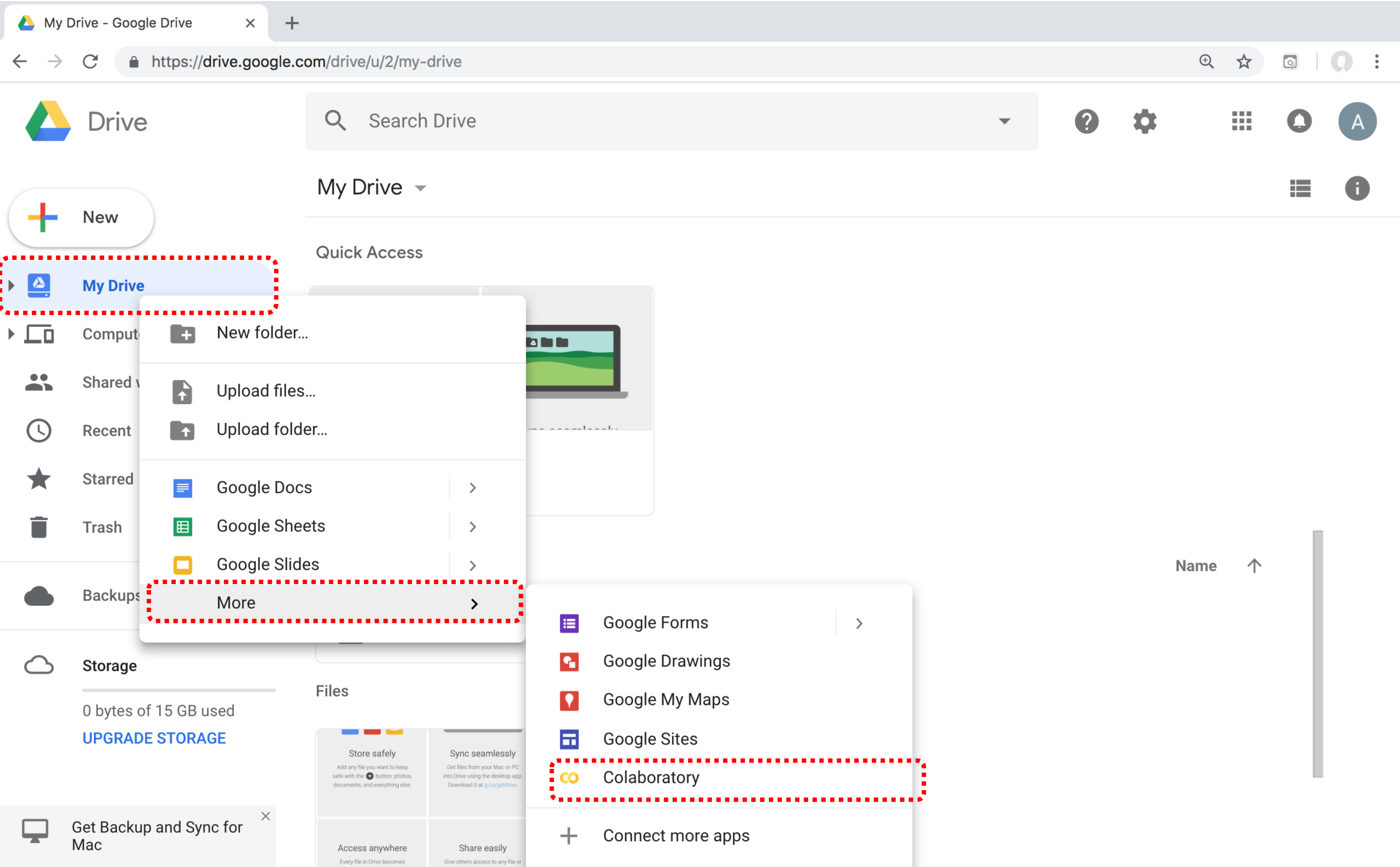
Google Colab



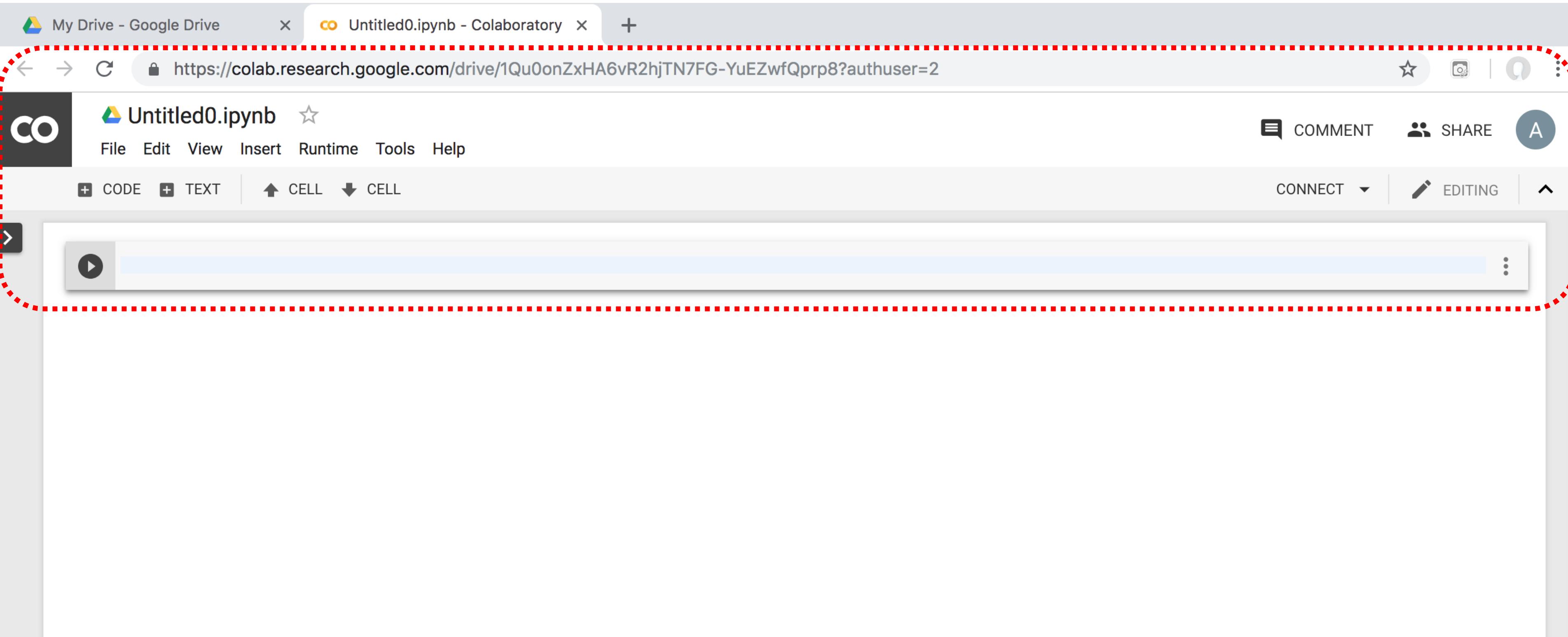
Connect Colaboratory to Google Drive



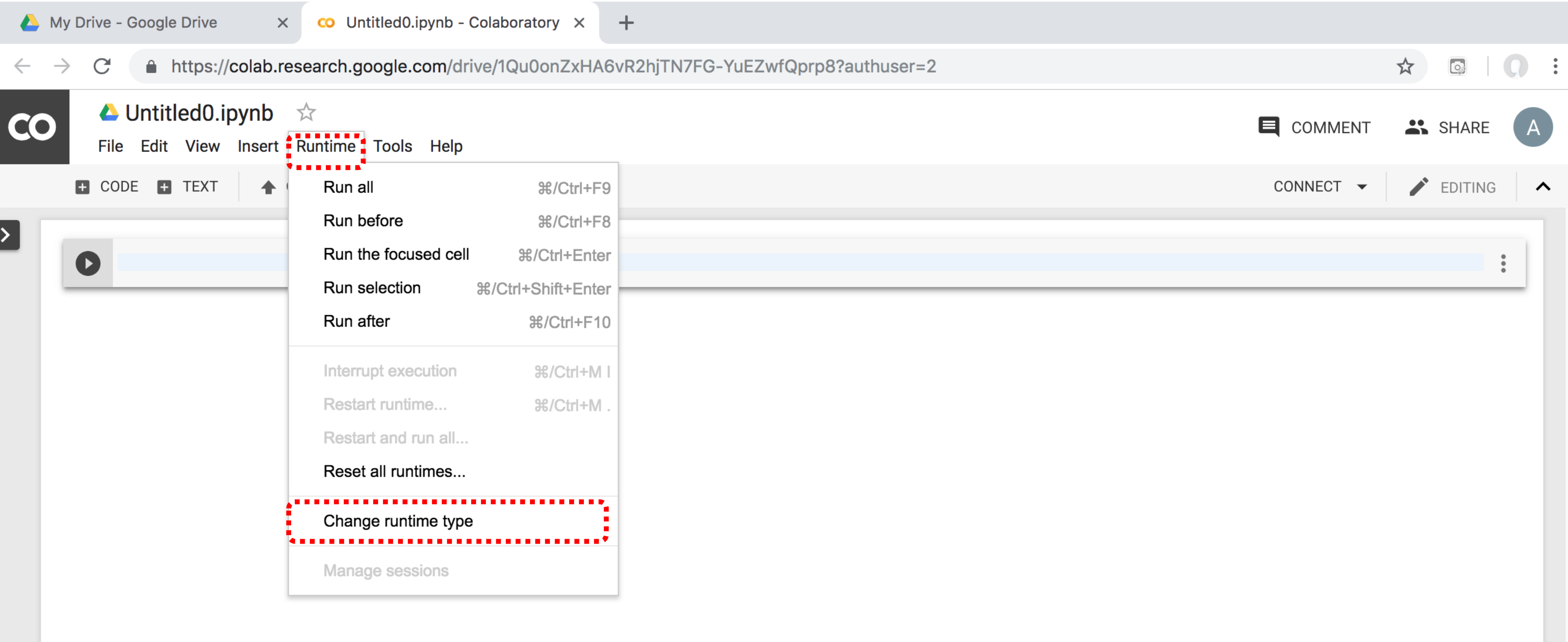
Google Colab



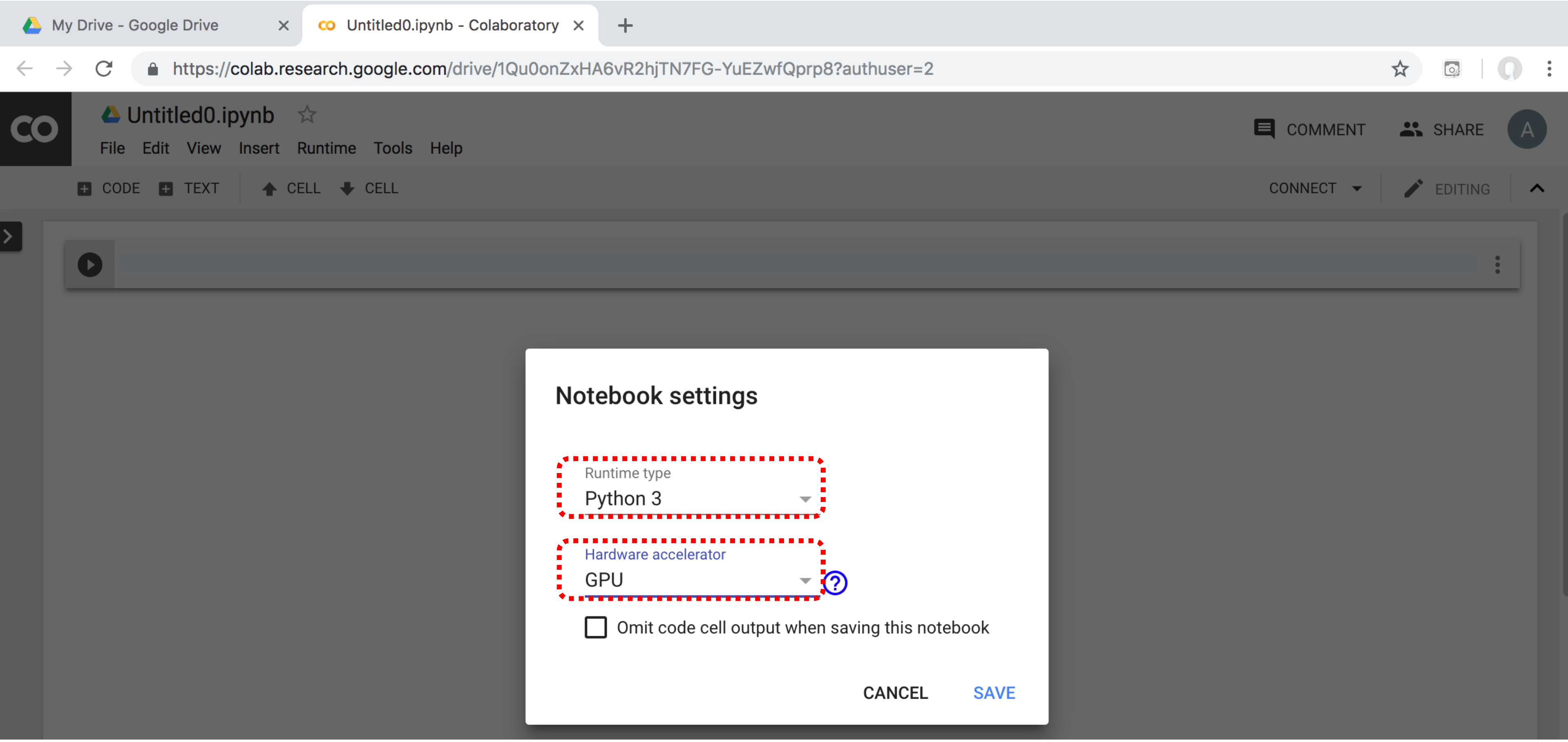
Google Colab



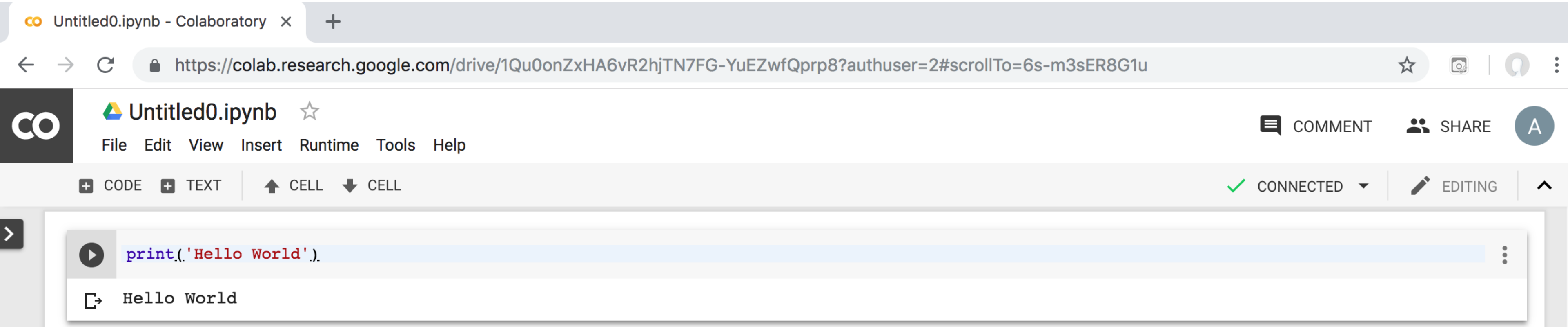
Google Colab



Run Jupyter Notebook Python3 GPU Google Colab



Google Colab Python Hello World
print('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	word's lemma
am → am	am → be
having → hav	having → have

Source: Nitin Hardeniya (2015), NLTK Essentials, Packt Publishing; Florian Leitner (2015), Text mining - from Bayes rule to dependency parsing

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

+ 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

```
#!python -m spacy download en_core_web_sm
```

```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
for token in doc:
    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

+ Code + Text

RAM Disk Editing

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

Comment Share

+ Code + Text

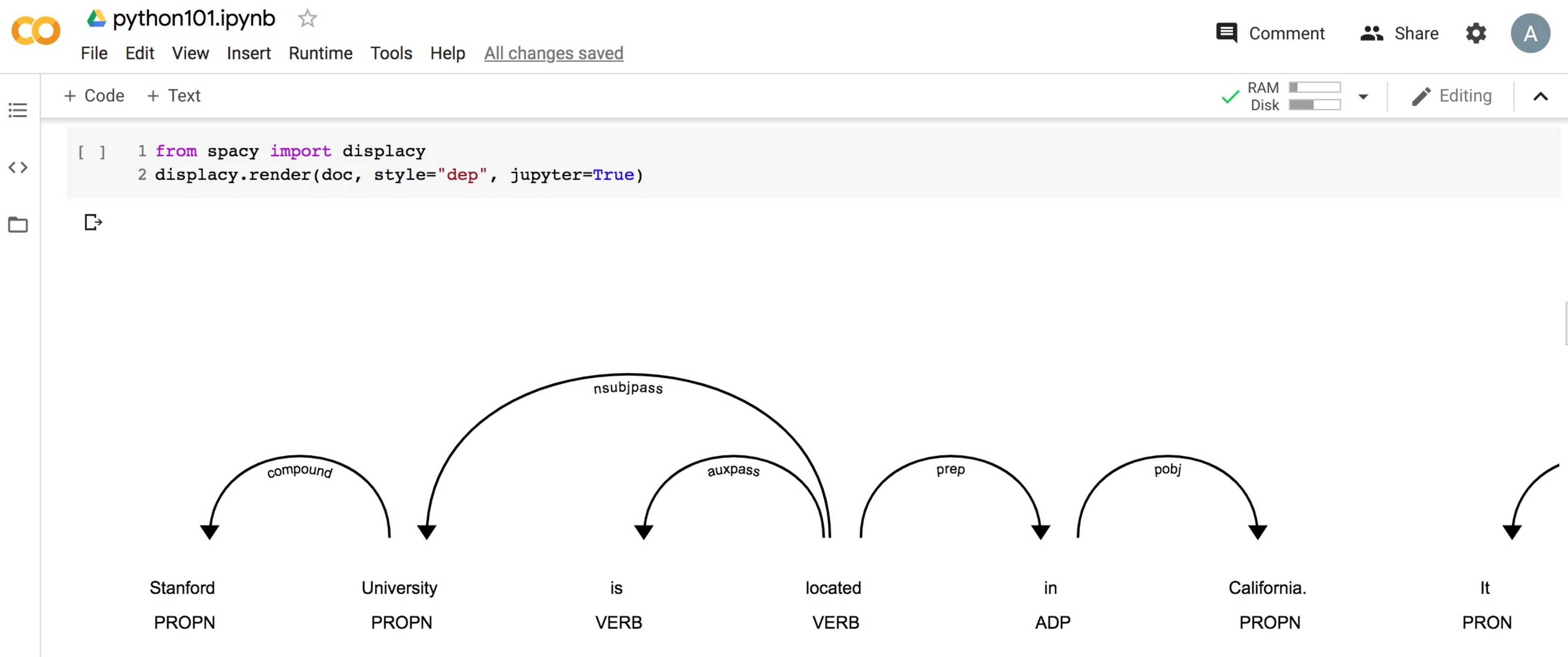
RAM Disk Editing

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



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

```python
import spacy
nlp = spacy.load("en_core_web_sm")
text = "Stanford University is located in California. It is a great university."
doc = nlp(text)
for ent in doc.ents:
    print(ent.text, ent.label_)
```

```
Stanford University ORG
California GPE
```

```python
from spacy import displacy
text = "Stanford University is located in California. It is a great university."
doc = nlp(text)
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

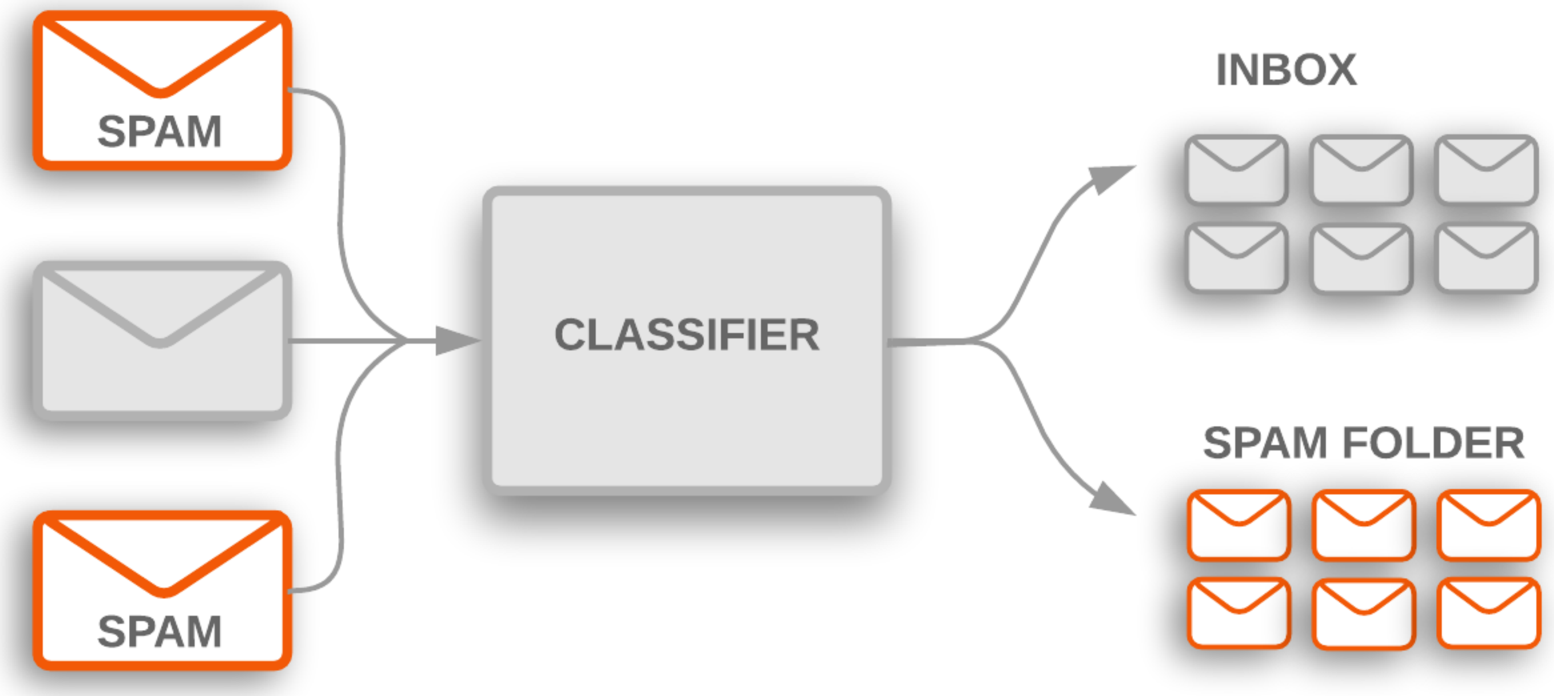
CODE TEXT CELL CELL CONNECT EDITING

Keras preprocessing text

```python
# keras.preprocessing.text 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)
```

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

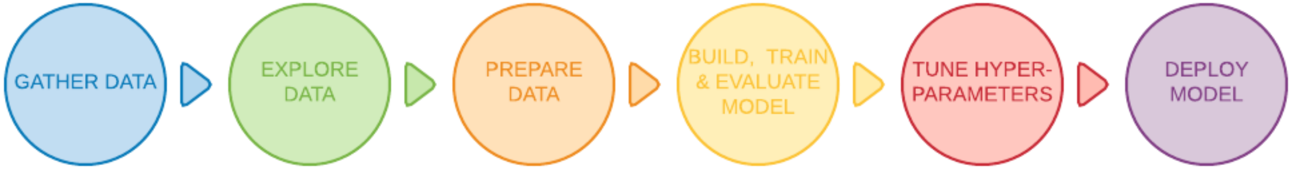
Text Classification



Source: https://developers.google.com/machine-learning/guides/text-classification/

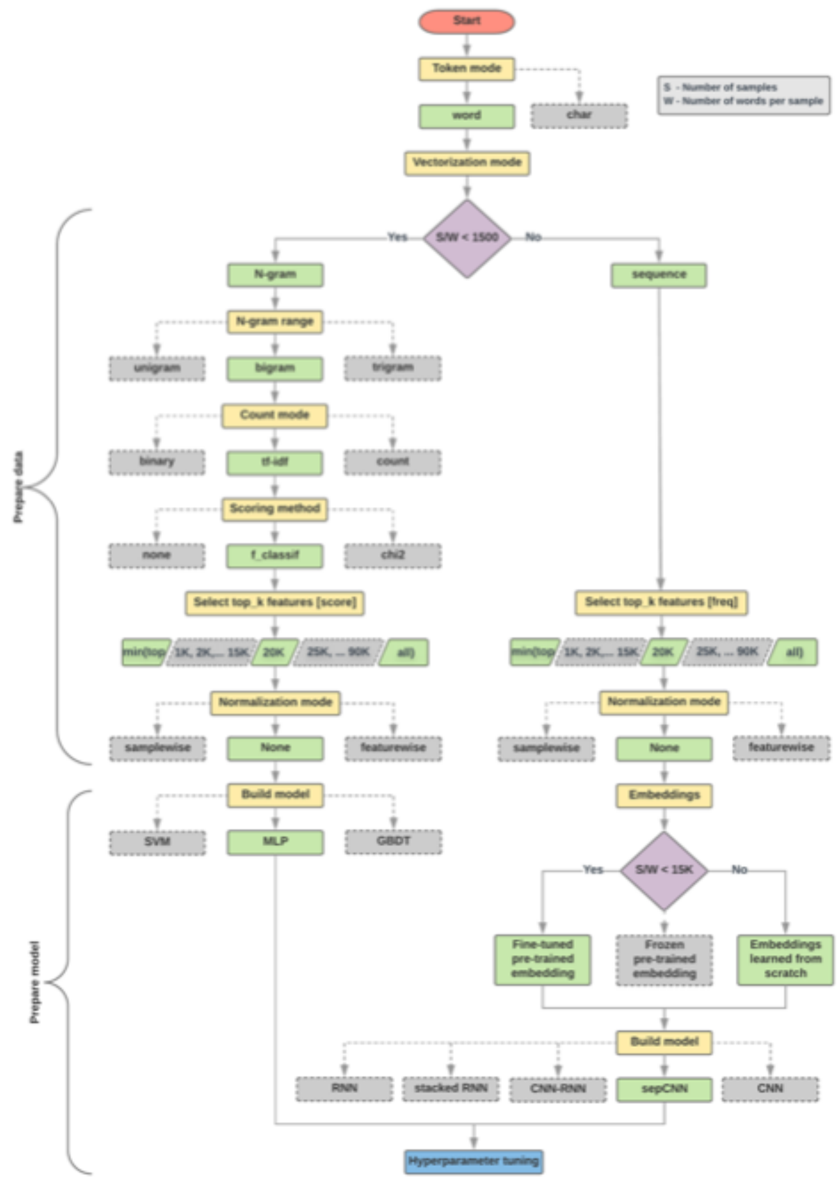
Text Classification Workflow

- 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



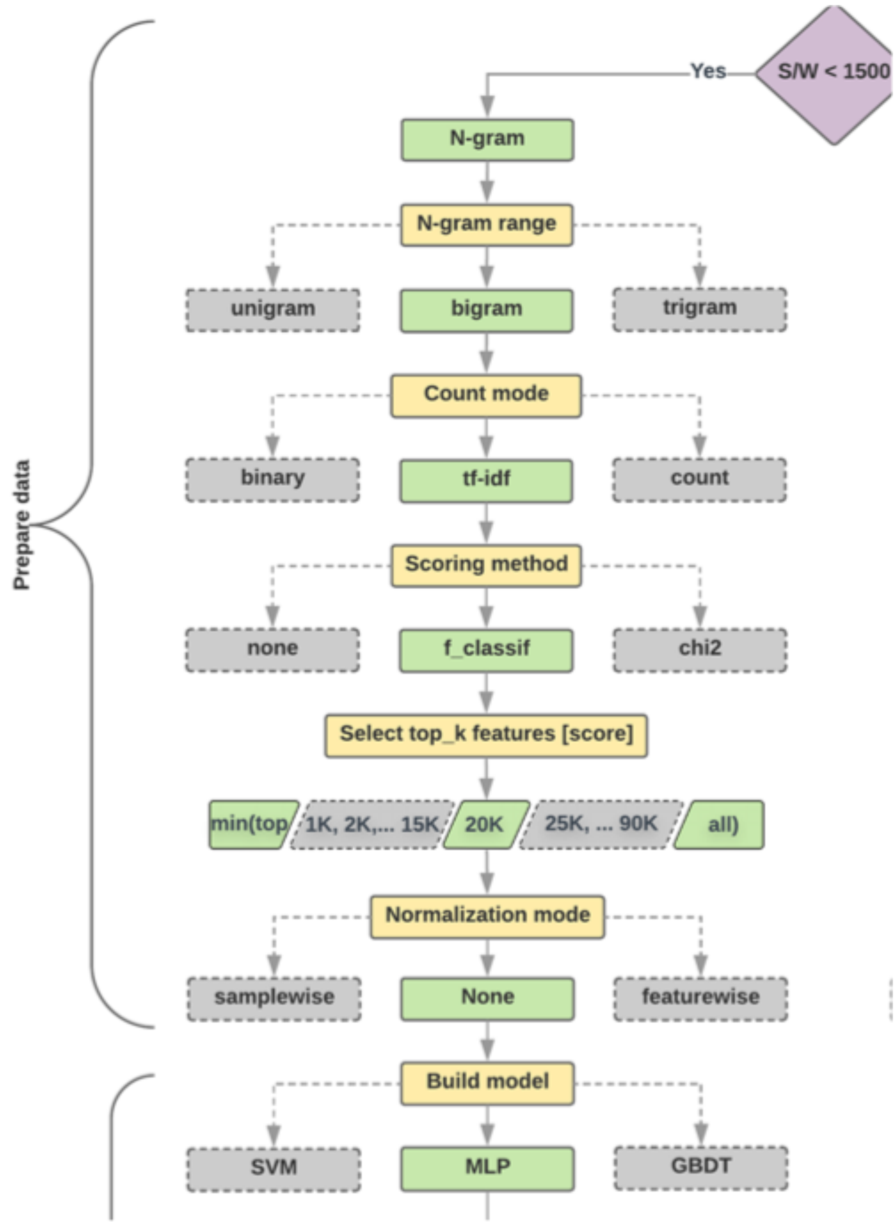
Source: https://developers.google.com/machine-learning/guides/text-classification/

Text Classification Flowchart



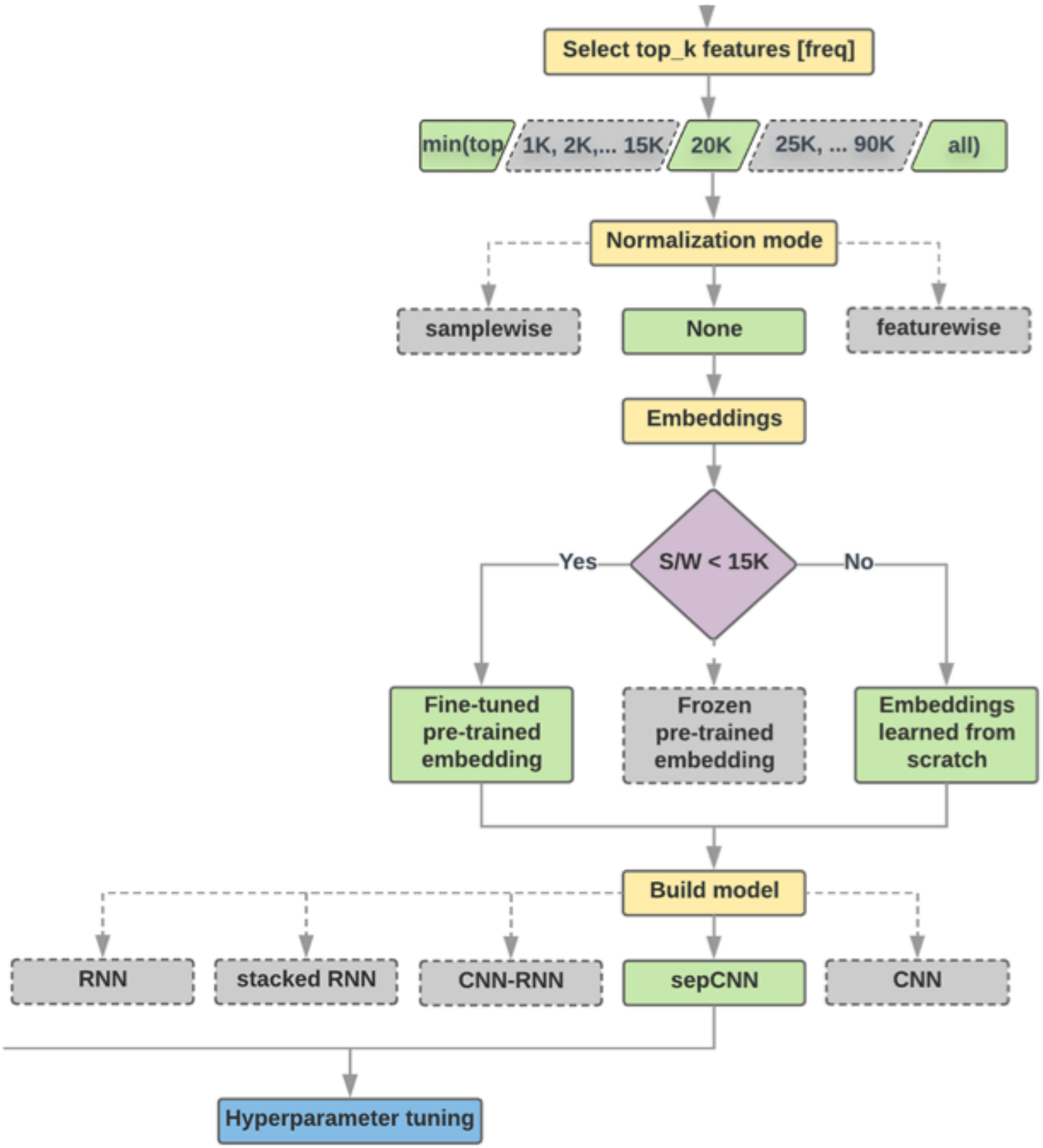
Source: https://developers.google.com/machine-learning/guides/text-classification/step-2-5

Text Classification S/W<1500: N-gram



Source: https://developers.google.com/machine-learning/guides/text-classification/step-2-5

Text Classification S/W>=1500: Sequence

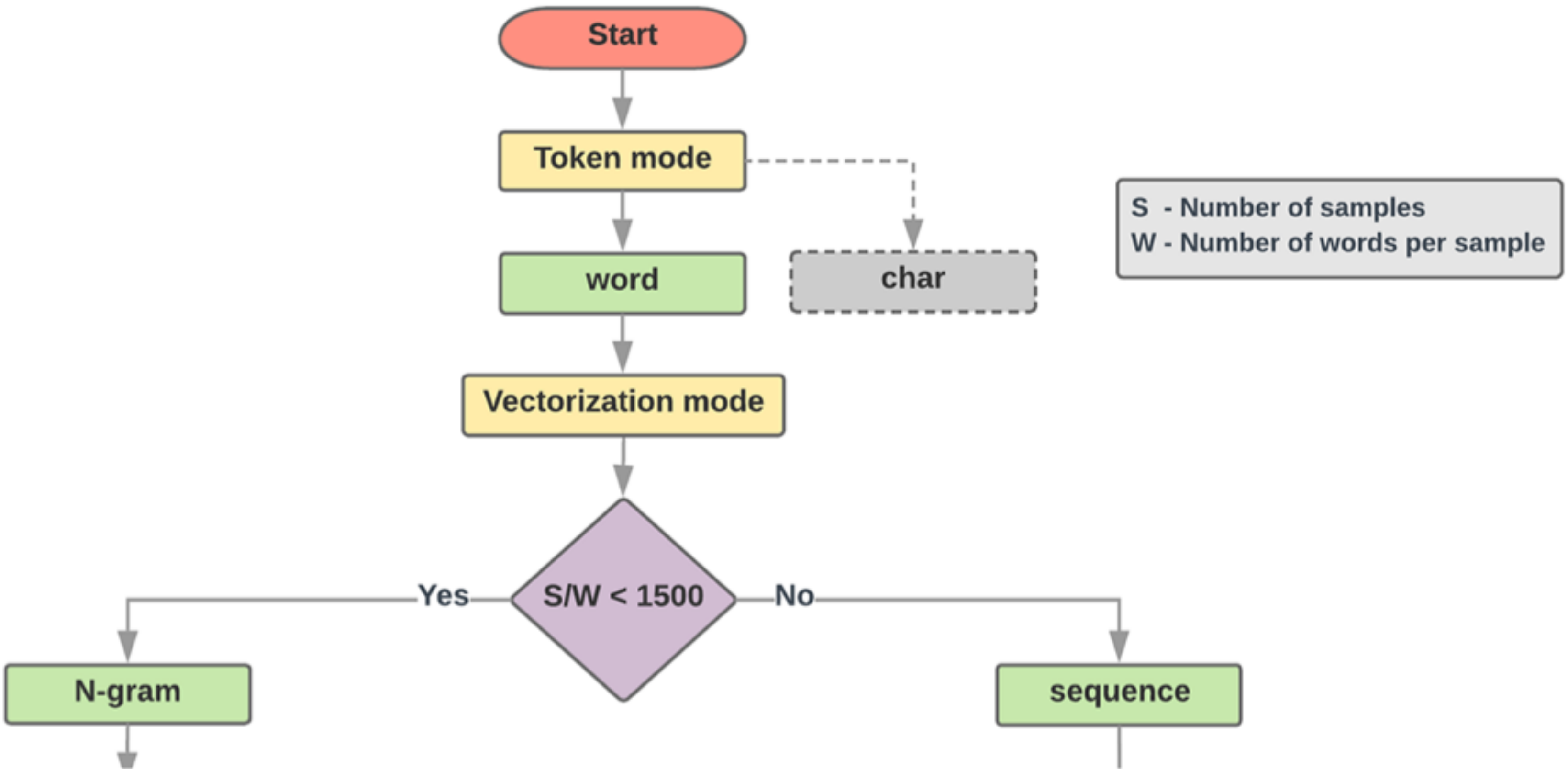


Source: https://developers.google.com/machine-learning/guides/text-classification/step-2-5

Step 2.5: Choose a Model

Samples/Words < 1500

150,000/100 = 1500



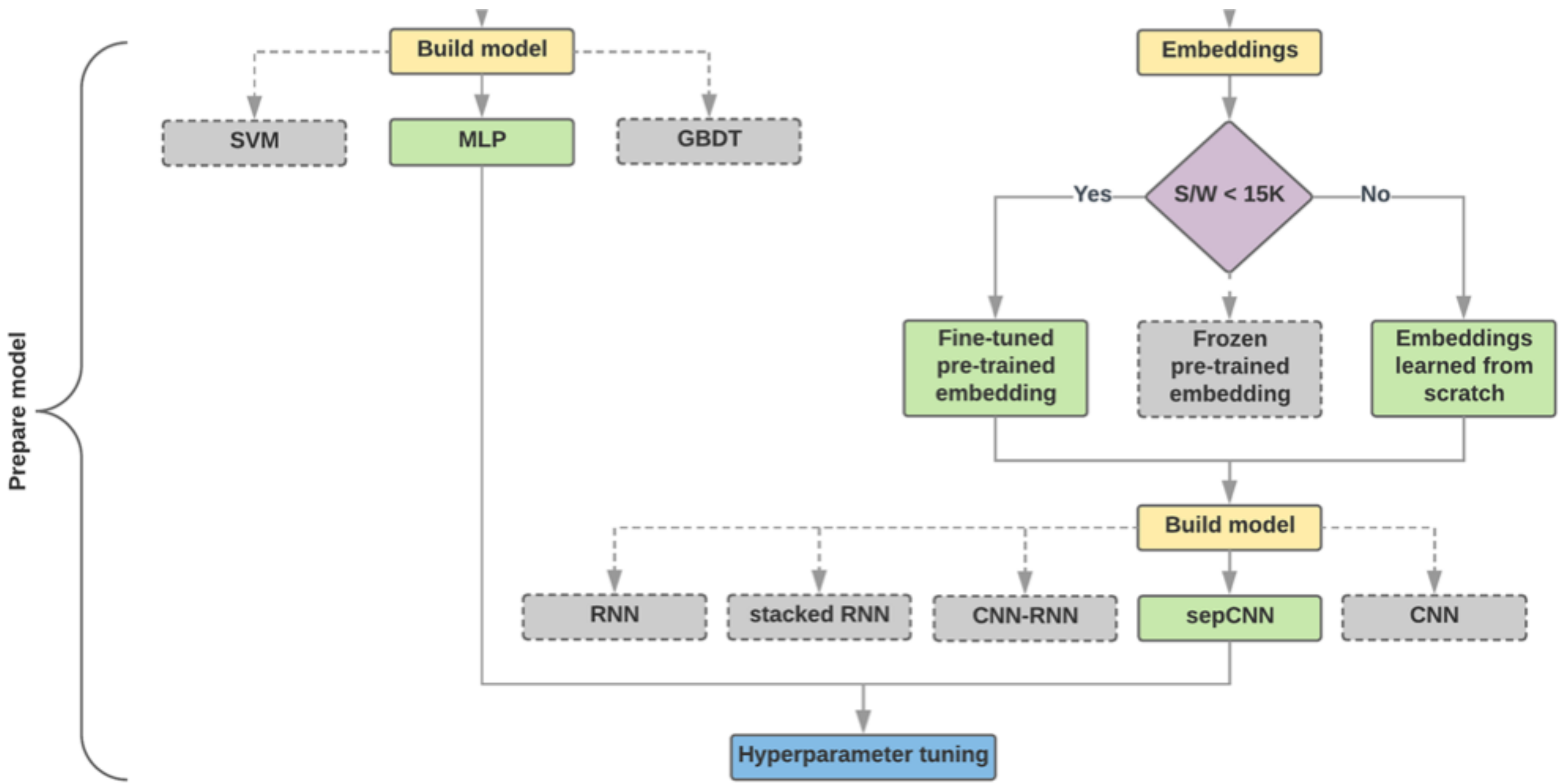
IMDb review dataset,
the samples/words-per-sample ratio is ~ 144

Source: https://developers.google.com/machine-learning/guides/text-classification/step-2-5

Step 2.5: Choose a Model

Samples/Words < 15,000

1,500,000/100 = 15,000



Source: https://developers.google.com/machine-learning/guides/text-classification/step-2-5

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]
T1: 'The mouse ran down' =
        [1,   2,   3,  6]
```

Source: https://developers.google.com/machine-learning/guides/text-classification/step-3

One-hot encoding

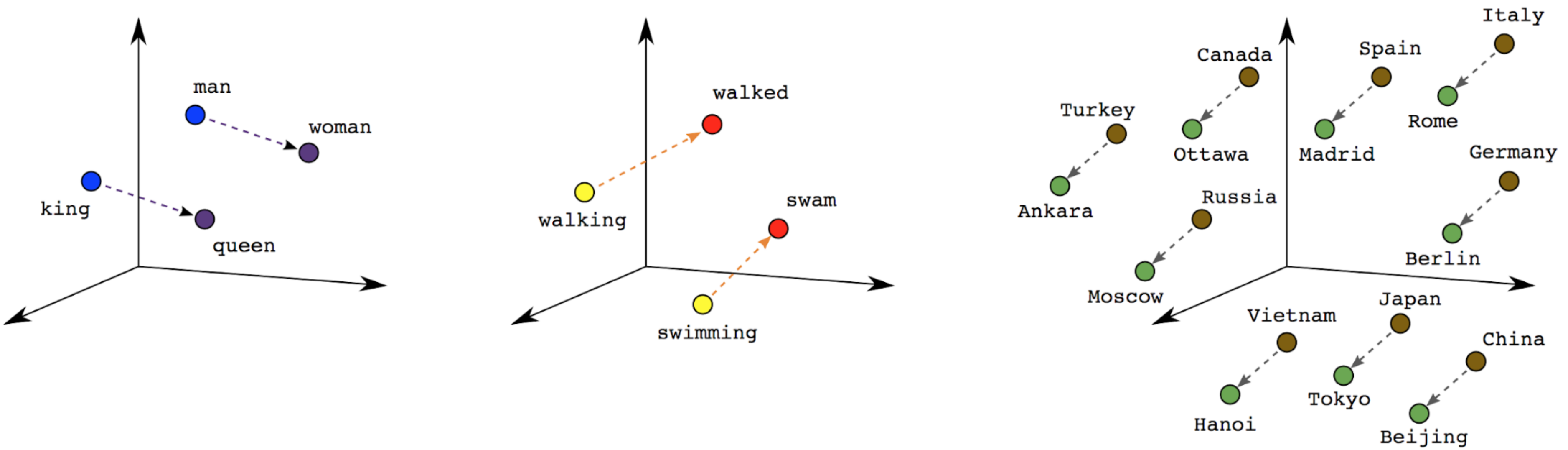
```
'The mouse ran up the clock' =

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]
```

Source: https://developers.google.com/machine-learning/guides/text-classification/step-3

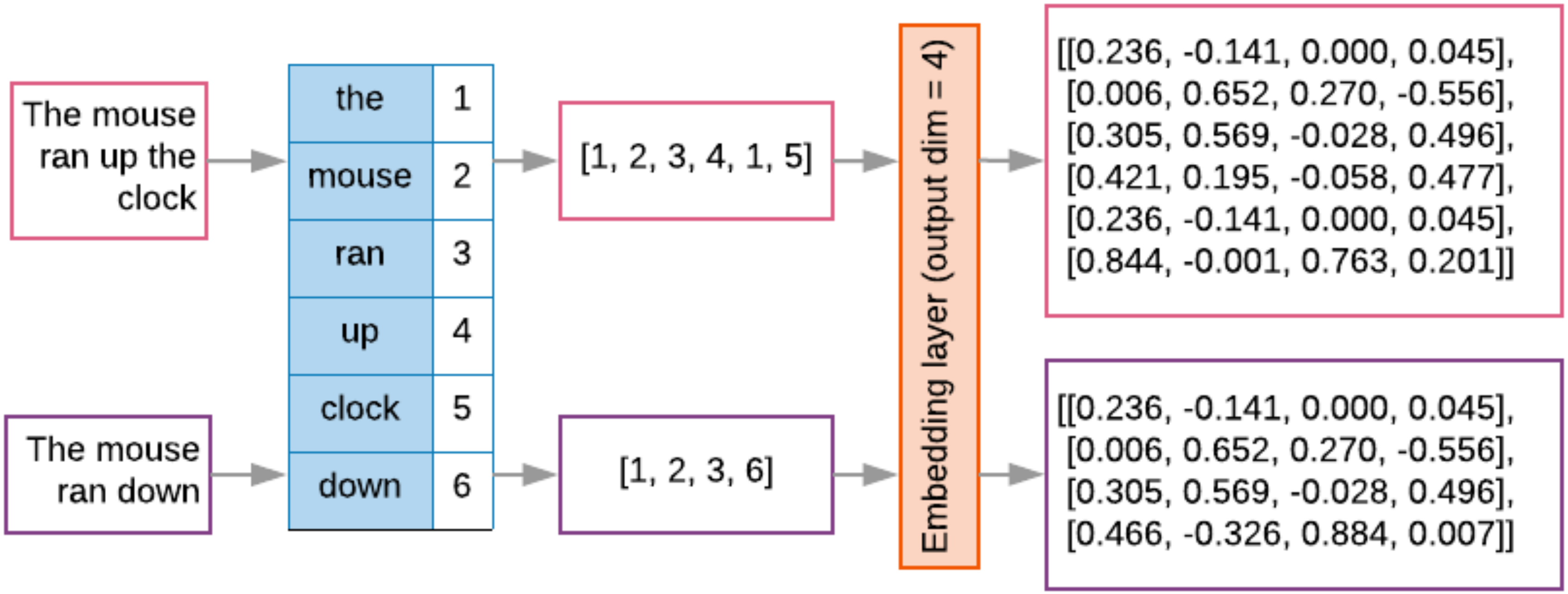
Word embeddings



Male-Female | Verb Tense | Country-Capital

Source: https://developers.google.com/machine-learning/guides/text-classification/step-3

Word embeddings



Source: https://developers.google.com/machine-learning/guides/text-classification/step-3

```python
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)
```


```python
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)
```


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

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

```python
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']
```

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

```python
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
```

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

```python
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
```

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

from keras.preprocessing.text import Tokenizer

```python
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)
```

```
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.]]
```

Source: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

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)
```

Source: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

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.]]
```

Source: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

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]]
```

Source: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

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

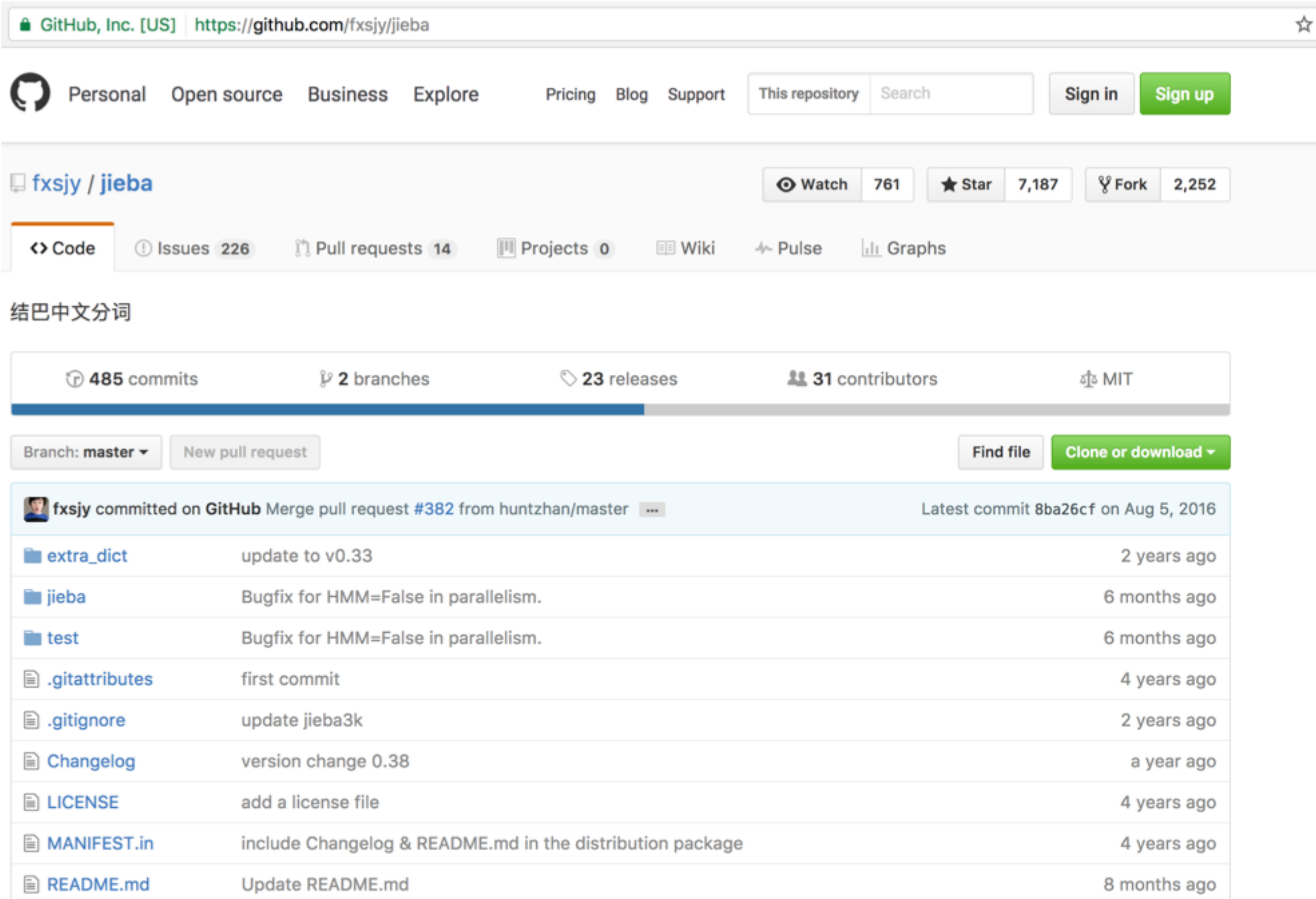
SEARCH

Go

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

http://www.nltk.org/

Python Jieba “结巴”中文分词



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)

```python
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)
```

python101.ipynb

File Edit View Insert Runtime Tools Help

CODE TEXT CELL CELL

```python
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())
```

```
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://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

Python Jieba “结巴”中文分词

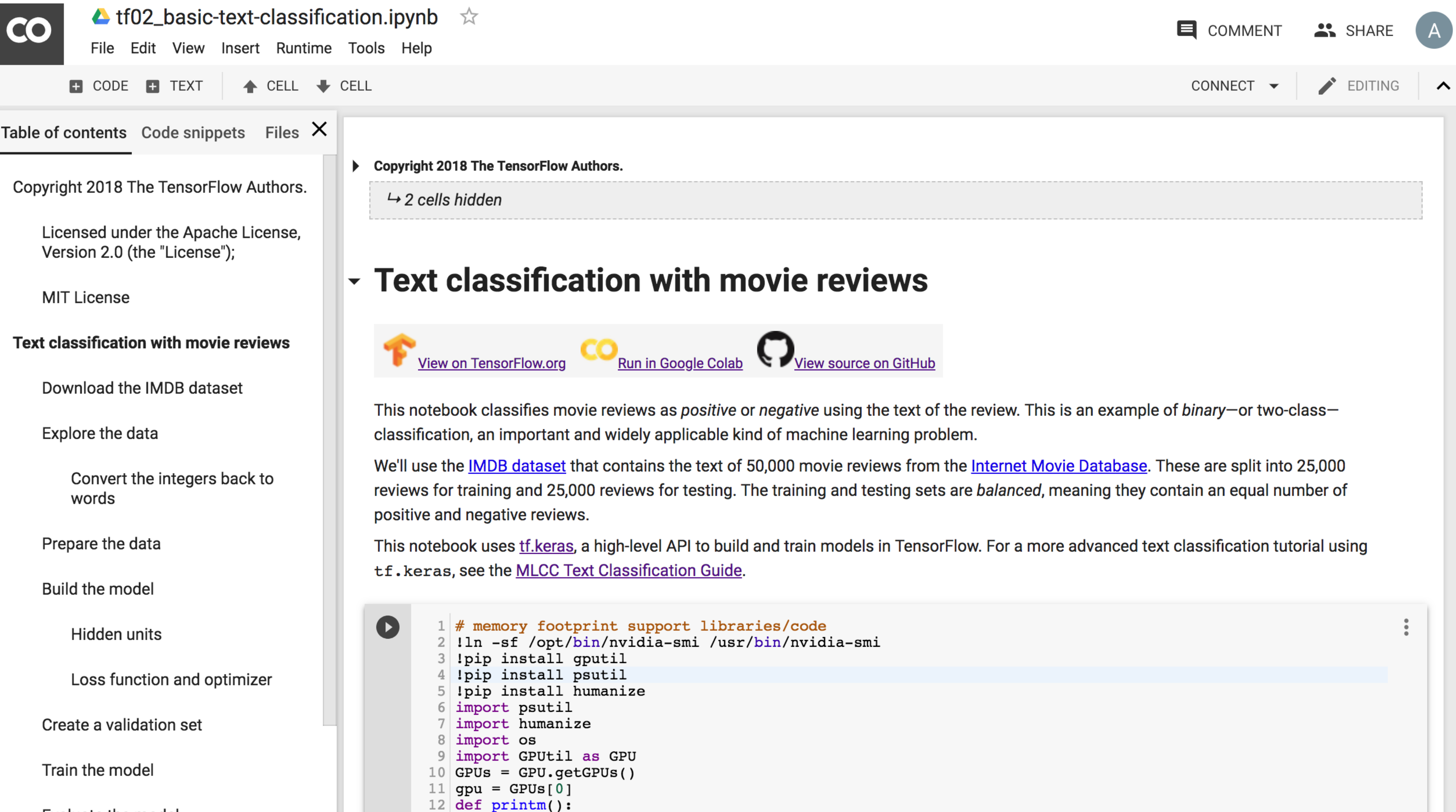
- https://github.com/fxsjy/jieba
- jieba.set_dictionary('data/dict.txt.big')
  - #/anaconda/lib/python3.5/site-packages/jieba
  - dict.txt (5.4MB)(349,046)
  - dict.txt.big.txt (8.6MB)(584,429)
  - dict.txt.small.txt (1.6MB)(109,750)
  - dict.tw.txt (4.2MB)(308,431)
- https://github.com/ldkrsi/jieba-zh_TW
  - 結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/1x16h1GhHsLIrLYtPCvCHaoO1W-i_gror



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.